Data Centre Management

3rd Edition



BIBLIOTEX
Digital Library

Data Centre Management

3rd Edition

BIBLIOTEX
Digital Library
www.bibliotex.com

DATA CENTRE MANAGEMENT

3RD EDITION



www.bibliotex.com
email: info@bibliotex.com

e-book Edition 2023
ISBN: 978-1-98468-211-6 (e-book)

This book contains information obtained from highly regarded resources. Reprinted material sources are indicated. Copyright for individual articles remains with the authors as indicated and published under Creative Commons License. A Wide variety of references are listed. Reasonable efforts have been made to publish reliable data and views articulated in the chapters are those of the individual contributors, and not necessarily those of the editors or publishers. Editors or publishers are not responsible for the accuracy of the information in the published chapters or consequences of their use. The publisher assumes no responsibility for any damage or grievance to the persons or property arising out of the use of any materials, instructions, methods or thoughts in the book. The editors and the publisher have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission has not been obtained. If any copyright holder has not been acknowledged, please write to us so we may rectify.

Notice: Registered trademark of products or corporate names are used only for explanation and identification without intent of infringement.

© 2023 3G E-learning LLC

In Collaboration with 3G E-Learning LLC. Originally Published in printed book format by 3G E-Learning LLC with ISBN 978-1-98467-961-1

EDITORIAL BOARD



Aleksandar Mratinković was born on May 5, 1988. in Arandjelovac, Serbia. He has graduated on Economic high school (2007), The College of Tourism in Belgrade (2013), and also has a master degree of Psychology (Faculty of Philosophy, University of Novi Sad). He has been engaged in different fields of psychology (Developmental Psychology, Clinical Psychology, Educational Psychology and Industrial Psychology) and has published several scientific works.



Dan Piestun (PhD) is currently a startup entrepreneur in Israel working on the interface of Agriculture and Biomedical Sciences and was formerly president-CEO of the National Institute of Agricultural Research (INIA) in Uruguay. Dan is a widely published scientist who has received many honours during his career including being a two-time recipient of the Amit Golda Meir Prize from the Hebrew University of Jerusalem, his areas of expertise includes stem cell molecular biology, plant and animal genetics and bioinformatics. Dan's passion for applied science and technological solutions did not stop him from pursuing a deep connection to the farmer, his family and nature. Among some of his interest and practices counts enjoying working as a beekeeper and onboard fishing.



Hazem Shawky Fouda has a PhD. in Agriculture Sciences, obtained his PhD. From the Faculty of Agriculture, Alexandria University in 2008, He is working in Cotton Arbitration & Testing General Organization (CATGO).



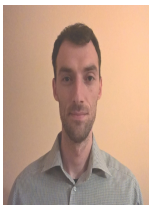
Felecia Killings is the Founder and CEO of LiyahAmore Publishing, a publishing company committed to providing technical and educational services and products to Christian Authors. She operates as the Senior Editor and Writer, the Senior Writing Coach, the Content Marketing Specialist, Editor-in-Chief to the company's quarterly magazine, the Executive and Host of an international virtual network, and the Executive Director of the company's online school for Authors. She is a former high-school English instructor and professional development professor. She possesses a Master of Arts degree in Education and a Bachelor's degree in English and African American studies.



Dr. Sandra El Hajj, Ph.D. in Health Sciences from Nova Southeastern University, Florida, USA is a health professional specialized in Preventive and Global Health. With her 12 years of education obtained from one of the most prominent universities in Beirut, in addition to two leading universities in the State of Florida (USA), Dr. Sandra made sure to incorporate interdisciplinary and multicultural approaches in her work. Her long years of studies helped her create her own miniature world of knowledge linking together the healthcare field with Medical Research, Statistics, Food Technology, Environmental & Occupational Health, Preventive Health and most noteworthy her precious last degree of Global Health. Till today, she is the first and only doctor specialized in Global Health in the Middle East area.



Fozia Parveen has a Dphil in Sustainable Water Engineering from the University of Oxford. Prior to this she has received MS in Environmental Sciences from National University of Science and Technology (NUST), Islamabad Pakistan and BS in Environmental Sciences from Fatima Jinnah Women University (FJWU), Rawalpindi.



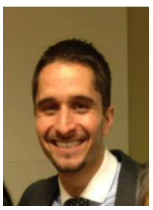
Igor Krunic 2003-2007 in the School of Economics. After graduating in 2007, he went on to study at The College of Tourism, at the University of Belgrade where he got his bachelor degree in 2010. He was active as a third-year student representative in the student parliament.Then he went on the Faculty of science, at the University of Novi Sad where he successfully defended his master's thesis in 2013. The crown of his study was the work titled Opportunities for development of cultural tourism in Cacak". Later on, he became part of a multinational company where he got promoted to a deputy director of logistic. Nowadays he is a consultant and writer of academic subjects in the field of tourism.



Dr. Jovan Pehcevski obtained his PhD in Computer Science from RMIT University in Melbourne, Australia in 2007. His research interests include big data, business intelligence and predictive analytics, data and information science, information retrieval, XML, web services and service-oriented architectures, and relational and NoSQL database systems. He has published over 30 journal and conference papers and he also serves as a journal and conference reviewer. He is currently working as a Dean and Associate Professor at European University in Skopje, Macedonia.



Dr. Tanjina Nur finished her PhD in Civil and Environmental Engineering in 2014 from University of Technology Sydney (UTS). Now she is working as Post-Doctoral Researcher in the Centre for Technology in Water and Wastewater (CTWW) and published about eight International journal papers with 80 citations. Her research interest is wastewater treatment technology using adsorption process.



Stephen obtained his PhD from the University of North Carolina at Charlotte in 2013 where his graduate research focused on cancer immunology and the tumor microenvironment. He received postdoctoral training in regenerative and translational medicine, specifically gastrointestinal tissue engineering, at the Wake Forest Institute of Regenerative Medicine. Currently, Stephen is an instructor for anatomy and physiology and biology at Forsyth Technical Community College.



Michelle holds a Masters of Business Administration from the University of Phoenix, with a concentration in Human Resources Management. She is a professional author and has had numerous articles published in the Henry County Times and has written and revised several employee handbooks for various YMCA organizations throughout the United States.

TABLE OF CONTENTS

Preface xi

CHAPTER 1 OVERVIEW OF DATA CENTER 1

Introduction 2
Perception Of Data Center 3
Data Center Facility 4
Data Center Virtualization 6
Application Architecture Models 7
Data Center Architecture 12
Layer of Data center Architecture 14
Data Center Services 23
IP Infrastructure Services 25
Application Services 28
Security Services 34
Storage Services 39
Business Continuance Infrastructure Services 39
Conclusion 41

CHAPTER 2 SERVER ARCHITECTURE 47

Introduction 48
Basic Concept of Server Architecture 49
Internet Data Center Server 51
Network Attachment 53
Client and Server Packet Processing 59
TCP And Server Applications 68
Segments, Datagrams, and Frames 69
TCP Interactive Traffic Profile 70
TCP Bulk Transfer Traffic Profile 76
Server Availability 78
Processes and Threads 79
TCP and UDP Connections to a Failed Server 81

Status of HTTP Applications 84
Conclusion **85**

CHAPTER 3 APPLICATION ARCHITECTURES **89**

Introduction **90**
Taxonomy Of Applications And Hosted Servers **92**
Integration Of Applications **97**
Enterprise Application Integration 98
Network Design Implications of EAI 99
Multitier Applications 101
Markup Languages: HTML AND XML **103**
HyperText Markup Language or HTML 105
Extensible Markup Language or XML 107
User Agents 108
Web Servers **112**
Server-Side Programming 113
Middleware **116**
Database Access **122**
Network Architecture Considerations **123**
Load Balancing 124
Clustering **126**
Cluster Models 127
Geographical Clustering 128
Security 132
Conclusion **136**

CHAPTER 4 DATA CENTER DESIGN **139**

Introduction **140**
Types Of Server Farms And Data Centers **143**
Internet Server Farms 144
Intranet Server Farms 146
Extranet Server Farms 148
Internet Data Center 149
Corporate Data Center 150
Data Center Topologies **150**
Generic Layer 3/Layer 2 Designs 151
The Need for Layer 2 at the Access Layer 154
Alternate Layer 3/Layer 2 Designs 156
Multiple-Tier Designs **157**
Expanded Multitier Design 159
Collapsed Multitier Design 160

The Need for Layer 2 at the Access Layer 161
Fully Redundant Layer 2 And Layer 3 Designs 162
The Need for Redundancy 162
Layer 2 and Layer 3 in Access Layer 164
Layer 2, Loops, and Spanning Tree 165
Fully Redundant Layer 2 And Layer 3 Designs With Services 168
Additional Services 169
Service Deployment Options 170
Design Considerations With Service Devices 171
Application Environment Trends 173
Application Architecture Trends 174
Network Infrastructure Trends 175
Conclusion 181

CHAPTER 5 DATA CENTER SECURITY 187

Introduction 188
Data Centers: Best Practices For Security And Performance 189
Security 190
Availability 191
Scalability 192
Manageability 193
Physical Security For Data Centers 193
Physical Data Center Security Considerations 201
Physical Data Center Security Systems 205
Importance Of Data Center Security 211
Building Infrastructure & Location 211
Access Control Systems & Biometrics 212
Surveillance 213
Redundancy 213
Conclusion 215

CHAPTER 6 SERVER LOAD-BALANCING 221

Introduction 222
Concept Of Server Load-Balancing 223
DNS-Based Load Balancing 225
Load Balancing Functions 228
Firewall Load Balancing 229
Networking Basics 230
Server Load Balancers 232
Redundancy 234
Provider Infrastructure 238
Anatomy Of A Server Load Balancer 240

Direct Server Return 242
SLB Methods 245
Connections Per Second 246
Total Concurrent Connections 247
Throughput 248
Generic Load Balancer Architecture 251
Critical Components of a Load Balancer 254
IP Address Configuration: Flat-Based SLB Versus NAT-Based SLB 255
One-Armed Versus Two-Armed 257
Conclusion 258

CHAPTER 7 SERVER FARM: IP, TCP, AND UDP 261

Introduction 262
Layers And Protocols 262
IP 266
IP Header 267
Time-To-Live Field 272
TCP 276
Basic Data Transfer 277
Reliability 278
Flow Control 278
Multiplexing 278
Connection 278
TCP Header 279
Source Port and Destination Port Fields 280
UDP 287
UDP Header 288
UDP Header Compression 290
Conclusion 290

CHAPTER 8 HTTP AND RELATED CONCEPTS 295

Introduction 296
Resources and Messages 296
URIs 297
URLs 301
URNs 307
Mime 309
HTTP Overview 311
Components of HTTP-based systems 312
Basic aspects of HTTP 314
What can be controlled by HTTP 316

HTTP flow 317
HTTP Messages 318
APIs based on HTTP 320
HTTP General Header **320**
General Headers 321
Client Request Headers 326
Server Response Headers 334
Entity Headers 338
Request Header **342**
Response Header **343**
Special handling of the Content-Security-Policy: frame-ancestor header 344
Special handling of Content-Security-Policy: frame-ancestor header for Internet Explorer 345
Conclusion **346**

INDEX **349**

PREFACE

Times have changed and the demand for information is higher than ever. Just a few decades ago, society was content with simple TV shows with few channels to choose from. Even communication is easy, and snail mail is widely accepted. However, this is no longer the case. Ever since the internet became an everyday necessity and almost everyone owns a smartphone, we spend most of our waking moments online. Whether for work or socialization, the internet plays a huge role in our lives. Of course, the demand for real-time data transmission is at an all-time high. This need for computers and other network devices to handle these demands is the catalyst for immersion in the modern data center. The management of large amounts of data and hardware systems that are part of a widespread network is data center management. Data centers are the powerhouses of the industrial world. As technological advances continue to transform the way companies do business, the need for data and performance increases. Oftentimes, this significant increase in resources is too much for many organizations to handle internally or with existing corporate infrastructure. Many hours and staff are required to keep up the high demand. Using a data center allows a company to outsource its energy needs. Businesses benefit from data centers in many ways. In particular, they allow a company to focus on its consumer goals rather than technological maintenance.

This edition is thoroughly revised and new topics are added. A separate chapter is also added. The book in this edition shows how data management systems and processes help to provide accurate and reliable data to support strategic decisions and day-to-day operations. It examines all steps to be taken with a view to formulating a strategic data center strategy; leverages awareness of data center developments and technology to be applied to businesses; involves all stakeholders and implements effective cost modeling; analyses efficiency, costs and capabilities to build an integrated data center strategy. This edition is organized into eight chapters.

Chapter 1 presents an overview of data center. A data center is a repository that houses computing facilities like servers, routers, switches and firewalls, as well as supporting components like backup equipment, fire suppression facilities and air conditioning.

Chapter 2 focuses on server architecture. A server is a computer program that provides a service to another computer programs (and its user). In a data center, the physical computer that a server program runs in is also frequently referred to as a server.

Chapter 3 is Application Architectures. The data center network design is based on a proven layered approach, which has been tested and improved over the past several years in some of the largest data center implementations in the world.

Chapter 4 is Data Center Design. The demands on data centers and networks are growing dramatically. Most traditional communications media including telephone, music, film, and television are being reshaped or redefined by the Internet, giving birth to new services such as Voice over Internet Protocol (VoIP) and Internet Protocol Television (IPTV).

Chapter 5 sheds light on Data Center Security. Data center security is the pursuit of practices that make a data center more secure from a range of different kinds of threats and attacks. The data center, as a major primary resource for companies, deserves this kind of dedicated security effort.

Chapter 6 focuses on Server Load-Balancing. Load balancing enhances the performance of the servers, leads to their optimal use, and ensures that no single server is overwhelmed.

Chapter 7 This chapter introduces the fundamental protocols used in intranets and, in particular, the server farms supporting web-based services. A number of other networking devices, such as firewalls, load balancers, and Secure Socket Layer (SSL) offloading devices, are also exposed to the operation of these fundamental protocols.

Chapter 8 presents an overview on HTTP and Related Concepts. This chapter describes the functional and operational aspects of HTTP and presents examples of its use based on traffic captured when real transactions are conducted. This chapter also presents concepts related to HTTP that are fundamental to the full understanding of how HTTP works.

CHAPTER 1

OVERVIEW OF DATA CENTER



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understand perception of data center
2. Explain data center architecture
3. Identify data center services

INTRODUCTION

A data center is a computer system's most important and vulnerable component. At its most basic, it is a physical place that houses a computer network's most critical systems, including backup power supplies, air conditioning, and security applications. We are talking a large amount of data here, stored in a large number of computers.

A typical example of a company that almost certainly has a data center is a bank or other kind of financial institution. A bank's data center will have a mainframe or other kind of computer network, on which customers' account information and other data are stored. A university will also have a similar facility, which includes not only personal information about the university's employees and students, but also information on the university's buildings, construction projects, and physical and intellectual history.

These kinds of data centers contain information that is critical to the continued operation of the bank, university, or other business. Therefore, that data cannot be lost. Security measures surrounding such data centers are usually very strong, as are systems of climate control that keep the center's computer systems from malfunctioning. Data centers will also almost certainly contain backup computers or mirror drives that protect against massive data loss. Such backup computers or mirror drives are routinely dependent not only on electronic power but also on battery power, so that they can continue to function in the case of an interruption of electronic power.



Other kinds of data centers can be found in government institutions; companies that have multiple headquarters; and

providers of electronic services such as television, mobile phones, and the like. A data center can also be a single computer, storing and accessing one company's or one person's critical data. Smaller centers usually have less complicated forms of data protection.

PERCEPTION OF DATA CENTER

A data center is a repository that houses computing facilities like servers, routers, switches and firewalls, as well as supporting components like backup equipment, fire suppression facilities and air conditioning. A data center may be complex (dedicated building) or simple (an area or room that houses only a few servers). Additionally, a data center may be private or shared.

"Wikipedia was offline after an overheating problem at one of its data centers. It was pretty bad. For a while there, people had nowhere to go for phony, inaccurate information."

—Jay Leno



It is a facility composed of networked computers and storage that businesses or other organizations use to organize, process, store and disseminate large amounts of data. A business typically relies heavily upon the applications, services and data contained within a data center, making it a focal point and critical asset for everyday operations.

Data centers are not a single thing, but rather, a conglomeration of elements. At a minimum, data centers serve as the principal repositories for all manner of IT equipment, including servers, storage subsystems, networking switches, routers and firewalls, as well as the cabling and physical racks used to organize and interconnect the IT equipment. A data center must also contain

an adequate infrastructure, such as power distribution and supplemental power subsystems, including electrical switching; uninterruptable power supplies; backup generators and so on; ventilation and data center cooling systems, such as computer room air conditioners; and adequate provisioning for network carrier (telco) connectivity. All of this demands a physical facility with physical security and sufficient physical space to house the entire collection of infrastructure and equipment.

Data Center Consolidation and Colocation

There is no requirement for a single data center, and modern businesses may use two or more data center installations across multiple locations for greater resilience and better application performance, which lowers latency by locating workloads closer to users.

Conversely, a business with multiple data centers may opt to consolidate data centers, reducing the number of locations in order to minimize the costs of IT operations. Consolidation typically occurs during mergers and acquisitions when the majority business does not need the data centers owned by the subordinate business.

Alternatively, **data** center operators can pay a fee to rent server space and other hardware in a colocation facility. Colocation is an appealing option for organizations that want to avoid the large capital expenditures associated with building and maintaining their own data centers. Today, colocation providers are expanding their offerings to include managed services, such as interconnectivity, allowing customers to connect to the public cloud.

Data Center Facility

A Data Center Facility (also called a data storage center, datacenter, data storage facility or data storage vault) is a building which houses mission-critical systems such as computing and communications hardware.

Data center facility generally operates on networks using IP protocols, with redundant connectivity provided from more than one network communications provider.

Security elements designed to prevent intrusion threats to data center facility include:

- Virtual private networks (VPNs)
- Intrusion detection systems
- Firewalls

Data backup and restore systems which employ the use of a disaster-hardened data center facility typically offer the most secure form of backup, since all data resides offsite and is therefore protected from theft or physical damage.

Types of Data Center Facilities

However, many different types of facilities and service models exist. The following list will help clarify the main differences:

- In-House Data Center – Many enterprises, especially larger organizations and those in the technology industry, design, build and operate their own facilities. The capabilities of an in-house facility will depend on the investment the firm is willing and able to make.
- Colocation – One example of a multi-tenant data center, colocation space can be sold to enterprises by the rack, cabinet or cage. Companies of all types and sizes, from small- and medium-sized businesses to *Fortune* 500 firms, benefit from colocation services. Customers still maintain control over their hardware, but outsource facility and internal systems maintenance to the provider.
- Wholesale Data Center – These providers sell data center space in larger capacities than a colocation model and typically have fewer customers. They may build-to-suit space for a single tenant. The concept is similar to leasing a warehouse or office space in which the landlord provides facility maintenance to the tenant.
- Dedicated Hosting – The provider operates and/or rents server capacity to single customers. In other words, servers are not shared among multiple customers. No additional services are provided and the customer maintains full control over the server beyond maintenance. However, some providers can add amenity services, such as remote hands to re-boot servers and upgrade software.

- Managed Hosting – In a managed hosting facility, the provider operates servers and storage for its customers, as well as provides additional administrative and engineering services. Although the scope of services can be diverse and extensive among the various providers, some common examples include database administration, operating system administration, managed security services, managed storage, application management services, disaster recovery, systems monitoring and remote management. The hardware may be owned by the customer or the provider.
- Shared Hosting – In this example, customers share server capacity. Well-known website hosting companies, including Go Daddy and **Network** Solutions, represent examples of providers operating shared hosting facilities. To deploy services, these providers create a user interface overlaying the physical server. This interface provides multi-tenant applications to help customers configure their services.

Data Center Virtualization

Data center virtualization is a method of moving information storage from physical servers to virtual ones, often in a different location. In the past, large companies would keep physical servers on site that held huge amounts of corporate information. These servers were expensive, both to purchase and maintain. With data center virtualization, it became possible to separate both the hardware and location from the data. This cuts costs and increases the data's availability.

Data center virtualization actually comes from a combination of two different technologies; high-speed data transfer and server virtualization. Without both of these components, data center virtualization becomes highly impractical. While high-speed data transfer has been around for a long time, server virtualization did not become practical until the beginning of the 21st century.

High-speed data transfer allows for the separation of information from location. With traditional data centers, information was stored on site, so it was always accessible to workers. As the Internet increased in speed, it became possible

to retrieve information from distant locations nearly as quickly as from a home server. With this type of speed on hand, it is possible to keep your data center nearly anywhere there is a good Internet connection.



Server virtualization centers on using software to do the job of hardware. A single computer running a single server will always have periods where it is sitting idle or doing very little. By running server virtualization software on that single machine, it can act as if it is two or more completely independent servers. Now when one server is idle, the other ones can still be working. This cuts down inefficiency and reduces the number of machines necessary to do tasks.

The business applications for data center virtualizations are numerous. Data can exist in multiple locations at the same time. When changes are made to a database, the change can migrate through a series of virtual servers until all have the new data. Instead of databases being offline when the server needs a repair, the company can simply access an off-site location. If one of the off-site locations becomes inaccessible, there are other servers available with the same information.

Data center virtualization costs less than traditional data storage as well. On the local front, the company has fewer servers to maintain, cutting hardware and staff costs. The off-site location has a series of virtual servers. These machines are doing the work of many physical servers, which cuts their cost as well.

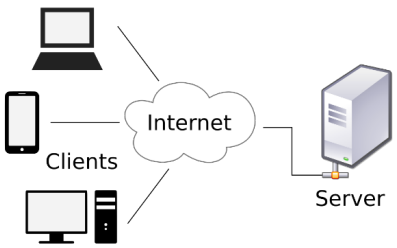
Application Architecture Models

Application architectures are constantly evolving, adapting to new requirements, and using new technologies. The most pervasive

models are the client/server and n-tier models that refer to how applications use the functional elements of communication exchange. The client/server model, in fact, has evolved to the n-tier model, which most enterprise **software** application vendors currently use in application architectures. This section introduces both models and the evolutionary steps from client/server to the n-tier model.

Client/server model

The client-server model is a distributed communication framework of network processes among service requestors, clients and service providers. The client-server connection is established through a network or the Internet.



The client-server model is a core network computing concept also building functionality for email exchange and Web/database access. Web technologies and protocols built around the client-server model are:

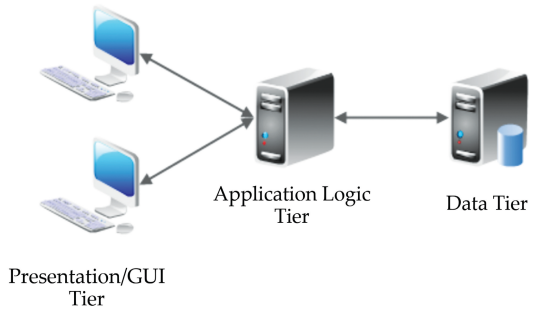
- Hypertext Transfer Protocol (HTTP)
- Domain Name System (DNS)
- Simple Mail Transfer Protocol (SMTP)
- Telnet

Clients include Web browsers, chat applications, and email software, among others. Servers include Web, database, application, chat and email, etc.

N-Tier Architecture

N-tier architecture is also called multi-tier architecture because the software is engineered to have the processing, data management, and presentation functions physically and logically separated.

That means that these different functions are hosted on several machines or clusters, ensuring that services are provided without resources being shared and, as such, these services are delivered at top capacity. The "N" in the name n-tier architecture refers to any number from 1.



Not only does your software gain from being able to get services at the best possible rate, but it is also easier to manage. This is because when you work on one section, the changes you make will not affect the other functions. And if there is a problem, you can easily pinpoint where it originates.

Benefits of N-Tier Architecture

There are several benefits to using n-tier architecture for your software. These are scalability, ease of management, flexibility, and security.

- *Secure:* You can secure each of the three tiers separately using different methods.
- *Easy to manage*: You can manage each tier separately, adding or modifying each tier without affecting the other tiers.
- *Scalable*: If you need to add more resources, you can do it per tier, without affecting the other tiers.
- *Flexible*: Apart from isolated scalability, you can also expand each tier in any manner that your requirements dictate.

In short, with n-tier architecture, you can adopt new technologies and add more components without having to rewrite the entire application or redesigning your whole software, thus making it easier to scale or maintain. Meanwhile, in terms of

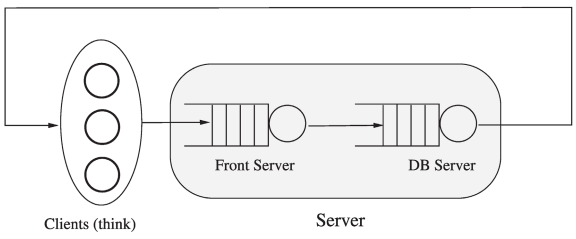
security, you can store sensitive or confidential information in the logic tier, keeping it away from the presentation tier, thus making it more secure.

Other benefits include:

- *More efficient development*: N-tier architecture is very friendly for development, as different teams may work on each tier. This way, you can be sure the design and presentation professionals work on the presentation tier and the database experts work on the data tier.
- *Easy to add new features*: If you want to introduce a new feature, you can add it to the appropriate tier without affecting the other tiers.
- *Easy to reuse*: Because the application is divided into independent tiers, you can easily reuse each tier for other software projects. For instance, if you want to use the same program, but for a different data set, you can just replicate the logic and presentation tiers and then create a new data tier.

Multi-Tier Model

The multi-tier data center model is dominated by HTTP-based applications in a multi-tier approach. The multi-tier approach includes web, application, and database tiers of servers. Today, most web-based applications are built as multi-tier applications. The multi-tier model uses software that runs as separate processes on the same machine using interposes communication (IPC), or on different machines with communications over the network. Typically, the following three tiers are used:



- Web-server
- Application
- Database

Multi-tier server farms built with processes running on separate machines can provide improved resiliency and security. Resiliency is improved because a server can be taken out of service while the same function is still provided by another server belonging to the same application tier. Security is improved because an attacker can compromise a web server without gaining access to the application or database servers. Web and application servers can coexist on a common physical server; the database typically remains separate.

Resiliency is achieved by load balancing the network traffic between the tiers, and security is achieved by placing firewalls between the tiers. You can achieve segregation between the tiers by deploying a separate infrastructure composed of aggregation and access switches, or by using VLANs.

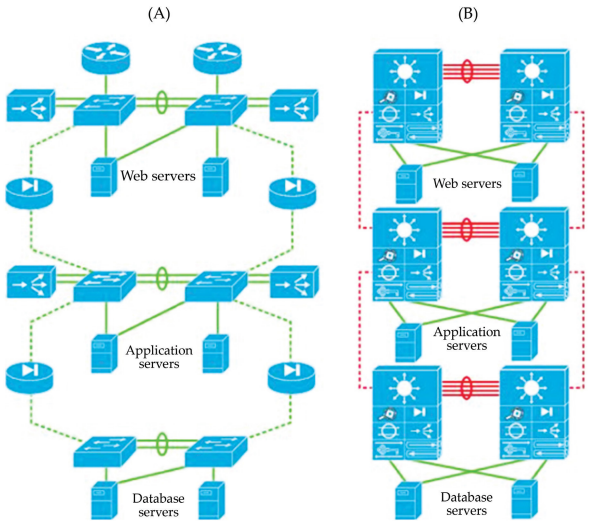


Figure 1: Physical Segregation in a Server Farm with Appliances (A) and Service Modules (B).

The design shown in Figure 2 uses VLANs to segregate the server farms. The left side of the illustration (A) shows the physical topology, and the right side (B) shows the VLAN allocation across the service modules, firewall, load balancer, and switch. The firewall and load balancer, which are VLAN-aware, enforce the VLAN segregation between the server farms. Note that not all of the VLANs require load balancing. For example, the database in the example sends traffic directly to the firewall.

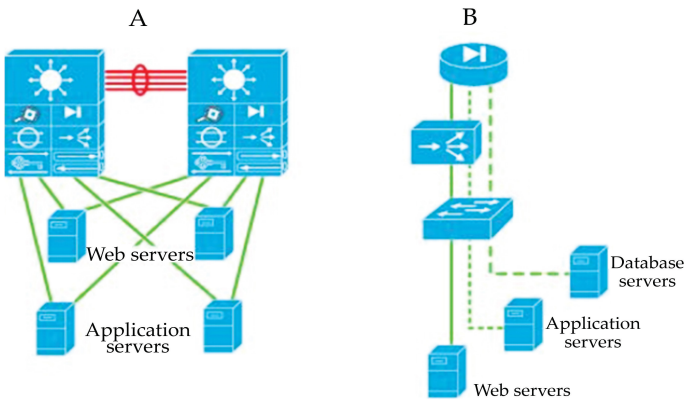


Figure 2: Logical Segregation in a Server Farm with VLANs.

Physical segregation improves performance because each tier of servers is connected to dedicated hardware. The advantage of using logical segregation with VLANs is the reduced complexity of the server farm. The choice of physical segregation or logical segregation depends on your specific network performance requirements and traffic patterns.

Business security and performance requirements can influence the security design and mechanisms used. *For example*, the use of wire-speed ACLs might be preferred over the use of physical firewalls. Non-intrusive security devices that provide detection and correlation, such as the Cisco Monitoring, Analysis, and Response System (MARS) combined with Route Triggered Black Holes (RTBH) and Cisco Intrusion Protection System (IPS) might meet security requirements. Cisco Guard can also be deployed as a primary defense against distributed denial of service (DDoS) attacks.

DATA CENTER ARCHITECTURE

The data center is home to the computational power, storage, and applications necessary to support an enterprise business. The data center infrastructure is central to the IT architecture, from which all content is sourced or passes through. Proper planning of the data center infrastructure design is critical, and performance, resiliency, and scalability need to be carefully considered.

"They're trying to put data centers in cold environments because they're actually generating so much heat now; they're using up so much electricity."

—Rick Smolan

Another important aspect of the data center design is flexibility in quickly deploying and supporting new services. Designing a flexible architecture that has the ability to support new applications in a short time frame can result in a significant competitive advantage. Such a design requires solid initial planning and thoughtful consideration in the areas of port density, access layer uplink bandwidth, true server capacity, and oversubscription, to name just a few.

Data Center connected to the Internet and supporting an intranet server farm, "Data Center Design Overview," follow the same architecture used for intranet server farms yet with different scalability, security, and management requirements. Figure 3 introduces the topology of the Data Center architecture.

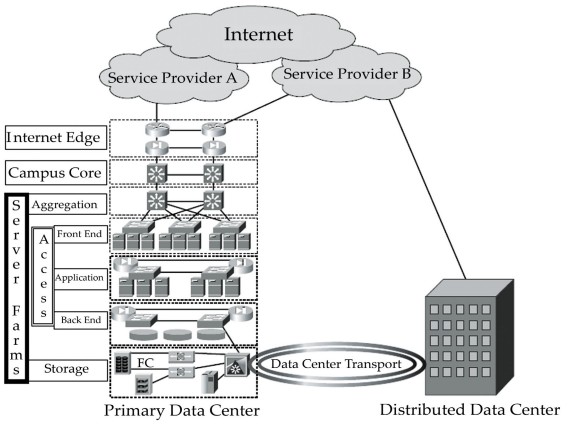


Figure 3: Topology of an Enterprise Data Center Architecture.

Figure 3 shows a fully redundant enterprise Data Center supporting the following areas:

- No single point of failure—redundant components
- Redundant Data Centers

The core connectivity functions supported by data centers are internet edge connectivity, campus connectivity, and server-farm connectivity, as presented in Figure 3.

The Internet Edge provides the connectivity from the enterprise to the Internet and its associated redundancy and security functions, such as the following:

- Redundant connections to different service providers
- External and internal routing through exterior border gateway protocol (EBGP) and interior border gateway protocol (IBGP)
- Edge security to control access from the **Internet**
- Control for access to the Internet from the enterprise clients

The campus core switches provide connectivity between the Internet Edge, the intranet server farms, the campus network, and the private WAN. The core switches physically connect to the devices that provide access to other major network areas, such as the private WAN edge routers, the server-farm aggregation switches, and campus distribution switches.

Layer of Data center Architecture

There are some layer of data center architecture, such as:

- Aggregation layer
- Access layer
  - Front-end segment
  - Application segment
  - Back-end segment
- Storage layer
- Data Center transport layer

Some of these layers depend on the specific implementation of the n-tier model or the requirements for Data Center-to-Data-Center connectivity, which implies that they might not exist in every Data Center implementation. Although some of these layers might be optional in the Data Center architecture, they represent the trend in continuing to build highly available and scalable enterprise Data Centers. This trend specifically applies to the storage and Data Center **transport layers** supporting storage consolidation, backup and archival consolidation, high-speed mirroring or clustering between remote servers farms, and so on.

Aggregation Layer

The aggregation layer is the aggregation point for devices that provide services to all server farms. These devices are multilayer switches, firewalls, load balancers, and other devices that typically support services across all servers. The multilayer switches are referred to as aggregation switches because of the aggregation function they perform. Service devices are shared by all server farms. Specific server farms are likely to span multiple access switches for redundancy, thus making the aggregation switches the logical connection point for service devices, instead of the access switches.

If connected to the front-end Layer 2 switches, these service devices might not offer optimal services by creating less than optimal traffic paths between them and servers connected to different front-end switches. Additionally, if the service devices are off of the aggregation switches, the traffic paths are deterministic and predictable and simpler to manage and maintain. Figure 4 shows the typical devices at the aggregation layer.

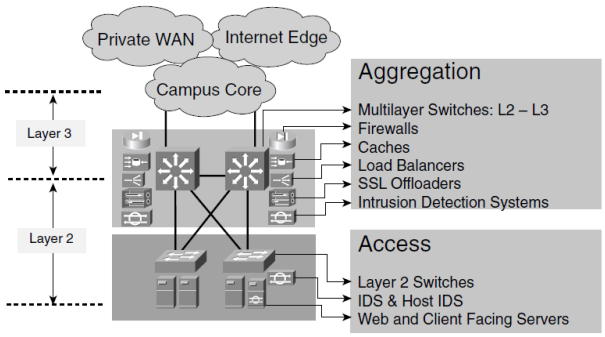


Figure 4: Aggregation and Access Layers

As depicted in Figure 4, the aggregation switches provide basic infrastructure services and connectivity for other service devices. The aggregation layer is analogous to the traditional distribution layer in the campus network in its Layer 3 and Layer 2 functionality.

The aggregation switches support the traditional switching of packets at Layer 3 and Layer 2 in addition to the protocols and features to support Layer 3 and Layer 2 connectivity. A more

in-depth explanation on the specific services provided by the aggregation layer appears in the section, "Data Center Services."

Access Layer

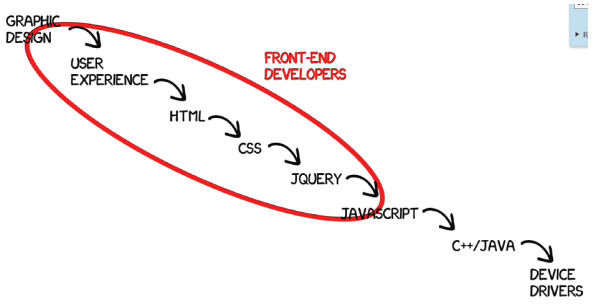
Access layer is responsible for providing end user devices with a connection to network resources. A data access layer (DAL) in computer software, is a layer of a computer program which provides simplified access to data stored in persistent storage of some kind, such as an entity-relational database. This acronym is prevalently used in Microsoft environments.

The access layer provides Layer 2 connectivity and Layer 2 features to the server farm. Because in a multitier server farm, each server function could be located on different access switches on different segments, such as:

Front-End Segment

The front-end segment consists of Layer 2 switches, security devices or features, and the front-end server farms. "Data Center Services" for a detailed description of the features provided by the devices at this layer. The front-end segment is analogous to the traditional access layer of the hierarchical campus network design and provides the same functionality. The access switches are connected to the aggregation switches in the manner depicted in Figure 4. The front-end server farms typically include FTP, Telnet, TN3270 (mainframe terminals), Simple Mail Transfer Protocol (SMTP), web servers, DNS servers, and other business application servers, in addition to network-based application servers such as IP television (IPTV) broadcast servers and IP telephony call managers that are not placed at the aggregation layer because of port density or other design requirements.

The specific network features required in the front-end segment depend on the servers and their functions. For example, if a network supports video streaming over IP, it might require multicast, or if it supports Voice over IP (VoIP), quality of service (QoS) must be enabled. Layer 2 connectivity through VLANs is required between servers and load balancers or firewalls that segregate server farms.



The need for Layer 2 adjacency is the result of Network Address Translation (NAT) and other header rewrite functions performed by load balancers or firewalls on traffic destined to the server farm. The return traffic must be processed by the same device that performed the header rewrite operations.

Layer 2 connectivity is also required between servers that use clustering for high availability or require communicating on the same subnet. This requirement implies that multiple access switches supporting front-end servers can support the same set of VLANs to provide layer adjacency between them.

Security features include Address Resolution Protocol (ARP) inspection, broadcast suppression, private VLANs, and others that are enabled to counteract Layer 2 attacks. Security devices include network-based intrusion detection systems (IDSs) and host-based IDSs to monitor and detect intruders and prevent vulnerabilities from being exploited. In general, the infrastructure components such as the Layer 2 switches provide intelligent network services that enable front-end servers to provide their functions.

The front-end servers are typically taxed in their I/O and CPU capabilities. For I/O, this strain is a direct result of serving content to the end users; for CPU, it is the connection rate and the number of concurrent connections needed to be processed.

Scaling mechanisms for front-end servers typically include adding more servers with identical content and then equally distributing the load they receive using load balancers. Load balancers distribute the load (or load balance) based on Layer 4 or Layer 5 information. Layer 4 is widely used for front-end servers to sustain a high connection rate without necessarily overwhelming the servers.

Application Segment

The application segment has the same network infrastructure components as the front-end segment and the application servers. The features required by the application segment are almost identical to those needed in the front-end segment, albeit with additional security. This segment relies strictly on Layer 2 connectivity, yet the additional security is a direct requirement of how much protection the application servers need because they have direct access to the database systems. Depending on the security policies, this segment uses firewalls between web and application servers, IDSs, and host IDSs. Like the front-end segment, the application segment infrastructure must support intelligent network services as a direct result of the functions provided by the application services.



Application servers run a portion of the software used by business applications and provide the communication logic between the front end and the back end, which is typically referred to as the *middleware* or *business logic*. Application servers translate user requests to commands that the back-end database systems understand. Increasing the security at this segment focuses on controlling the protocols used between the front-end servers and the application servers to avoid trust exploitation and attacks that exploit known application vulnerabilities. Figure 5 introduces the front-end, application, and back-end segments in a logical topology.

The application servers are typically CPU-stressed because they need to support the business logic. Scaling mechanisms for application servers also include load balancers. Load balancers can select the right application server based on Layer 5 information.

Deep packet inspection on load balancers allows the partitioning of application server farms by content. Some server farms could be dedicated to selecting a server farm based on the scripting language (.cgi, .jsp, and so on). This arrangement allows application administrators to control and manage the server behavior more efficiently.

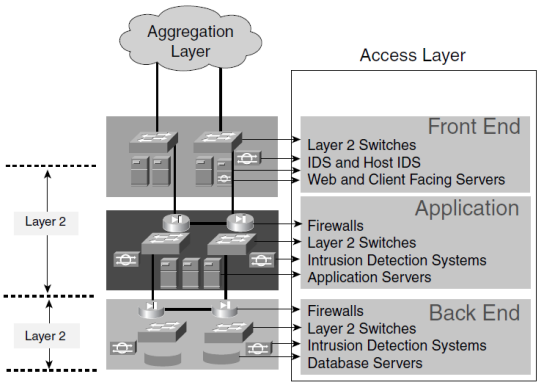


Figure 5: Access Layer Segments

Back-End Segment

The back-end segment is the same as the previous two segments except that it supports the connectivity to database servers. The back-end segment features are almost identical to those at the application segment, yet the security considerations are more stringent and aim at protecting the data, critical or not.

The hardware supporting the database systems ranges from medium-sized servers to high-end servers, some with direct locally attached storage and others using disk arrays attached to a SAN. When the storage is separated, the database server is connected to both the Ethernet switch and the SAN. The connection to the SAN is through a Fiber Channel interface. Figure 6 presents the back-end segment in reference to the storage layer.

The security requirements do not call for physical separation between the different server tiers.

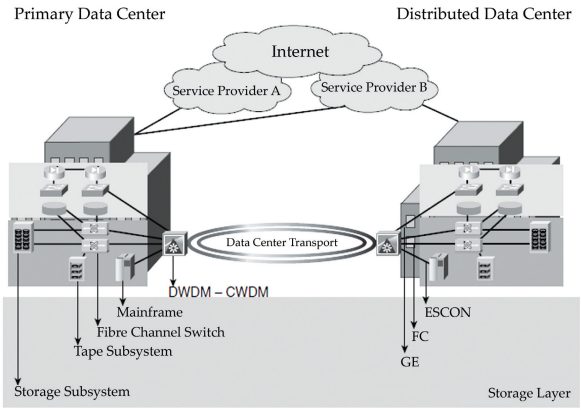
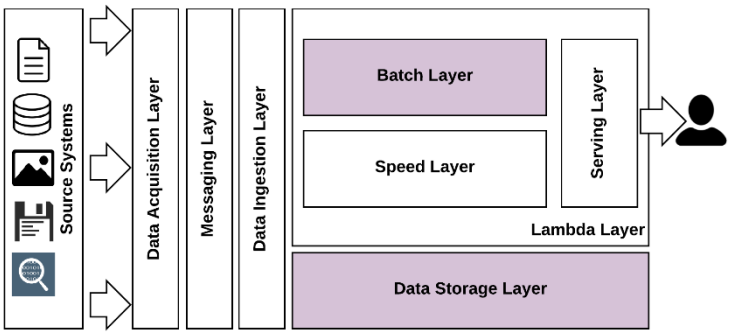


Figure 6: Storage and Transport Layers

Storage Layer

The storage layer consists of the storage infrastructure such as Fiber Channel switches and routers that support small computer system interface (SCSI) over IP (iSCSI) or Fiber Channel over IP (FCIP). Storage network devices provide the connectivity to servers, storage devices such as disk subsystems, and tape subsystems.

The network used by these storage devices is referred to as a SAN. The Data Center is the location where the consolidation of applications, servers, and storage occurs and where the highest concentration of servers is likely, thus where SANs are located. The current trends in server and storage consolidation are the result of the need for increased efficiency in the application environments and for lower costs of operation.



Data Center environments are expected to support high-speed communication between servers and storage and between storage devices. These high-speed environments require block-level access to the information supported by SAN technology. There are also requirements to support file-level access specifically for applications that use Network Attached Storage (NAS) technology. Figure 6 introduces the storage layer and the typical elements of single and distributed Data Center environments.

Figure 6 shows a number of database servers as well as tape and disk arrays connected to the Fiber Channel switches. Servers connected to the Fiber Channel switches are typically critical servers and always dual-homed. Other common alternatives to increase availability include mirroring, replication, and clustering between database systems or storage devices. These alternatives typically require the data to be housed in multiple facilities, thus lowering the likelihood of a site failure preventing normal systems operation. Site failures are recovered by replicas of the data at different sites, thus creating the need for distributed Data Centers and distributed server farms and the obvious transport technologies to enable **communication** between them. The following section discusses Data Center transport alternatives.

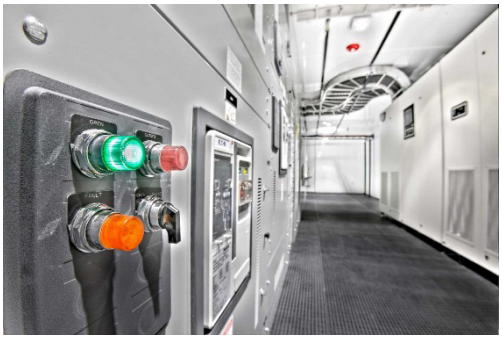
Data center Transport Layer

The Data Center transport layer includes the transport technologies required for the following purposes:

- Communication between distributed Data Centers for rerouting client-to-server traffic
- Communication between distributed server farms located in distributed Data Centers for the purposes of remote mirroring, replication, or clustering

For user-to-server communication, the possible technologies include Frame Relay, ATM, DS channels in the form of T1/E1 circuits, Metro Ethernet, and SONET.

For server-to-server and storage-to-storage communication, the technologies required are dictated by server media types and the transport technology that supports them transparently.



For example, as depicted in Figure 6, storage devices use Fiber Channel and Enterprise Systems Connectivity (ESCON), which should be supported by the metro optical transport infrastructure between the distributed server farms.

If ATM and Gigabit Ethernet (GE) are used between distributed server farms, the metro optical transport could consolidate the use of fiber more efficiently. For example, instead of having dedicated fiber for ESCON, GE, and ATM, the metro optical technology could transport them concurrently.

The likely transport technologies are dark fiber, coarse wavelength division multiplexing (CWDM), and dense wavelength division multiplexing (DWDM), which offer transparent connectivity (Layer 1 transport) between distributed Data Centers for media types such as GE, Fiber Channel, ESCON, and fiber connectivity (FICON). Distributed Data Centers often exist to increase availability and redundancy in application environments. The most common driving factors are disaster recovery and business continuance, which rely on the specific application environments and the capabilities offered by the transport technologies.

- Blade servers
- Grid computing
- Web services
- Service-oriented Data Centers

All these trends influence the Data Center in one way or another. Some short-term trends force design changes, while some long-term trends force a more strategic view of the architecture.

For example, the need to lower operational costs and achieve better computing capacity at a relatively low price leads to the

use of blade servers. Blade servers require a different topology when using Ethernet switches inside the blade chassis, which requires planning on port density, slot density, oversubscription, redundancy, connectivity, rack space, power consumption, heat dissipation, weight, and cabling. Blade servers can also support compute grids. Compute grids might be geographically distributed, which requires a clear understanding of the protocols used by the grid middleware for provisioning and load distribution, as well as the potential interaction between a compute grid and a data grid.

Blade servers can also be used to replace 1RU servers on web-based applications because of scalability reasons or the deployment or tiered applications. This physical separation of tiers and the ever-increased need for security leads to application layer firewalls.

An example of this is the explicit definition for application layer security is (included in the Web Services Architecture [WSA] document). Security on Web Services is in reference to a secure environment for online processes from a security and privacy perspective. The development of the WSA focuses on the identification of threats to security and privacy and the architect features that are needed to respond to those threats. The infrastructure to support such security is expected to be consistently supported by applications that are expected to be distributed on the network. Past experiences suggest that some computationally repeatable tasks would, over time, be offloaded to network devices, and that the additional network intelligence provides a more robust infrastructure to complement Web Services security (consistency- and performance-wise).

Finally, a services-oriented Data Center implies a radical change on how Data Centers are viewed by their users, which invariably requires a radical change in the integration of the likely services. Interoperability and manageability of the service devices become a priority for the Data Center designers.

DATA CENTER SERVICES

Data center services are services that help to either create, implement or maintain a data center, or to enhance what that data center does for an enterprise. This very broad term encompasses

many different kinds of services that may assist planners in brainstorming around how to set up a data center. Other data center services may help to provide better results for what comes out of a data center.

"The first wave of the Internet was really about data transport. And we didn't worry much about how much power we were consuming, how much cooling requirements were needed in the data centers, how big the data center is in terms of real estate. Those were almost afterthoughts."

—Padmasree Warrior

Building a data center requires quite a bit of research about a company's broader IT infrastructure. At the outset, planners need to consider what kinds of data are valuable to the business, what approximate volume of data storage will be needed, and how data from a wide array of sources can flow into a central repository smoothly and be handled in a consistent way. Many of the data center services that experts might classify as services to a data center will help with this kind of precise design and planning. Later on, after a data center has been set up, administrators will often need to adjust or improve specific processes for better performance. Here, data center services can help with optimizing data backup or data recovery, the flow of data to individual employees or customers, and the use of aggregated big data to feed analytics machines that can show businesses more about where they are going in the future.

While many data center services can be primarily software oriented, others are still relatively physical, where comprehensive systems may involve a mix of hardware, software and human resources. Many of these services may apply to the physical structures involved in the data center, such as setting up adequate environmental and security equipment, as well as handling needs for utilities such as power and broadband. Any product that is oriented toward promoting better use of data centers and other similar frameworks could be called a data center service, and within this general category, many different kinds of IT setups and vendor services apply.

The following each service area and its associated features figure 7 introduces the Data Center services.

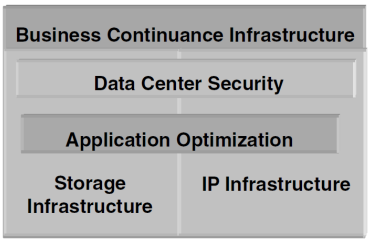


Figure 7: Data Center Services

Services in the Data Center are not only related to one another but are also, in certain cases, dependent on each other. The IP and storage infrastructure services are the pillars of all other services because they provide the fundamental building blocks of any network and thus of any service. After the infrastructure is in place, you can build server farms to support the application environments. These environments could be optimized utilizing network technology, hence the name *application services*. Security is a service expected to leverage security features on the networking devices that support all other services in addition to using specific security technology. Finally, the business continuance infrastructure, as a service aimed at achieving the highest possible redundancy level. The highest redundancy level is possible by using both the services of the primary Data Center and their best practices on building a distributed Data Center environment.

IP Infrastructure Services

Infrastructure services include all core features needed for the Data Center IP infrastructure to function and serve as the foundation, along with the storage infrastructure, of all other Data Center services. The IP infrastructure features are organized as follows:

- Layer 2
- Layer 3
- Intelligent Network Services

Layer 2

Layer 2 features support the Layer 2 adjacency between the server farms and the service devices (VLANs); enable media access; and support a fast convergence, loop-free, predictable,

and scalable Layer 2 domain. Layer 2 domain features ensure the **Spanning Tree Protocol** (STP) convergence time for deterministic topologies is in single-digit seconds and that the failover and failback scenarios are predictable.

Spanning Tree Protocol

Network devices that act as open systems interconnection (OSI) model layer two bridging devices, including switches, are often connected together redundantly to help prevent network failures. The spanning tree protocol (STP) is a set of rules, methods and communications functions that are implemented in bridge software and that prevent data link layer frames from endlessly looping within a redundant bridge topology. The spanning tree protocol is a data link layer protocol that establishes a singular, lowest-cost pathway for data to travel from any one bridged network endpoint to another.

Bridges that use the spanning tree protocol send special messages called bridge protocol data units (BPDUs) to each other in order to share information and notify each other about network changes. Most bridges participate in the spanning tree process, but network administrators might designate some bridges as pass-through-only devices. All of the bridging devices that actively use the spanning tree protocol exchange BPDUs in order to establish and maintain a loop-free, minimum spanning tree topology.



STP initiates recalculation of a new spanning tree topology whenever bridges are added to or removed from the network, when pathways fail within the network, when unidirectional link detection (UDLD) detects a broken fiber or when other topological changes occur and disrupt the established flow of

data. The bridges that actively use the spanning tree protocol begin topological reconfiguration by first agreeing upon and designating one of the bridging devices as a root node for the new spanning tree. They then collaborate by sending BPDUs to each other that contain current information and path costs in order to calculate a new spanning tree that has the lowest cost for the topology. As soon as the bridges agree on a new topology, bridge ports connected to redundant segments that are not a part of the new spanning tree and are not part of a designated ether channel are temporarily blocked to prevent forwarding of data traffic. Blocking these unused redundant pathways creates a loop-free topology.

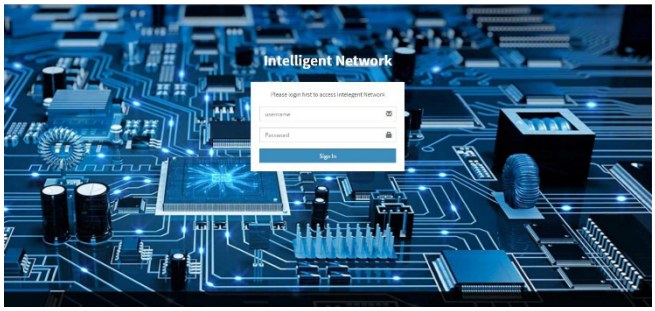
The spanning tree protocol was first developed and implemented by the Digital Equipment Corporation (DEC). It was shortly thereafter incorporated into the Institute of Electrical and Electronics Engineers (IEEE) 802.1D standard. A newer, faster version of the spanning tree protocol called the rapid spanning tree protocol (RSTP) was introduced by the IEEE 802.1w proposal. RSTP has superseded STP as the standard protocol for loop-free bridging and was designated as the protocol of choice in the revised IEEE 802.1D-2004 standard. STP has been refined even further with the release of the multiple spanning tree protocol (MSTP) as detailed in IEEE 802.1Q-2005, which allows separate instances of STP to coexist in multiple virtual local area network (VLAN) configurations.

Layer 3

Layer 3 features enable a fast-convergence routed network, including redundancy for basic Layer 3 services such as default gateway support. The purpose is to maintain a highly available Layer 3 environment in the Data Center where the network operation is predictable under normal and failure conditions. The list of available features includes the support for static routing, Border Gateway Protocol (BGP) and Interior Gateway Protocols (IGPs) such as Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), Intermediate System-to-Intermediate System (IS-IS), gateway redundancy protocols such as the Hot Standby Routing Protocol (HSRP), Multigroup HSRP (MHSRP), and Virtual Router Redundancy Protocol (VRRP) for default gateway support.

Intelligent Network Services

Intelligent network services encompass a number of features that enable application services network-wide. The most common features are QoS and multicast. Yet there are other important intelligent network services such as private VLANs (PVLANs) and policy-based routing (PBR). These features enable applications, such as live or on-demand video streaming and IP telephony, in addition to the classic set of enterprise applications.



QoS in the Data Center is important for two reasons marking at the source of application traffic and the port-based rate-limiting capabilities that enforce proper QoS service class as traffic leaves the server farms. For marked packets, it is expected that the rest of the enterprise network enforces the same QoS policies for an end-to-end QoS service over the entire network.

Multicast in the Data Center enables the capabilities needed to reach multiple users concurrently. Because the Data Center is the source of the application traffic, such as live video streaming over IP, multicast must be supported at the server-farm level (VLANs, where the source of the multicast stream is generated). As with QoS, the rest of the enterprise network must either be multicast-enabled or use tunnels to get the multicast stream to the intended destinations.

Application Services

An application service provider is a large segment within application services that provides software-based services to other companies that access those services over a network. Application services include a number of features that enhance the network awareness of applications and use network intelligence

to optimize application environments. These features are equally available to scale server-farm performance, to perform packet inspection at Layer 4 or Layer 5, and to improve server response time. The server-farm features are organized by the devices that support them. The following is a list of those features:

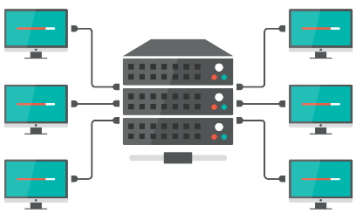
- Load balancing
- Caching
- SSL termination

Load Balancing

Load balancers perform two core functions:

- Scale and distribute the load to server farms
- Track server health to ensure high availability

To perform these functions, load balancers virtualize the services offered by the server farms by front-ending and controlling the incoming requests to those services. The load balancers distribute requests across multiple servers based on Layer 4 or Layer 5 information. The mechanisms for tracking server health include both in-band monitoring and out-of band probing with the intent of not forwarding traffic to servers that are not operational. You also can add new servers, thus scaling the capacity of a server farm, without any disruption to existing services. Layer 5 capabilities on a load balancer allow you to segment server farms by the content they serve.



For example, you can separate a group of servers dedicated to serve streaming video (running multiple video servers) from other groups of servers running scripts and application code. The load balancer can determine that a request for an .mpg file (a video file using MPEG) goes to the first group and that a request for a .cgi file (a script file) goes to the second group.

Caching

Caching involves keeping frequently used data "close" to where it is needed, or preloading data in anticipation of future operations. Data read from disks may be cached until a subsequent write makes it invalid, and data written to disk is usually cached so that many consecutive changes to the same file may be written out in a single operation. In NFS, data caching means not having to send an RPC request over the network to a server: the data is cached on the NFS client and can be read out of local memory instead of from a remote disk. Depending upon the file system structure and usage, some cache schemes may be prohibited for certain operations to guarantee data integrity or consistency with multiple processes reading or writing the same file. Cache policies in NFS ensure that performance is acceptable while also preventing the introduction of state into the client-server relationship.



File attribute Caching

Not all file system operations touch the data in files; many of them either get or set the attributes of the file such as its length, owner, modification time, and node number. Because these attribute-only operations are frequent and do not affect the data in a file, they are prime candidates for using cached data. Think of *ls -l* as a classic example of an attribute-only operation: it gets information about directories and files, but does not look at the contents of the files.

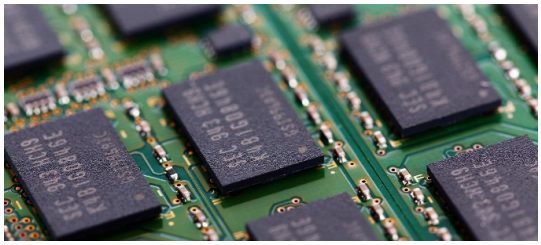
NFS caches file attributes on the client side so that every *getter* operation does not have to go all the way to the NFS server. When a file's attributes are read, they remain valid on

the client for some minimum period of time, typically three seconds. If the file's attributes remain static for some maximum period, normally 60 seconds, they are flushed from the cache. When an application on the NFS client modifies an NFS attribute, the attribute is immediately written back to the server. The only exceptions are implicit changes to the file's size as a result of writing to the file. As we will see in the next section, data written by the application is not immediately written to the server, so neither is the file's size attribute.

Data Caching

The async thread's management of an NFS client's buffer cache. The async threads perform read-ahead and write-behind for the NFS client processes. We also saw how NFS moves data in NFS buffers, rather than in page- or buffer cache-sized chunks. The use of NFS buffers allows NFS operations to utilize some of the sequential disk I/O optimizations of UNIX disk device drivers.

Reading in buffers that are multiples of the local file system block size allows NFS to reduce the cost of getting file blocks from a server. The overhead of performing an RPC call to read just a few bites from a file is significant compared to the cost of reading that data from the server's disk, so it is to the client's and server's advantage to spread the RPC cost over as many data bytes as possible. If an application sequentially reads data from a file in 128-byte buffers, the first read operation brings over a full (8 kilobytes for NFS Version 2, usually more for NFS Version 3) buffer from the file system. If the file is less than the buffer size, the entire file is read from the NFS server. The next *read* ( ) picks up data that is in the buffer (or page) cache, and following reads walk through the entire buffer. When the application reads data that is not cached, another full NFS buffer is read from the server. If there are async threads performing read-ahead on the client, the next buffer may already be present on the NFS client by the time the process needs data from it. Performing reads in NFS buffer-sized operations improves NFS performance significantly by decoupling the client application's system call buffer size and the VFS implementation's buffer size.



Going the other way, small write operations to the same file are buffered until they fill a complete page or buffer. When a full buffer is written, the operating system gives it to an async thread, and async threads try to cluster write buffers together so they can be sent in NFS buffer-sized requests. The eventual *write* RPC call is performed synchronous to the async thread; that is, the async thread does not continue execution (and start another write or read operation) until the RPC call completes. What happens on the server depends on what version of NFS is being used.

- For NFS Version 2, the write RPC operation does not return to the client's *async thread* until the file block has been committed to stable, nonvolatile storage. All write operations are performed synchronously on the server to ensure that no state information is left in volatile storage, where it would be lost if the server crashed.
- For NFS Version 3, the write RPC operation typically is done with the *stable* flag set to off. The server will return as soon as the write is stored in volatile or nonvolatile storage.

There are elements of a write-back cache in the async threads. Queuing small write operations until they can be done in buffer-sized RPC calls leaves the client with data that is not present on a disk, and a client failure before the data is written to the server would leave the server with an old copy of the file. This behavior is similar to that of the UNIX buffer cache or the page cache in memory-mapped systems. If a client is writing to a local file, blocks of the file are cached in memory and are not flushed to disk until the *operating system* schedules them. If the machine crashes between the time the data is updated in a file cache page and the time that page is flushed to **disk**, the file on disk is not changed by the write. This is also expected of systems with local disks -- applications running at the time of the crash may not leave disk files in well-known states.

Server-side Caching

The client-side caching mechanisms file attribute and buffer caching reduce the number of requests that need to be sent to an NFS server. On the server, additional cache policies reduce the time required to service these requests. NFS servers have three caches:

- The inode cache, containing file attributes. Inode entries read from disk are kept in-core for as long as possible. Being able to read and write these attributes in memory, instead of having to go to disk, make the get- and set-attribute NFS requests much faster.
- The directory name lookup cache, or DNLC, containing recently read directory entries. Caching directory entries means that the server does not have to open and re-read directories on every pathname resolution. Directory searching is a fairly expensive operation, since it involves going to disk and searching linearly for a particular name in the directory. The DNLC cache works at the VFS layer, not at the local file system layer, so it caches directory entries for all types of file systems. If you have a CD-ROM drive on your NFS server, and mount it on NFS clients, the DNLC becomes even more important because reading directory entries from the CD-ROM is much slower than reading them from a local hard disk.
- The server's buffer cache, used for data read from files. As mentioned before, file blocks that are written to NFS servers cannot be cached, and must be written to disk before the client's RPC *write* call can complete. However, the server's buffer or page cache acts as an efficient read cache for NFS clients. The effects of this caching are more pronounced in page-mapped systems, since nearly all of the server's memory can be used as a read cache for file blocks.

SSL Termination

A secure socket layer (SSL) connection uses a certificate for authentication before sending encrypted data from a client computer to the web server. SSL termination, a form of SSL offloading, shifts some of this responsibility from the web server to a different machine.

- *Function*: SSL termination works by intercepting encrypted traffic at the server receiving data from the SSL connection. It helps the server by decrypting and verifying the data on a different device so the server does not need to handle the process.
- *Effects:* SSL termination acts as the end of the line, or termination point, for an SSL connection. The process prepares the data for the server and organizes incoming connections in order to allow the web server to concentrate other tasks such as loading web pages.
- *Benefits*: SSL termination allows servers with an SSL connection to handle a large volume of simultaneous connections, or sessions, and cookies. SSL termination also works to increase site and web application performance by increasing server speed. It ensures accuracy by performing decryption on a separate device.

Security Services

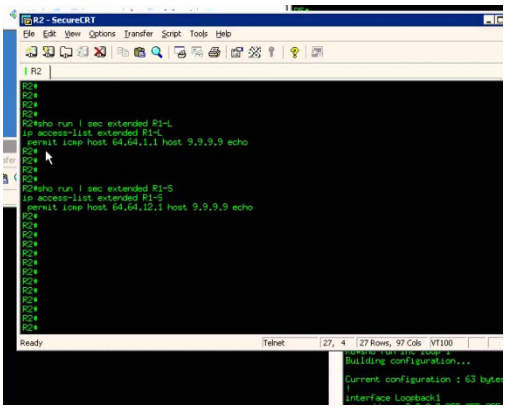
Security services include the features and technologies used to secure the Data Center infrastructure and application environments. Given the variety of likely targets in the Data Center, it is important to use a systems approach to securing the Data Center. This comprehensive approach considers the use of all possible security tools in addition to hardening every network device and using a secure management infrastructure. The security tools and features are as follows:

- Access control lists (ACLs)
- Firewalls
- IDSs and host IDSs
- Secure management

Access control lists (ACLs)

An access control list (ACL) is an attachment to a file, directory, or other object that provides information about permissions associated with the object. If there is no access control list, anyone can interact with the object and do anything with it. If a list is present, however, access and activities are limited to people on the list and the abilities of individual users may be

restricted at different levels. The list can specify users, roles, or groups. Users are individual users who are registered in the system, such as an office network. Roles are titles that are assigned to people. For example, a user might have the role "System Administrator." When an access control list restricts access to certain roles, only people in those roles will be able to manipulate the object. Groups are collections of users who are registered together, such as "Secretarial Pool."



Access control lists can determine who is allowed to view, edit, delete, or move an object. This can be useful on a security level and it can also prevent mistakes. For example, system administrators can limit access to key system files so that people who are not experienced will not accidentally alter, delete, or move those files and cause a problem. Likewise, a file could be made read only except for one user to ensure that if other people on the network access the file, they cannot make changes to it.

Using an access control list for security is part of capability-based security, in which layers of security are provided through the use of tokens that are provided by users of the system. A token provides information about a user's authority and it is matched up with permissions that determine whether or not the user is authorized to perform a given option. This security method allows security at a highly flexible level as individual files and directories can have different permissions.

The access control list is only as good as the security of individual identities on a network. If people do not use passwords or use weak passwords, it is possible to hijack their identities and use them in the system. If a system is penetrated with a keystroke

logger or similar malware, it can also become compromised and make it possible for an unauthorized person to enter the system. This is why security is organized in layers, so that a weakness in one area will not bring down the whole system.

Firewalls

In construction, a firewall is a non-flammable wall that prevents fires from spreading throughout a building. Homes, for example, may have a firewall between the garage and the rest of the house to prevent garage fires from threatening other rooms.

With the rapid popularization of the internet, the term *firewall* is more commonly used in computer networking. Like the firewalls used in homes and buildings, computer firewalls act as a barrier between computers on a network. For companies with a computer network or for individuals with a permanent connection to the internet (such as through DSL or cable), a firewall is critical. Without a firewall, intruders on the network would likely be able to destroy, tamper with or gain access to the files on your computer. Firewalls can come in the form of hardware or software. Without getting into the complex details of how firewalls work, suffice it to say that function with a set of filters that are constantly monitoring traffic on the network. Whenever a packet of information triggers one of the filters, the firewall prevents it from passing through in the attempt to prevent damage. Of course, firewalls sometimes block wanted traffic, and through a continual process of refinement, the filters can be customized to improve their efficacy.



Many computer users who access the internet via a broadband router, may already be benefitting from a firewall. The router itself may be configured to serve as a firewall; any nefarious

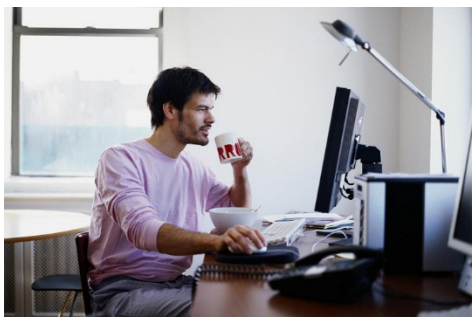
attacks from the network, are halted at the router thereby sparing any ill effects to the computer. Such a hardware firewall can be further bolstered with a secondary line of defense in the form of a software firewall; you can never be too safe when using the internet!

If you would like more detailed information regarding firewalls, how they work or how to configure them, etc. please visit the link provided in the source section below.

IDSs and host IDSs

Information networks can be highly susceptible to malicious attacks from worms, viruses and various other network threats, with regular new issues cropping up on these fronts. Such attacks can paralyze the networks, destroy important data and adversely affect productivity. To prevent this from happening, intrusion detection systems (IDS) are set up to protect information networks.

An intrusion detection system acts as a safeguard that detects attacks before or as they happen, alerts the system administration and then takes appropriate steps to disable the attacks, restoring the network to its normal working capacity. A certain degree of human supervision and investigation is usually required in intrusion detection systems, as the IDS is not completely foolproof. An intrusion detection system may, for instance, fail to identify some network threats or, in cases of busy networks, may not be able to check all the traffic that passes through the network.



In its day to day operation, the intrusion detection system monitors the user activity and traffic on the network, and keeps

watch on the system configurations and the system files. If any abnormalities or attacks are detected, the intrusion detection system immediately sets up an alarm to bring the matter to the attention of the system administrator. The system may then proceed to deal with the network threats, or let the administrator decide on the best way to tackle the problem.

There are three main types of intrusion detection systems that together form an intrusion prevention system. The first is the network intrusion detection, which maintains a library of known network threats. The system checks around the Internet and constantly updates this library; this way the system is kept informed about the latest network threats and is able to better protect the network. The passing traffic is monitored and checked with the library, and if any known attack or any abnormal behavior matches with the ones in the library, the system gears up to block it.

The network node intrusion detection is the second part of the intrusion prevention system. It checks and analyzes the traffic that passes from the network to a specific host.

The third part is the host intrusion detection system, which checks for any changes to the current system; if any files are modified or deleted, the host intrusion detection system sounds the alarm. It may either directly disable the attack or set up a new, improved security environment.

Secure Management

Secure management include the use of SNMP version 3; Secure Shell (SSH); authentication, authorization, and accounting (AAA) services; and an isolated LAN housing the management systems. SNMPv3 and SSH support secure monitoring and access to manage **network devices**.

AAA provides one more layer of security by preventing user's access unless they are authorized and by ensuring controlled user access to the network and network devices with a predefined profile. The transactions of all authorized and authenticated users are logged for accounting purposes, for billing, or for postmortem analysis.

Storage Services

Storage services include the capability of consolidating direct attached disks by using disk arrays that are connected to the network. This setup provides a more effective disk utilization mechanism and allows the centralization of storage management. Two additional services are the capability of consolidating multiple isolated storage area networks (SANs) on to the same larger SAN and the virtualization of storage so that multiple servers concurrently use the same set of disk arrays.

Consolidating isolated SANs on to one SAN requires the use of virtual SAN (VSAN) technology available on the SAN switches. VSANs are equivalent to VLANs yet are supported by SAN switches instead of Ethernet switches. The concurrent use of disk arrays by multiple servers is possible through various network-based mechanisms supported by the SAN switch to build logical paths from servers to storage arrays.

Other storage services include the support for FCIP and iSCSI (Internet Small Computer System Interface) on the same storage network infrastructure. FCIP connects SANs that are geographically distributed, and iSCSI is a lower-cost alternative to Fibre Channel. These services are used both in local SANs and SANs that might be extended beyond a single Data Center.

Business Continuance Infrastructure Services

It is about identifying the dependencies and getting the business to keep operating. This includes storage technology options that enables you to recover frequently used primary data almost instantly, replication capabilities that allow you to replicate data to other locations quickly and automatically, and proven backup and recovery solutions to keep all your data safe and protected.

Business continuance infrastructure services support the highest levels of application availability through the use of networking technology in the three major areas, such as:

- Site selection
- SAN extension
- Data Center interconnectivity

Site Selection

Site selection refers to the features that allow the automatic detection of the failure of a Data Center on the application level and the subsequent reroute of all requests to an available site and server farm. You can use the technology for site selection over the Internet or the intranet. The mechanisms for site selection vary from the use of **domain name system (DNS)** to the host routes and the routed network.

Storage Area Network (SAN) Extension

Storage Area Network (SAN) extension refers to the process of stretching an existing SAN to a secondary location, which could be located in the same Data Center or on a different geographical location. You make this extension to allow the replication of the data from the primary to the secondary SAN. Depending on the distance, the application transaction rate, and the latency between the distributed Data Centers, the replication is synchronous or asynchronous.



Data Center interconnectivity

Data center interconnect refers to the networking of two or more different data centers to achieve business or IT objectives. This interconnectivity between separate data centers enables them to work together, share resources and/or pass workloads between one another.



A data center interconnect is mainly accomplished using networking tools and techniques along with supported software and data center management processes.

Typically, data center interconnect is achieved by connecting data centers through a VPN, leased lines or the Internet. Once connected, these data centers can use all or selected IT resources from within the network.

Some of the ways data center interconnect is used are:

- To connect multiple data centers within the same organization
- To connect partner/customer data centers with shared data and/or resources
- To allow for workload sharing, wherein some or the bulk of the processes are performed at the interconnected data center
- To poool data center resources to meet scalability needs
- To use an interconnected data center as a disaster recovery site

Data Center interconnectivity services are connectivity alternatives provided by various technologies. These connectivity alternatives support the communication requirements for site selection and SAN extension.

CONCLUSION

Data Centers are strategic components of an enterprise that house the critical assets of the business: applications, data, and the computing infrastructure. The Data Center network is vital to sustaining the normal operations of the business. The Data Center network architecture is driven by business requirements.

The criteria that guide the design of a Data Center are availability, scalability, security, performance, and manageability. The Data Center designs described in this book are based on these principles. The distinct services likely offered by the Data Center network include IP infrastructure connectivity, SAN infrastructure connectivity, application optimizations, security, and business continuance.

The IP infrastructure connectivity function refers to routing and switching. The SAN function refers to the Fibre Channel fabric switching. The application optimization functions include load balancing, caching, and SSL offloading. The security function refers to the use of ACLs, firewalls, IDSs, and secure management. The business continuance function refers to the use of site selection (IP based or DNS based), SAN extension, and Data Center Interconnectivity.

CASE STUDY

CALIFORNIA PUBLIC UNIVERSITY OPTS FOR A PREFABRICATED MODULAR DATA CENTER

Data Specialties Inc. (DSI) completed a successful project with a nationally ranked public university in California.

The university, which serves over 20,000 students, needed to demolish their existing data center because the outdated facility was larger than needed given new IT technologies like virtualization and storage and because it was housed in a building slated to be demolished because of seismic risk.

DSI worked with the university's facilities team and a local engineering firm to design a new, state-of-the-art data center to serve campus needs. The university chose DSI's Prefabricated Modular Data Center solution and deployed it on a piece of land otherwise unusable for a new campus building. This new data center included two N+1 UPS systems, 26 server cabinets, air conditioning, hot aisle containment, fire suppression, and a data center monitoring system.

"This new modular data center provides the university with redundancy, reliability, and plenty of room for updated IT technology in a seismically safe location on campus," said Phil Rafferty, Co-Founder and President, DSI. "We are pleased to have worked closely with the university on this project to ensure their students and faculty can be successful."

KEY TERMS

Network
Internet
Communication
Disk
Software
Transport layer
Spanning Tree Protocol
Network devices

DISCUSSION QUESTIONS

1. What do you understand the data center? Explain its facility
2. How data center works? Explain.
3. What is a data center VS server?
4. Describe the data center virtualization.
5. What are the 3 main components of a data center infrastructure?
6. What are the layers of data center?
7. What are the five core elements of the data center infrastructure?
8. Which are the types of data center network architecture?
9. Explain the application architecture models.
10. Discuss about the data center architecture.
11. Write the short not on:
    a. IP Infrastructure Services
    b. Application Services
    c. Security Services

REFERENCES

1. Anderson, C. (2008) 'The end of theory: the data deluge makes the scientific method obsolete', *Wired*, 23 June, http://www.wired.com/science/discoveries/magazine/16-07/pb_theory (last accessed 12 October 2012).
2. Andrejevic, M. (2007) *iSpy: Surveillance and Power in the Interactive Era*. University of Kansas, Lawrence, KS.
3. Barnes, T.J. and Hannah, M. (2001) 'The place of numbers: histories, geographies, and theories of quantification', *Environment and Planning D: Society and Space*, 19: 379–83.
4. Barnett, C. (2013) 'Theory as political technology', *Antipode* (online first), http://radicalantipode.files.wordpress.com/2013/07/book-review_barnett-on-amin-and-thrift.pdf (last accessed 2 August 2013).
5. Bates, J. (2012) '"This is what modern deregulation looks like": co-optation and contestation in the shaping of the UK's Open Government Data Initiative', *The Journal of Community Informatics*, 8(2), http://www.ci-journal.net/index.php/ciej/article/view/845/916 (last accessed 6 February 2013).
6. Beer, D. and Burrows, R. (2007) 'Sociology and, of and in Web 2.0: some initial considerations', *Sociological Research Online*, 12(5), http://www.socresonline.org.uk/12/5/17.html.
7. Beer, D. and Burrows, R. (2013) 'Popular culture, digital archives and the new social life of data', *Theory, Culture and Society*, 30(4): 47–71.
8. Benkler, Y. (2006) *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press, New Haven, CT.
9. Berners-Lee, T. (2009) *Linked Data*, http://www.w3.org/DesignIssues/LinkedData.html(last accessed 30 January 2013).
10. Berry, D. (2011) 'The computational turn: thinking about the digital humanities', *Culture Machine*, 12, http://www.culturemachine.net/index.php/cm/article/view/440/470 (last accessed 3 December 2012).
11. Bertolucci, J. (2013) 'IBM, universities team up to build data scientists', *InformationWeek*, 15 January. http://www.informationweek.com/big-data/big-data-analytics/ibm-universities-team-up-to-build-data-scientists/ (last accessed 16 January 2014).
12. Bettencourt, L.M.A., Lobo, J., Helbing, D., Kuhnert, C. and West, G.B. (2007) 'Growth, innovation, scaling, and the pace of life in cities', *Proceedings of the National Academy of Sciences*, 104(17): 7301–06.

CHAPTER 2

SERVER ARCHITECTURE



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understand basic concept of server architecture
2. Explain TCP and server applications
3. Focus on server availability

INTRODUCTION

A server is a computer program that provides a service to another computer programs (and its user). In a data center, the physical computer that a server program runs in is also frequently referred to as a server. That machine may be a dedicated server or it may be used for other purposes as well.

In the client/server programming model, a server program awaits and fulfills requests from client programs, which may be running in the same or other computers. A given application in a computer may function as a client with requests for services from other programs and also as a server of requests from other programs.

A network for a large business commonly has a client-server architecture, also known as a two-tier architecture. No matter what it is called, this type of architecture is a division of labor for the computing functions required by a large business.



Under the structure of the client-server architecture, a business's computer network will have a server computer, which functions as the "brains" of the organization, and a group of client computers, which are commonly called *workstations*. The server part of this architecture will be a large-capacity computer, perhaps even a mainframe, with a large amount of data and functionality stored on it. The client portions are smaller computers that employees use to perform their computer-based responsibilities.

Servers commonly contain data files and applications that can be accessed across the network, by workstations or employee computers.

An employee who wants to access company-wide data files, for instance, would use his or her client computer to access the data files on the server. Other employees may use a common-access application by accessing the server through their client computers.

This type of server is called an *application server*. It takes full advantage of the client-server architecture by using the server as a storage device for applications and requiring the clients to log in to the server in order to use those applications.

Examples of this kind of application are numerous; among the most popular are word processors, spreadsheets, and graphic design programs. In each case, the use of the applications illustrates the client-server architecture.

The server is not just for storage, however. Many networks have a client-server architecture in which the server acts as a processing power source as well. The client computers are virtually "plugged in" to the server and gain their processing power from it. A client computer can simulate the greater processing power of a server without having the requisite processor stored within its framework. Here, the client-server architecture describes a virtual sort of power plant.

BASIC CONCEPT OF SERVER ARCHITECTURE

Client-server architecture (client/server) is a network architecture in which each computer or process on the network is either a client or a server.

Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

"Today, Web services is really about developing for the server. What it means to developers is any set of systems services that you make a Web service you to access by any kind of device with a highly interactive client, not just a browser."

—John Fowler

Types of Servers

Servers are often categorized in terms of their purpose. A Web server, for example, is a computer program that serves requested HTML pages or files. The program that is requesting web content is called a client. For example, aWeb browser is a client that requests HTML files from Web servers.

Here are a few other types of servers, among a great number of other possibilities:

- An application server is a program in a computer in a distributed network that provides the business logic for an application program.
- A proxy server is software that acts as an intermediary between an endpoint device, such as a computer, and another server from which a user or client is requesting a service.
- A mail server is an application that receives incoming e-mail from local users (people within the same domain) and remote senders and forwards outgoing e-mail for delivery.
- A virtual server is a program running on a shared server that is configured in such a way that it seems to each user that they have complete control of a server.
- A blade server is a server chassis housing multiple thin, modular electronic circuit boards, known as server blades. Each blade is a server in its own right, often dedicated to a single application.
- A file server is a computer responsible for the central storage and management of data files so that other computers on the same network can access them.
- A policy server is a security component of a policy-based network that provides authorization services and facilitates tracking and control of files.



Choosing the Right Server

There are many factors to consider in the midst of a server selection, including virtual machine (VM) and container consolidation. When choosing a server, it is important to evaluate the importance of certain features based on the use cases. Security capabilities are also important and there will probably be a number of protection, detection and recovery features to consider, including native data encryption to protect data in flight and data at rest, as well as persistent event logging to provide an indelible record of all activity. If the server will rely on internal storage, the choice of disk types and capacity is also important because it can have a significant influence on input/output (I/O) and resilience.

Many organizations are shrinking the number of physical servers in their data centers as virtualization allows fewer servers to host more workloads. The advent of cloud computing has also changed the number of servers an organization needs to host on premises. Packing more capability into fewer boxes can reduce overall capital expenses, data center floor space and power and cooling demands. Hosting more workloads on fewer boxes, however, can also pose an increased risk to the business because more workloads will be affected if the server fails or needs to be offline for routine maintenance.

A server maintenance checklist should cover physical elements as well as the system's critical configuration.

Internet Data Center Server

While people often think of the Internet as an immaterial object, the data that makes up the Internet is actually stored at a very physical location: data servers. Businesses sometimes operate

their own servers or rent out servers from data centers, enabling them to store online files -- like those that make up a website -- and make them globally accessible. Essentially, an Internet data center server is the saving location for files accessible from the Internet.

Data Server

A data server is basically a computer without peripherals like monitors and keyboards. A server operates entirely as a saving location, and is connected to a network to make that data accessible to computers. In the case of an Internet server, the device is connected to the Web, so that any computer with a Web connection can access the files stored on the server. Server's store and process data just like a computer, and are connected to the Internet through wired or wireless connections.

Data Center

A data center is made up of numerous data servers housed in the same complex. Like computers, servers require certain environmental conditions, such as moderate levels of heat and humidity, as well as network connections to access the Internet. Large data centers with huge amounts of hard disk storage allow the companies that operate them to capitalize on economies of scale, saving more data for similar costs onsite maintenance and connectivity infrastructure. Many large Internet companies operate their own dedicated data centers, while some data centers offer storage services for multiple clients.

How It Works

Just like two computers connected in a local network, Internet servers send information to Web browsers through network connections. Data stored on a server is divided into packets for transmission, and is sent through special computers called routers that determine the best path for that data to travel -- through a series of wired and wireless networks -- to reach an Internet service provider and ultimately a computer. Every time you type a Web address into a browser, you are requesting information from a server, and the process is simply reversed when you want to upload information to an Internet server.

Data and Security

Like any type of computer data, server data is vulnerable to corruption, transmission errors, loss and hacking. While security protocols vary across companies and data centers, most data centers implement comprehensive security measures that include data encryption during transmission, multiple saving locations in case one server becomes disabled, simultaneous saving at two or more locations so that files can be reconciled and backed up in case they are not properly saved at one, and complete wipes of disk **data** when a server has exhausted its useful life.

Network Attachment

Servers and clients attach to the network via network interface cards (NIC). This covers relevant aspects of using NICs when designing a data center network.

Servers can also be attached to a storage-area network by means of host bus adapters (HBAs) for Fiber Channel connectivity.

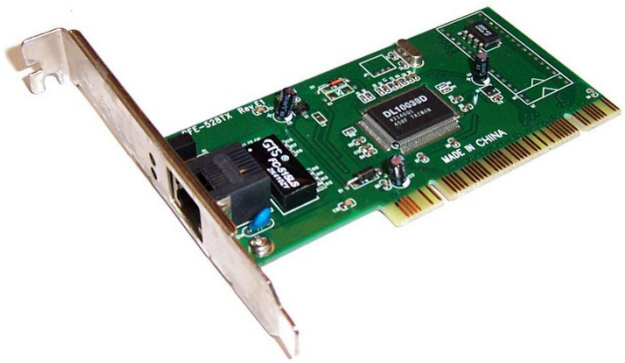
Network attached storage (NAS) is a dedicated server, also referred to as an appliance, used for file storage and sharing. NAS is a hard drive attached to a network, used for storage and accessed through an assigned network address. It acts as a server for file sharing but does not allow other services (like emails or authentication). It allows the addition of more storage space to available networks even when the system is shutdown during maintenance.

NAS is a complete system designed for heavy network systems, which may be processing millions of transactions per minute. NAS provides a widely supported storage system for any organization requiring a reliable network system.

Network Interface Cards (NIC)

Network interface cards (NICs), also known as network adapters or adapters for short, provide connectivity between hosts (client and servers) and LANs. NICs translate the computer data into signals to be transmitted on the network, and vice versa. It is a device that allows computers to be joined together in a network, typically a Local Area Network (LAN). Networked computers

communicate with each other using a particular protocol or agreed-upon language for transmitting data packets between the different machines or "nodes." The network interface card acts as an interpreter, allowing the machine to both send and receive data on a LAN. Information Technology (IT) specialists often use these cards to setup wired or wireless networks.



Function and Purpose of an NIC

One of the most common languages or protocols used with a LAN is Ethernet. There are also other, lesser-used protocols such as Token Ring. When building a LAN, a network interface card is installed in each computer on the network and each one must use the same architecture. For example, all the cards must be Ethernet cards, Token Ring cards, or an alternate technology.

An Ethernet network interface card is installed in an available slot inside the computer, typically on the motherboard. The NIC assigns a unique Media Access Control (MAC) address to the machine, which is used to direct traffic between the computers on a network. Network cards also change data from a parallel format, used by computers, to a serial format necessary in data transfers; and then back again for received information.

Wired Networks

A card's back plate features a port that fits a data cable, such as an Ethernet cable, which runs from each NIC to a central hub or switch. The hub acts like a relay, passing information between computers using their MAC addresses and allowing

them to share resources like printers and scanners. In a wired network, a cable physically connects each computer to each other or to a hub.

Wireless Networks

A network interface card does not have to be hard-wired with physical cable. Wireless cards are installed like their wired counterparts, but rather than a port for a cable, the card features a small antenna. The NIC communicates with a central wireless switch or hub via radio waves. Wireless LANs are often convenient, but may have some restrictions depending on the material a building is made from. For example, lead in walls can block wireless signals between the network interface card and a hub or switch.

Choosing the Right NIC

When buying components for a LAN, it is important to make sure the NICs and hub or switch have the same capabilities. The entire network should be either wired or wireless, unless components are specifically chosen that have both functionalities. In addition, newer versions of hardware often support more features and greater data speeds than older equipment. It is important to make sure a central switch or hub is just as good as the individual cards used in a network.

Wide Area Networks and NICs

Computer users can also connect LANs located in different areas of a city, region, or country, through Asynchronous Transfer Mode (ATM) and the creation of a Wide Area Network (WAN). LANs are built with a network interface card in each computer, but ATM uses Internet connections to link multiple LANs to an online switch, making each one part of a **network**. This type of WAN is referred to as an "Internetwork," as the larger WAN consists of individual nodes that are each a smaller LAN.

Features of Server Adapters

Adapters have buffers whose size varies according to the model and the vendor; Gigabit adapters typically have 64-KB onboard

buffers. Incoming traffic from the network is stored in the buffer and eventually passed to the memory. When the data is ready to be transferred, the adapter triggers a hardware interrupt to invoke the server's CPU. NICs typically offer several options to reduce the number of interrupts to the system processors and the involvement of the CPU in moving the data.

Most server adapters offer the following features:

- ***Direct Memory Access (DMA)****:* This facilitates the transfer of data from the NIC to the computer memory without involving the system processor.
- ***Interrupt coalescing***: This reduces the number of interrupts sent by the NIC to the CPU by delaying the notification of a newly arrived packet. This feature is especially useful with Gigabit Ethernet NICs because the number of interrupts per second at wire rate can take 50 percent or more of the server CPU.
- ***Jumbo frames support****:* This supports the use of bigger Ethernet frames (typically 9-KB frames). Using jumbo frames also alleviates the interrupts to the CPU and allows higher throughput.
- ***IEEE 802.1Q support****:* This supports the connection of a server to several VLANs.
- ***TCP/IP offloading***: Some NICs can offload part of the TCP processing from the OS TCP stack. NICs supporting TCP offloading typically offer TCP checksum calculation and segmentation functions that can significantly reduce the CPU utilization.
- ***Fault tolerance and load balancing****:* These features facilitate attaching a server to the network with multiple ports, to increase availability and performance.
- ***IEEE 802.3ad support****:* 802.3ad is a protocol that allows negotiation between ports of separate devices to form a channel.

Server Multihoming

It is possible and likely for a single server to use multiple NICs for both redundancy and performance. The connectivity between the server and the network device (switch) can be composed of several ports out of a single NIC or of several ports, one per

NIC. The key point is that a single server connects to one or multiple switches with multiple ports.

Fault tolerance

One port (or NIC) is active (receives and transmits traffic) while the other ports are in standby (these ports do not forward or receive traffic). When the active port fails, one of the ports that was previously in standby takes over with the same MAC address. This option is depicted as option in figure 1 (a).

Load balancing

Only one port receives traffic; all the ports transmit traffic. If the receive port fails, a new port is elected for the receive function. In figure 1, this is option **b**.

Link Aggregation

A number of ports (NICs) form a bundle that logically looks like a single link with a bandwidth equivalent to the sum of the bandwidth of each single link. In Figure 1, this is option **c**.

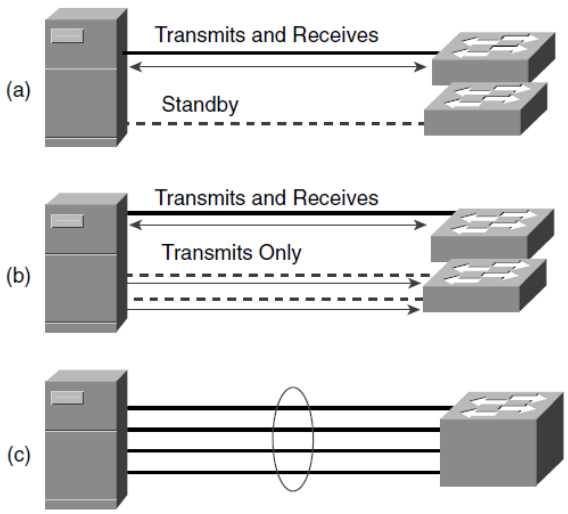


Figure 1: Server Attachment with Multiple Ports: (a) Fault Tolerance, (b) Load Balancing, and (c) Link Aggregation

PCI and PCI-X Buses

NICs provide the interface between the host's CPU and the network, and translate parallel data from the bus into a serial transmission on the wire. The bus allows the communication between the CPU/memory and the NIC.

The predominant bus architecture today is Peripheral Component Interface (PCI), which Intel originally developed. PCI is typically deployed as a 32-bit bus at 33 MHz, which allows a maximum theoretical bandwidth of 133 MBps (1.064 Gbps). Using a gigabit NIC in conjunction with a 32-bit PCI bus could overwhelm the bus by saturating the available bandwidth.

The availability of 64-bit PCI buses with 66 MHz, which provides for a peak bandwidth of 533 MBps (4.25 Gbps). This bandwidth can still be a bottleneck when deploying multiple NICs for server multihoming because 4-Gbps NICs could again saturate the bus.

PCI-X

Peripheral component interconnect-extended (PCI-X) is a computer architecture standard used for 32-bit PCI bus expansion slots. These slots are used to enhance the existing capabilities of a computer motherboard.

PCI-X is a computer bus technology (the "data pipes" between parts of a computer) that increases the speed that data can move within a computer from 66 MHz to 133 MHz. PCI-X doubles the speed and amount of data exchanged between the computer processor and peripherals. With the current PCI design, one 64-bit bus runs at 66 MHz and additional buses move 32 bits at 66 MHz or 64 bits at 33 MHz. The maximum amount of data exchanged between the processor and peripherals using the current PCI design is 532 MB per second. With PCI-X, one 64-bit bus runs at 133 MHz with the rest running at 66 MHz, allowing for a data exchange of 1.06 GB per second. PCI-X is backwards-compatible, meaning that you can, for example, install a PCI-X card in a standard PCI slot but expect a decrease in speed to 33 MHz. You can also use both PCI and PCI-X cards on the same bus but the bus speed will run at the speed of the slowest card. PCI-X is more fault tolerant than PCI.

For example, PCI-X is able to reinitialize a faulty card or take it offline before computer failure occurs. IBM, HP, and Compaq designed PCI-X for servers to increase performance for high bandwidth devices such as Gigabit Ethernetcards, Fibre Channel, Ultra3 Small Computer System Interface, and processors that are interconnected as a cluster. Compaq, IBM, and HP submitted PCI-X to the PCI Special Interest Group (Special Interest Group of the Association for Computing Machinery) in 1998. PCI SIG approved PCI-X, and it is now an open standard that can be adapted and used by all computer developers. PCI SIG controls technical support, training and compliance testing for PCI-X. IBM, Intel, Microelectronics and Mylex plan to develop chipsets to support PCI-X. 3Com and Adaptec intend to develop PCI-X peripherals.

Client and Server Packet Processing

Packet processing refers to the wide variety of algorithms that are applied to a packet of data or information as it moves through the various network elements of a communications network. There are two broad classes of packet processing algorithms that align with the standardized network subdivision of control plane and data plane. The algorithms are applied to either:

- Control information contained in a packet and are used to transfer the packet safely and efficiently from origin to destination
- The data content (frequently called the payload) of the packet is used to provide some content-specific transformation or take a content-driven action.

User Mode and Kernel Mode

The kernel of an OS provides the core services, including the management of threads and processes, interrupt handling, and the management of input/output (I/O) operations. UNIX based operating systems have a kernel that handles all possible services, from process management to network operations. Other OSs provide a microkernel that handles operations such as thread scheduling, memory management, and interrupt handling. When using OS with microkernels, support for I/O operations or process management is offered by separate services.

The system services (I/O handling, memory management, thread scheduling, and so on) execute in kernel mode. Applications execute in user mode. Code executing in kernel mode has access to the system memory and direct access to the hardware.

Applications that need to perform I/O operations use specific interfaces (in UNIX, these are called system calls) that eventually cause the process to change and run in kernel mode.

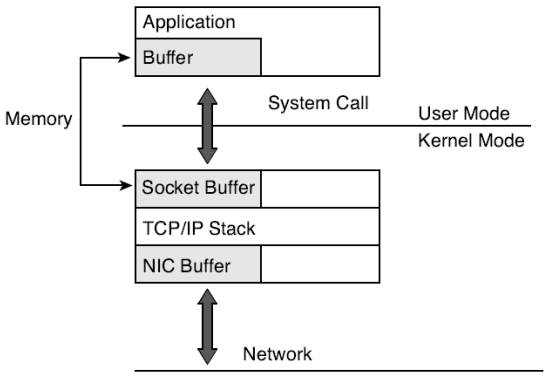


Figure 2: User Mode and Kernel Mode for I/O Processing

When an application must send traffic on the network, it invokes a system call from the process where it runs, which causes the invocation of procedures from the socket layer. This also causes the change of execution mode in the process from user mode to kernel mode. The process data is copied into the memory buffers used by the kernel. Figure 2 shows the switch from user to kernel mode; notice that data is copied between memory buffers.

Input processing occurs when a new packet is received by the NIC and the interrupt invokes the Ethernet driver (kernel mode). Eventually, the kernel wakes up the process waiting for incoming traffic. The data is copied from the socket buffer to the process.

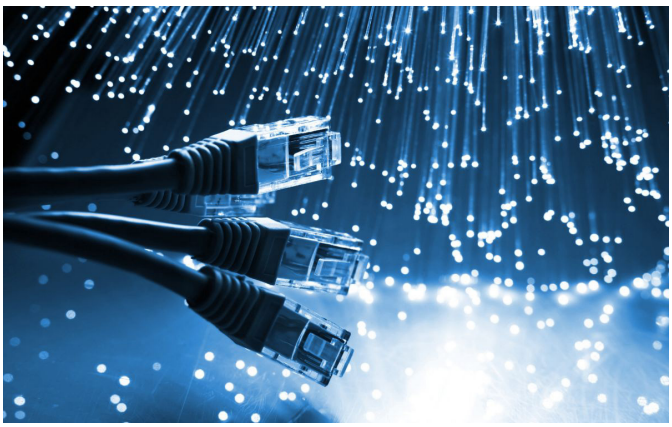
As just changing between user mode and kernel mode requires copying between the application buffer and the socket buffer for the data structures holding the data for **transmission** on the network or the data received from the network.

Ethernet Driver

Ethernet drivers are software programs that provide hardware-software interaction between the operating system of a personal computer (PC) and its local area network (LAN) port. Installing compatible drivers will enable network access for the user. This enables a user to communicate with other PCs, connect to the Internet, or access a common printer or scanner.

A driver is a program that facilitates interaction between a computer's hardware and operating system (OS). It is necessary for the accurate functioning of a computer. Drivers enable end users to access the hardware devices connected to their computer. Drivers are hardware and OS specific. The hardware manufacturer usually provides a CD or DVD containing the setup file of the relevant driver package.

Ethernet drivers determine the rate at which data can be sent and received between a computer and a network. These drivers, however, differ from the PC's motherboard LAN configuration and the operating system in use. Advancement in computer bus communication technology allows new drivers to communicate faster and more efficiently than outdated ones. It is therefore important to use the latest drivers to complement high-speed LAN and Internet connections.



To install an Ethernet driver, a driver scan can be performed to give a summary of all drivers that need to be updated. This is automatically performed by some operating systems. The appropriate drivers are either automatically installed by the OS or can be manually installed from a CD/DVD, hard

disk, or removable media, such as a universal serial bus (USB) flash device. Alternately, they can be downloaded from the manufacturer's website, which might be a cumbersome task when seeking Ethernet drivers because, without one, there may be difficulty accessing the Internet. The drivers must then be installed by following the instructions of the setup wizard.

Regular driver updates are usually made available online by software providers via their website. Paperback and online user manuals published by vendors provide information on how a user can extract maximum value from her or his hardware. This includes a step-by-step guide on correct installation and compatibility checking.

Most operating systems have an automatic network error troubleshooting protocol, which attempts to repair the drivers in case of malfunction. Accordingly, driver packages are released for specific types of hardware. There is a driver corresponding to each external device. Examples include USB drivers for USB devices and CD/DVD drivers for internal or external magnetic drives. Ethernet drivers manage the wired network connections of a PC, while Wi-Fi drivers manage wireless connections.

Ethernet Controller Drivers

Ethernet controller drivers sit between a computer's network system and operating system, facilitating communication between the two parts of the computer. Overall, while network card specific Ethernet controller drivers do exist, they are often of a lower priority than other systems. Most modern operating systems contain default drivers that provide full network access without any additional software. If specific Ethernet controller drivers are required, they may be found on the website of the computer's assembler or the network card's manufacturer.

Drivers allow software and hardware to interact more efficiently and help provide a feature-rich environment. Many drivers contain coding that speeds up information transferal and power usage. These features let the computer run faster while interacting with the hardware and reduce the power consumed by the peripheral.

Hardware drivers will also unlock features on a piece of hardware that the software system does not know how to use.

Hardware that is more complex is most likely to have these inaccessible features. For example, every personal computer has a video system; it allows the computer to display information on a monitor. The operating system understands how to use the video card well enough to display information, but very little else. By installing the driver that goes along with the specific card, the computer will learn how to use additional features.



With that being said, network controller cards are typically very simple. Most operating systems do not require additional Ethernet controller drivers beyond those that come with the system. These are called generic drivers; they work on a wide range of card types and styles, often without any interaction from the user. As a general rule, these drivers rarely need an update or change. Often, the only time the drivers are even considered is when a computer is acting up and the network system seams at fault.

There are three common places to find card specific Ethernet controller drivers. The card may have come with a compact disk that had a set of drivers on it. These drivers are likely several versions out of date and should only be used as a means to find updated drivers online. Many cards no longer come with disks at all; the manufacturers assume users will use generic drivers to go online without installation.

The other two places are both online. If the network card came inside a pre-built computer from a major manufacturer, it

is likely that the most up-to-date drivers are on the assembler's website. These drivers are often tailor-built to match not only the card, but also the assembler's overall computer design. In addition, the card's manufacturer will have drivers. In order to find the right drivers, a user needs the make and model of the network card.

Server TCP/IP Processing

TCP/IP for IP version 4 (IPv4) is a networking protocol suite that Microsoft Windows uses to communicate over the internet with other computers. It interacts with Windows naming services like DNS and security technologies, such as IPsec primarily, as these help facilitate the successful and secure transfer of IP packets between machines. Ideally, TCP/IP is used whenever Windows-based computers communicate over networks. A server application based on TCP waits for incoming connections on specific Layer 4 ports. A client requesting an application sends a TCP SYN to one of these ports and performs a TCP handshake with the server. Which ports the server is listening to by using the command netstat –a. The previous command also displays information about the send and receive queue.

For example, among others, the following servers: an Apache HTTP server configured for SSL, a DNS server, a Telnet server, and an FTP server.

```
[root@localhost admin]# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.20.5.14:telnet       10.21.4.5:1073          ESTABLISHED
tcp        0      0 10.20.5.14:domain       *:*                     LISTEN
tcp        0      0 mp5:domain              *:*                     LISTEN
tcp        0      0 *:www                   *:*                     LISTEN
tcp        0      0 *:https                 *:*                     LISTEN
tcp        0      0 *:587                   *:*                     LISTEN
tcp        0      0 *:smtp                  *:*                     LISTEN
tcp        0      0 *:printer               *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 *:login                 *:*                     LISTEN
tcp        0      0 *:shell                 *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
tcp        0      0 *:ftp                   *:*                     LISTEN
tcp        0      0 *:finger                *:*                     LISTEN
tcp        0      0 *:auth                  *:*                     LISTEN
tcp        0      0 *:1024                  *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
udp        0      0 *:2154                  *:*
udp        0      0 10.20.5.14:domain       *:*
```

The typical connection status that you can observe is either LISTEN (before the connection is established), TIME_WAIT (right after the connection has been closed), or ESTABLISHED if the connection is active.

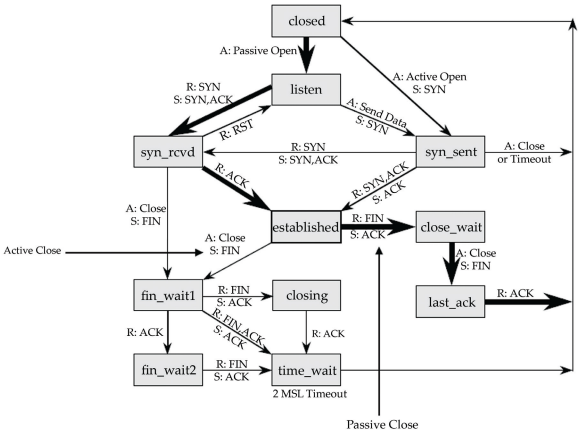


Figure 3: TCP State Machine with Highlighted Sequence of States Taken by a Server

The bold arrows show the typical states taken by a connection on a server: At the reception of a SYN, a server responds with a SYN ACK, which brings the state to SYN-RCVD (passive open). After receiving one ACK, the connection is in the ESTABLISHED state. From the ESTABLISHED state, the typical server connection closure is initiated by the client (passive close).

Connections often can be stuck in a state, and this is the symptom of a problem:

SYN-RCVD

The server has received a SYN (passive open), and it has sent a SYN ACK but has not received an ACK back from the client. Among the possible causes is an asymmetric traffic path. The traffic from the client hits a device that performs NAT on the destination IP to the real server IP address. The traffic from the server instead could bypass the device that translates the **IP address**. As a result, the client does not recognize the source of the SYN ACK.

SYN-SENT

The server has sent a SYN (active open) but has not received a SYN ACK from the client. Among the possible causes, the routing could be misconfigured.

FIN-WAIT1

The server has sent a FIN but has not received an ACK or a FIN ACK. Among the possible causes for this problem, the client could be down as the result of either an administrative action (power cycling) or a failure.

The output of the netstat –a command can help you detect problems that relate either to a network misconfiguration (routing issues) or to an anomaly on the client.

The closure of the TCP connection can take the form of a FIN handshake or a TCP RST. When either side of a connection sends a TCP FIN, the other entity can finish sending the content of its send buffer and eventually close the connection.

When either the client or the server sends a TCP RST, it means that the connection is closing, and the data left in the send buffer of either entity is discarded. Under normal conditions, either side of an application should close connections with the TCP FIN sequence, but there are exceptions; for example, some browsers use TCP RSTs. For some applications, receiving a TCP RST is the symptom of a failure on the client side.

Sockets

Sockets provide application processes with a protocol-independent interface to the network communication. In UNIX, the communication between the process and the OS (kernel) uses system calls.

The following list shows the sequence of system calls used by a server that uses TCP as the transport protocol:

- **Socket()**: Creates a socket for streams (TCP). The socket descriptor created on the server defines the capability of the server to accept connections of a specific type, for the IP address defined by the Bind() call.
- **Bind()**: Associates the socket with a local IP address and Layer 4 port number.
- **Listen()**: Defines the length of the receive queue for incoming connection requests.
- **Accept()**: Used by the server to wait for incoming connection requests. This call puts the process in a

waiting state for incoming connections. When a new connection request is received, the accept call returns a new socket descriptor. This descriptor references a specific client connection to the server.

- **Read()/Write()**: Exchange data between the server and the client.

Client applications based on TCP use the socket() and connect() calls to establish the connection to the server, and write() and read() calls to exchange data.

Server applications based on UDP use different system calls than those listed because UDP does not establish connections between the client and the server. As a result, there is no need to use the Listen() and Accept() system calls on a UDP server. The following list shows the sequence of system calls used by a server using UDP as the transport protocol:

- **Socket()**: This call creates a socket descriptor of the datagram type.
- **Bind()**: This call binds the socket descriptor to an IP address and a port.
- **Recvfrom()**: A process calling recvfrom() waits for incoming connections. When UDP datagrams arrive, recvfrom() returns the IP address of the client together with the datagram.
- **Sendto()**: This call sends datagrams to the IP address specified in the parameters to this call.

Sockets allow options to be defined to control the TCP and UDP functions in the kernel, such as these:

- **SO_KEEPALIVE**: This option controls the capability of TCP to verify whether an idle connection is still alive by sending a keep alive probe after a configurable amount of time (the default is 2 hours). If no ACK is received, the socket is closed.
- **SO_RCVBUF**: TCP or UDP receives buffer size. This buffer size is used as the receive window by the TCP protocol during the handshake.
- **SO_SNDBUF**: TCP or UDP send buffer size.

Server applications that enable the SO_KEEPALIVE option should configure an appropriate TCP KEEPALIVE. Under normal

conditions, for many applications a TCP connection is never idle for 2 hours. The implication of using the default setting is the possibility of keeping servers' resources allocated for clients that did not close connections.

The receive buffer size has a direct implication on the performance of TCP transfers. Suppose that the two communicating entities are host A and host B, and suppose that the receive buffer configured on host B is 64 KB.

Assuming that the maximum segment size (MSS) is 1460 bytes, host A can send 44 segments (64 KB/1460 bytes, rounded to an even number of MSS) without receiving any acknowledgements from host B.

When you compare this with a transfer in which host A can send only one segment after receiving an ACK from host B, the benefit in terms of throughput is evident.

Typical TCP window sizes can range from 8 to 64 KB, or more if the OS supports the Window Scale option (defined in RFC 1323).

TCP AND SERVER APPLICATIONS

Applications based on TCP take advantage of the features offered by TCP, such as reliability, flow control, and congestion avoidance. Examples of TCP applications and to identify some of its reliability, flow control, and congestion-avoidance features.

- ***Interactive traffic***: With Telnet. Telnet is an interactive application that generates a limited amount of traffic on a network. This means that, most of the time, the transmit buffer is empty or filled with small segments.
- ***Bulk transfer***: With a Hypertext Transfer Protocol (HTTP) transfer of a big object. The transmit buffer is continuously filled, and most of the segments sent over the network are as big as the maximum segment size (typically 1460 bytes).

"The X server has to be the biggest program I've ever seen that doesn't do anything for you."

—Ken Thompson

Segments, Datagrams, and Frames

TCP data is a stream of bytes. The application data goes into the TCP send buffer; in the case of Telnet, the application data could be a sequence of keystrokes. TCP is unaware of keystrokes; for TCP, the send buffer contains a sequence of bytes. For the duration of the TCP connection, the TCP stack takes data from the send buffer and slices the stream of bytes into chunks (called segments) to send on the network. In figure 4 can help you understand how application data is turned into frames to transmit on the network. At the top of the figure, you can see the application generating records of different sizes.

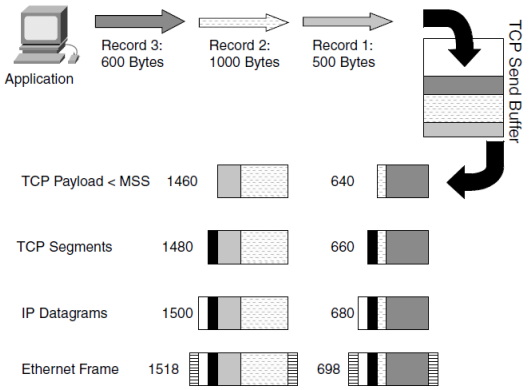


Figure 4: TCP Processing of Application Data

These records end in the TCP send buffer. From the TCP point of view, all these records together are a stream of bytes to be sent on the network. TCP takes this data out of the send buffer to form segments. A TCP header with a TCP payload is called a segment. When forming segments, TCP tries to fill the maximum segment size (MSS) which, on an Ethernet LAN, is typically 1460 bytes. The MSS specifies how big the payload of a TCP segment can be for it to be compatible with the maximum transmission unit (MTU) of the physical network between the sender and the receiver.

The application writes three records in the TCP send buffer. TCP forms the first segment with two records the first record and a portion of the second. The second segment is formed by the remaining portion of the second record and by the third record.

The size of the first segment is 1480, which equals the MSS (1460 bytes) plus the TCP header (20 bytes). This segment plus the IP header form a datagram. Its size grows to 1500 bytes because the IP header is 20 bytes. The IP datagram plus the Ethernet header and the trailer become an Ethernet frame; its size is 1518 **bytes** (the MTU size).

TCP Interactive Traffic Profile

TCP Interactive is operationally state equivalent to the conventional TCP except applications can optionally subscribe, receive, and in real-time react to selected local end-point events. Consequently, it itself does not modify any TCP or traffic dynamics. It is also backward compatible.

Transient ware operates within the full access and authentication framework of the host operating system and it does not pose any special security risk.

Since, the solutions are coded in transient ware(s) - outside of network layers- the solutions can be arbitrarily sophisticated yet remarkably easy to implement, deploy, discard and improve. Solutions can be real-time because of their event based instantiations.

The Telnet application mainly transmits keystrokes from a client computer to the server and sends back to the client the characters to be displayed.

The exception of the connection-establishment segments, tcpdump shows the sequence number as the offset from the initial values instead of their absolute value. Also notice that:

The notation is *first:last(nbytes)*, which means "sequence numbers first up to but not including last," which is *nbytes* bytes of user data.

Connection Setup

A connection is a term that describes the link between a plug and connector into a port or jack. The client 10.21.4.5 opens a TCP connection to port 23 on 10.20.5.14 from a random source port (1159). The TCP handshake is the sequence of SYN (see the S in the trace) followed by a SYN ACK and by the ACK.

Each segment carries a sequence number that tells the receiver how to compose the byte stream with all the received segments. For example, the first byte of the TCP payload in the last segment is 13 bytes away from the first byte of the data stream (13:28). The sequence number between two consecutive segments is increased by the amount of data—compare Frame 4 with the last frame. Frame 4 carried a TCP segment with a payload of 12 bytes, and it was the first segment carrying data; as a result, the next segment's sequence number is the sequence number in Frame 4 augmented by 12.

```
1  09.1064 10.21.4.5.1159 > 10.20.5.14.telnet: S 4196118748:4196118748(0)
     win 32120 <mss 1460
2  09.1068 10.20.5.14.telnet > 10.21.4.5.1159: S 730524282:730524282(0)
     ack 4196118749 win 32120 <mss 1460
3  09.1068 10.21.4.5.1159 > 10.20.5.14.telnet: . ack 1 win 32120
4  09.1143 10.20.5.14.telnet > 10.21.4.5.1159: P 1:13(12) ack 1 win 32120
5  09.1144 10.21.4.5.1159 > 10.20.5.14.telnet: . ack 13 win 32120
6  09.1312 10.21.4.5.1159 > 10.20.5.14.telnet: P 1:25(24) ack 13 win 32120
7  09.1315 10.20.5.14.telnet > 10.21.4.5.1159: . ack 25 win 32120
8  09.1315 10.21.4.5.1159 > 10.20.5.14.telnet: P 25:28(3) ack 13 win 32120
9  09.1315 10.20.5.14.telnet > 10.21.4.5.1159: P 13:28(15) ack 25 win 32120
```

Figure 5: TCP Connection Setup and First Few Frames of a Telnet Session

The sequence number's initial value can be seen in the SYN, SYN ACK sequence. It is assumed that the reader is already familiar with the meaning of the ACK flag. In Frame 5 for example, 10.21.4.5 "ack 13" acknowledges having received the 12th byte from 10.20.5.14 and communicates that it expects to receive the 13th byte

The presence of the MSS Field. In the first frame, the client 10.21.4.5 reports to the server that the maximum TCP payload that it can receive is 1460 bytes. As a result, the server makes sure to packetize the byte stream in chunks no bigger than 1460 bytes. The second frame has the same meaning but for the opposite direction the server notifies the client about its MSS.

Maximum Segment Size

The maximum segment size (MSS) is the largest amount of data, specified in bytes, that a computer or communications device can handle in a single, un-fragmented piece. For optimum communications, the number of bytes in the data segment and the header must add up to less than the number of bytes in the

maximum transmission unit (MTU). The MSS is an important consideration in Internet connections, particularly Web browsing. When the Internet's Transmission Control Protocol (TCP) is used to achieve an Internet connection, the computers being connected must each agree on, and then set, the MTU size acceptable to both. Typical MTU size in TCP for a home computer Internet connection is either 576 or 1500 bytes. Headers are 40 bytes long; the MSS is equal to the difference, either 536 or 1460 bytes. In some instances the MTU size is less than 576 bytes, and the data segments must therefore be smaller than 536 bytes.

As data is routed over the Internet, it must pass through multiple gateway routers. Ideally, each data segment can pass through every router without being fragmented. If the data segment size is too large for any of the routers through which the data passes, the oversize segment(s) are fragmented. This slows down the connection speed as seen by the computer user. In some cases the slowdown is dramatic. The likelihood of such fragmentation can be minimized by keeping the MSS as small as reasonably possible. For most computer users, the MSS is set automatically by the operating system.

TCP Retransmission

Retransmission, essentially identical with Automatic repeat request (ARQ), is the resending of packets which have been either damaged or lost. Retransmission is one of the basic mechanisms used by protocols operating over a packet switched computer network to provide reliable communication (such as that provided by a reliable byte stream, for example TCP).

Such networks are usually 'unreliable', meaning they offer no guarantees that they will not delay, damage, or lose packets, or deliver them out of order. Protocols which provide reliable communication over such networks use a combination of acknowledgments (i.e. an explicit receipt from the destination of the data), retransmission of missing or damaged packets (usually initiated by a time-out), and checksums to provide that reliability.

If data sent has not been acknowledged within a certain time frame, which is a multiple of RTT, the data is retransmitted to the remote host. The retransmission timeout starts out at the RTT and

is doubled with each retransmission. The retransmission timeout is always limited within CFG_MINRTO and CFG_MAXRTO. If CFG_RETRANS_TMO time elapses since the data was first transmitted, the connection is closed, that is. the state is set to CLOSED.

When a timeout occurs, all the unacknowledged data in the output window is retransmitted. The data is repackaged; therefore, the packets will not be identical to the original packets.

For example, if a packet is sent with 10 bytes of data, then a packet with 30 bytes of data is sent and the first packet is lost, 40 bytes of unacknowledged data will be in the output window. When a timeout occurs, all 40 bytes will be sent in one packet (assuming the MSS is greater than or equal to 40).

Delayed ACK

TCP delayed acknowledgment is a technique used by some implementations of the Transmission Control Protocol in an effort to improve network performance. In essence, several ACK responses may be combined together into a single response, reducing protocol overhead. However, in some circumstances, the technique can reduce application performance.

Nagle Algorithm

Nagle's algorithm is a means of improving the efficiency of TCP/IP networks by reducing the number of packets that need to be sent over the network. It was defined by John Nagle while working for Ford Aerospace. It was published in 1984 as a Request for Comments (RFC) with title Congestion Control in IP/TCP Internetworks.

The RFC describes what he called the "small-packet problem", where an application repeatedly emits data in small chunks, frequently only 1 byte in size. Since TCP packets have a 40-byte header (20 bytes for TCP, 20 bytes for IPv4), this results in a 41-byte packet for 1 byte of useful information, a huge overhead. This situation often occurs in Telnet sessions, where most key presses generate a single byte of data that is transmitted immediately. Worse, over slow links, many such packets can be in transit at the same time, potentially leading to congestion collapse. Nagle's

algorithm works by combining a number of small outgoing messages and sending them all at once. Specifically, as long as there is a sent packet for which the sender has received no acknowledgment, the sender should keep buffering its output until it has a full packet's worth of output, thus allowing output to be sent all at once.

Algorithm

The RFC defines the algorithm as

inhibit the sending of new TCP segments when new outgoing data arrives from the user if any previously transmitted data on the connection remains unacknowledged.

Where MSS is the maximum segment size, the largest segment that can be sent on this connection, and the window size is the currently acceptable window of unacknowledged data, this can be written in pseudocode as

```
if there is new data to send
  if the window size >= MSS and available data is >= MSS
    send complete MSS segment now
  else
    if there is unconfirmed data still in the pipe
      enqueue data in the buffer until an acknowledge is received
    else
      send data immediately
    end if
  end if
end if
```

This algorithm interacts badly with TCP delayed acknowledgments, a feature introduced into TCP at roughly the same time in the early 1980s, but by a different group. With both algorithms enabled, applications that do two successive writes to a TCP connection, followed by a read that will not be

fulfilled until after the data from the second write has reached the destination, experience a constant delay of up to 500 milliseconds, the "ACK delay". For this reason, **TCP** implementations usually provide applications with an interface to disable the Nagle algorithm. This is typically called the TCP_NODELAY option.

Limitations

Nagle's algorithm is only usable with TCP.

Other protocols including UDP do not support it.

TCP applications that need fast network response, like Internet phone calling or first-person shooter games, may not work well when Nagle is enabled. The delays caused while the algorithm takes extra time to assemble smaller chunks of data together can trigger noticeable lag visually on a screen or in a digital audio stream. These applications typically disable Nagle.

This algorithm was originally developed at a time when computer networks supported much less bandwidth than they do today. The example described above was based on John Nagle's experiences at Ford Aerospace in the early 1980s, where nagling tradeoffs on their slow, heavily-loaded long distance network made good sense. There are increasingly fewer situations where network applications can benefit from his algorithm today.

Connection Closure

In the normal case, each side terminates its end of the connection by sending a special message with the FIN (finish) bit set. This message, sometimes called a FIN, serves as a connection termination request to the other device, while also possibly carrying data like a regular segment. The device receiving the FIN responds with an acknowledgment to the FIN to indicate that it was received. The connection as a whole is not considered terminated until both sides have finished the shutdown procedure by sending a FIN and receiving an ACK.

Thus, termination is not a three-way handshake like establishment: it is a pair of two-way handshakes. The states that the two devices in the connection move through during a normal connection shutdown are different because the device initiating the shutdown must behave differently than the one

that receives the termination request. In particular, the TCP on the device receiving the initial termination request must inform its application process and wait for a signal that the process is ready to proceed. The initiating device does not need to do this, since the application is what started the ball rolling in the first place.

TCP Bulk Transfer Traffic Profile

Bulk data transfer is software-based mechanism designed to move large data file using compression, blocking and buffering methods to optimize transfer times. Every single second of everyday, data moves from computer to another. Some move over the local area network, other over the wide area network or over the internet.

Data can move from storage to another different physical media. These media can include telephone lines, fiber optics, infra reds, radio signals and many more. But since these media have different capacities and transmission speeds, they need a software to manage the transfer of data from one storage to another. These software can manage bulk data traffic and at the same time perform compression processes to minimize sizes and therefore save bandwidth.

One of the most common ways to transfer bulk of data is by using the File Transfer Protocol (FTP). FTP is a client-server protocol which means that one computer acts as the server and another as the client. The FTP server runs a server software and constantly listens to requests on a network until a client initiates a request for file transfer. When the connection is established, the client can already do many sorts of files manipulation such as downloading, changing files names, creating folders, moving files between folders, and deleting files. An FTP server can services many simultaneous clients depending on the settings. FTP runs over the TCP layers of the TCP/IP, a suite of protocols that the internet is heavily depending on for communication. FTP listens for connection on port 21.

Despite FTP's widespread popularity as a means to transfer bulks of data, it is not without its problem. One of the biggest problems with FTP is security. Many FTP clients do not have secure methods of encryption and usernames, passwords,

configurations, transferred files may be viewed by anyone on the same network using some tools like packet sniffer. But this can be easily overcome by using SSH files transfer protocol (SFTP) or FTP over SSL (FTPS) which adds TLS or SSL encryptions to the FTP.

A protocol which specifically serves bulk data transfers is the Network Block Transfer (NETBLT). This is a transport level protocol designed to rapidly transfer very large quantity of data over the network. The protocol is structured to provide maximum efficiency and the flow is very controlled for better reliability.

TCP Windows

Receive buffers on host machines influence the TCP receive window advertised by the receiver to the sender. In TCP, two windows decide how many segments the sender can transmit without receiving an ACK from the receiver:

- ***Receive window***: This is the amount of space available in the receiver TCP buffer.
- ***Congestion window***: The sender uses this window to avoid causing congestion in the network.

The size of the congestion window is determined by the slow start algorithm and the congestion-avoidance algorithm. The sender needs to ensure that it does not cause congestion in the network and does not overwhelm the receiver. As a result, the sender sends a number of segments whose total size in bytes is less than the minimum of each window. The windowing mechanism limits the number of segments that the sender can send when the transmit buffer is continuously filled. This is typically the case for bulk transfers such as FTP or even HTTP.

In Example 2-8, the client sends ACKs with no data, and the TCP receive window progressively reduces as follows:

- 32,120 bytes at 30.3163 sec
- 30,660 bytes at 30.3181 sec
- 29,200 at 30.3188 sec

This means that the application on the client side is slow at retrieving data from the TCP receive buffer; as a result, the space available in the buffer reduces.

ACK Scheme

In data networking, telecommunications, and computer buses, an acknowledgement (ACK) is a signal passed between communicating processes, computers, or devices to signify acknowledgement, or receipt of message, as part of a communications protocol. The negative-acknowledgement (NAK or NACK) signal is sent to reject a previously received message, or to indicate some kind of error. Acknowledgements and negative acknowledgements inform a sender of the receiver's state so that it can adjust its own state accordingly.

RFC 1122, "Requirements for Internet Hosts," calls for one ACK every two full-size segments. The OS used on the client in figure 6 ACKs follows the standard:

- At 30.3181 sec, the client ACKs the reception of data up to byte 1460 (which is the frame received at 30.3174 sec).
- The next ACK at 30.3188 sec ACKs the reception of bytes up to 4380 (which is the frame received at 30.3182 sec).

```
30.316225 < 10.21.4.5.1122 > 10.20.5.14.www: S 2777311705:2777311705(0)
  win 32120 <mss 1460>
30.316241 > 10.20.5.14.www > 10.21.4.5.1122: S 3157467248:3157467248(0)
  ack 2777311706 win 32120 <mss 1460>
30.316383 < 10.21.4.5.1122 > 10.20.5.14.www: P 1:317(316) ack 1 win 32120
30.316415 > 10.20.5.14.www > 10.21.4.5.1122: . 1:1(0) ack 317 win 31804
30.317416 > 10.20.5.14.www > 10.21.4.5.1122: P 1:1461(1460) ack 317 win 32120
30.317434 > 10.20.5.14.www > 10.21.4.5.1122: P 1461:2921(1460) ack 317 win 32120
30.318150 < 10.21.4.5.1122 > 10.20.5.14.www: . 317:317(0) ack 1461 win 30660
30.318217 > 10.20.5.14.www > 10.21.4.5.1122: P 2921:4381(1460) ack 317 win 32120
30.318227 > 10.20.5.14.www > 10.21.4.5.1122: P 4381:5841(1460) ack 317 win 32120
30.318846 < 10.21.4.5.1122 > 10.20.5.14.www: . 317:317(0) ack 4381 win 29200
30.318880 > 10.20.5.14.www > 10.21.4.5.1122: P 5841:7301(1460) ack 317 win 32120
30.318888 > 10.20.5.14.www > 10.21.4.5.1122: P 7301:8761(1460) ack 317 win 32120

30.318898 > 10.20.5.14.www > 10.21.4.5.1122: P 8761:10221(1460) ack 317 win 32120
```

Figure 6: ACKing Every Other Segment

Other rules that govern how the client or the server ACK include the need for immediate ACKs at the reception of an out-of-order packet and the need to communicate an update of the receive window.

SERVER AVAILABILITY

The ability of a server to be in a state to perform a required function at a given instant of time or at any instant of time within a given time interval. From the user point of view, availability over a specified time interval is the percentage of that interval during which the system was available for normal use.

"The Domain Name Server (DNS) is the Achilles heel of the Web. The important thing is that it's managed responsibly."

—Tim Berners-Lee

High availability is one of the key design considerations of a Data Center. The necessary background on the operating systems aspects of an application and the implications on the availability of the application. No matter how robust the application is, failures are always possible.

Processes and Threads

Server applications are basically processes listening on certain Layer 4 ports. Examples of applications/servers are web servers, common gateway interface (CGI) applications, Active Server Pages (ASP) servers, servlet engines, and so on.

Server implementations differ depending on how incoming connections are handled. A process can be spawned on a per-connection basis by another process listening on multiple ports (such as inetd in UNIX), or it can be a *standalone* server forking when new connections arrive.

Additionally, servers can be categorized as *forking* servers if they rely on creating different processes to serve new connections, or *threaded* servers if they rely on threads instead. The processes in a Linux system running an HTTP forking server. When launching the web server, you have the root process and one child (process 31785). Processes on a Standalone HTTP Server before Receiving Clients' Requests

```
[admin@localhost conf]# ps -ef | grep httpd
root         31784     1  0 21:44 ?        00:00:00 /usr/sbin/httpd
httpserver   31785 31784  0 21:44 ?        00:00:00 /usr/sbin/httpd
```

The main process is like a supervisor to the activities of the other server processes and typically does not perform I/O operations.

When one client sends a connection request, a new process is created. If the same client continues to open connections without closing them, new child processes are created to handle

the new connections. For example HTTP Server Spawns a New Process to Serve a Client's Request

```
[admin@localhost conf]# ps -ef | grep httpd
root     31784     1  0 21:44 ?        00:00:00 /usr/sbin/httpd -D HAVE_PERL
httpserver   31785 31784  0 21:44 ?        00:00:00 /usr/sbin/httpd -D HAVE_PERL
httpserver   31833 31784  0 21:46 ?        00:00:00 /usr/sbin/httpd -D HAVE_PERL
```

Threads are similar to processes, in that they allow concurrent operation, but they execute inside the same process and have less overhead than processes. Processes have separate memory space and file descriptors; threads are separate execution instances that share the same memory space. At a very high level, you can think of threads as separate program counters inside the same process.

FastCGI is an example of a multiprocess application server. Servlet engines (which are basically Java Virtual Machines) are examples of threaded servers.

The main differences between forking servers and threaded servers are as follows:

Performance

Context switching between threads is much lighter than between processes, so the impact of an increasing number of connections on the **CPU** utilization is less with threaded servers than with forking servers. Because of this, threaded servers are more efficient than forking servers.

Robustness

When you run an application inside a single process, the failure of one thread can affect the entire process and the other connections with it. A forking server is more robust because a main process does not deal with I/O operations. Additionally, this process periodically kills the child processes, which prevents memory leaks from crashing the server. If one child fails, other connections still succeed. As a result, forking servers are more robust than threaded servers.

Shared state information

When dealing with a client that sends requests to the same application over a period of time, you typically want to save the session (state) information about this client. Processes do not share memory, which means that dispatching client's connections to different processes forces the application developer to use temporary files or databases to save state information. Threads running in a single process share the memory space, which makes it possible to save the state information across multiple connections of the same application.

Today's servers can be forking, threaded, and hybrid. Hybrid servers keep a number of active processes for the same application and allow each process to serve a number of connections by using the threaded approach. You can make implementations of threaded servers more robust by launching several instances of them. Multiprocess servers provide a solution to the need of sharing state information by writing this information on temporary files shared across servers or by using mechanisms such as session affinity. FastCGI is an example of a multiprocess application server with session affinity. CGI initially had limitations related to the creation of a separate process for each connection request with performance and session state limitations. FastCGI solves the problem by providing the option of running as a multithreaded server or implementing session affinity.

TCP and UDP Connections to a Failed Server

No matter how robust the design of the application is, server failures are still possible for the following reasons:

- Power failure
- Hard disk failures
- NIC failures
- Cable failures
- Software bugs
- Memory leaks
- Disk data corruption
- Configuration errors

Depending on the server failure, the client detects the failure for an established connection as follows:

- ***Server process fails***: The server sends a TCP FIN to the client, or the client sends a TCP segment and receives a TCP RST. In case of UDP, the server sends an ICMP port unreachable message.
- ***Server machine fails or loses network connectivity*:** The TCP connection times out on the client, unless the application has other mechanisms to detect the failure and close the connection.

When a client sends a request to an application for a new connection, there are two possible results, depending on which failure affects the server:

- ***Server process failed:*** The client sends a TCP SYN, and the server sends a TCP FIN. In case of UDP, the server sends an ICMP port unreachable message.
- ***Server machine failed or lost network connectivity***: The client sends TCP SYNs for a number of times (these SYN packets are called TCP SYN retransmissions), depending on the operating system implementation of the TCP stack.

TCP Timeout

When a server is disconnected from the network or powered down, no data flows back to the client, and the connection can stay idle for hours. The TCP stack can keep a connection idle for a configurable amount of time. Because the default is typically two or three hours, however, it is up to the application to age out connections. As an example, in the case of FTP on Linux, if you use proftpd in the configuration file proftpd.conf, you can set the parameter Timeout Idle *TimeoutIdle* seconds (in which the default is 600 seconds) to age out connections that have been idle for the configured timeout.

If the application keeps the TCP connection idle for more than 2 hours, the TCP stack sends a *TCP keep alive* to verify that the peer entity (client or server) is still alive. The sender not receiving a response sends a number of keep lives (a typical number is 9) at an interval of approximately every 1 minute and eventually closes the TCP connection.



Although the recommendation is to not change the TCP keepalive defaults, on most OSs, you can change the value of the TCP keepalive. On UNIX-based OSs, this variable can take the name of tcp_keepalive_interval, tcp_keepidle, and tcp_keepalive_time. In Windows, this variable is called KeepAliveTime. The socket interface also allows the application to decide whether TCP should implement the keepalive mechanism.

SYN Retransmission

When a client tries to open a connection to a failed server, the TCP stack reacts by retransmitting the SYN for a number of times, depending on the OS. Windows 2000 and XP perform two retransmissions; other OSs perform more. The timeout depends on the number of retransmissions and the interval between them.

The following list shows the behavior of three OSs:

Windows

Uses TcpMaxConnectRetransmissions, which is 3 SYNs in Windows NT and 2 SYNs in Windows 2000. This translates into a timeout of 22 seconds in **Windows 2000** (with the first retransmission happening in 3 sec) and 45 seconds in Windows NT.

Solaris

Uses tcp_ip_abort_cinterval, which is the timeout when TCP performs an active open and, by default, is four minutes. The total number of retransmissions depends on other parameters, such as the retransmission interval.

Linux

Uses tcp_syn_retries with a default of 10.

Status of HTTP Applications

The impact of a failed server on TCP and UDP traffic. Another scenario could be an application whose process is alive and capable of accepting new connections, yet is not functioning properly. This analyzes the behavior of a web server and how the server informs the client about possible errors.

From a black box point of view, a web server is a device that receives HTTP requests from a client and returns three types of answers:

- ***Static web page or web page information***: A static web page is the simplest type of output from a web server. Such a page consists of HTML code that seldom changes, together with GIFs and similar objects.
- ***Dynamic web page***: A page is dynamically generated by input from the client (such as a search page), a change in objects that are displayed (such as a playlist on an Internet radio), and information randomly generated (such as stock quotes of commercial banners).
- ***Error message or status code***: All HTTP responses include a status code. The typical status message when everything is fine is "200 Ok." If some failure occurs, a web server returns error messages (for example, "404 not found").

Status codes in the 400 range refer to client-side issues; the 500 range is for status code on server-side issues. Examples of the 400 status codes are as follows:

- ***Bad request 400***: A syntax error exists in the URL.
- ***Unauthorized 401***: The user identified by the user-agent field is not authorized to get the content requested.
- ***Forbidden 403***: The client does not have privileges for the page.
- ***Not found 404***: The requested URL is not available. The URL either is no longer active or is temporarily disabled.

Examples of the 500 error code are as follows:

- ***Internal error 500***: The server has an internal software error that prevents it from sending the HTML document to you.
- ***Service unavailable 503***: The server is processing too many requests.
- ***Gateway timeout 504***: The connection is timing out.
- ***HTTP version not supported 505***: The client is requesting a version that the server does not support.

As emerges from this description, a server can be alive, can accept TCP connections, and can report error codes to the client. In the case of HTTP, the 500 error codes relate mainly to server-side issues.

CONCLUSION

Today's Data Centers are based on client/server architectures that rely on TCP/IP for the network communication. Servers receive the traffic from network interface cards and pass the data to the application via sockets. NICs implement Layer 2 functions such as speed negotiation, collision detection, serialization, and deserialization. NICs can be and should be connected to the infrastructure with teaming for increased availability.

Data that arrives on a NIC is initially processed by the Ethernet driver, then is passed up to the TCP stack, and eventually is copied to the socket buffer. Understanding the basics of packet processing in current OSs helps troubleshooting network problems and making design decisions on issues such as the need for reverse proxy caching, jumbo frames, interrupt coalescing, and the receive window size. Network traces of an interactive TCP application and of a bulk transfer show how the TCP stack operates while client and server communicate. The traces show how TCP optimizes the use of the network and implements flow control during a session. Knowing TCP algorithms and timeouts also helps when troubleshooting problems of performance and making the right choice on which socket options to use when writing an application.

A server application can be thought of as one or more Layer 4 ports waiting for incoming connections and one or more processes. Servers can be made of multiple processes for robustness and can use threads as well for optimized performance and sharing of session information. Even multiprocess servers can fail. In this case, the client receives a TCP FIN or a TCP RST, or, when setting up a new connection, the client does not receive any SYN ACK. Understanding the failure scenarios and how TCP behaves in such conditions is important when designing network services for a highly available application. Network devices provide the infrastructure for TCP/IP applications and can also optimize server performance and increase server availability. Jumbo frames, for example, can optimize the achievable throughput of a server by using frames larger than the regular 1518 bytes. Reverse proxy caching also helps optimize server performance by delivering objects from a kernel-level cache. Load balancing can prevent server overload and can monitor connections between clients and servers to identify malfunctioning servers.

KEY TERMS

Printer	Virtualization
Data	Network
Transmission	IP address
Byte	TCP
CPU	Windows 2000

DISCUSSION QUESTIONS

1. What do you understand the server? Explain its types
2. Explain the internet data center server.
3. Describe the network interface cards (NIC).
4. Define the client and server packet processing.
5. Explain the TCP and server applications.
6. Write the short note on:
   a. Nagle Algorithm
   b. Server Availability
   c. Server Multihoming

REFERENCES

1. "Alacritech SLIC: A Data Path TCP Offload methodology", [online] Available: http://www.alacritech.com/html/techreview.html.
2. J. Chase, "End System Optimizations for High-Speed TCP", IEEE Communications Special Issue on High-Speed TCP, June 2000.
3. S. Gochman et al., "The Intel® Pentium® M Processor: Microarchitecture and Performance", Intel Technology Journal, May 2003, [online] Available: http://developer.intel.com/technology/itj/.
4. "High Performance Network Adapters and Drivers in Windows", December 2001, [online] Available: http://www.microsoft.com/hwdec/tech/.
5. Y. Hoskote et al., "A 10GHz TCP Offload Accelerator for 10Gb/s Ethernet in 90nm Dual-Vt CMOS", IEEE International Solid-State Circuits Conference (ISSCC), 2003.
6. R. Huggahalli, R. Iyer, "Direct Cache Access for Coherent Network I/O", submitted to an international conference, 2003.
7. "iSCSI IP Storage Working Group", Internet Draft, [online] Available: http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-20.txt.
8. R. Iyer, "CASPER: Cache Architecture Simulation and Performance Exploration using Refstreams", Intel Design and Test Technology Conference (DTTC), July 2002.
9. R. Iyer, "On Modeling and Analyzing Cache Hierarchies using CASPE", 11th Symposium on Modeling Analysis and Simulation of Computer and Telecommunication Systems, Oct 2003.
10. K. Kant, "TCP offload performance for front-end servers" in to appear in Globecom, San Francisco:, 2003.
11. K. Kant, V. Tewari, R. Iyer, "GEIST - A Generator for E-commerce and Internet Server Traffic", 2001 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Oct 2001.
12. H. Kim, V. Pai, S. Rixner, "Increasing Web Server Throughput with Network Interface Data Caching", International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 239-250, October, 2002.
13. D. McConnell, "IP Storage: A Technology Overview", [online] Available: http://wwww.dell.com/us/en/biz/topics/vectors_2001-ipstorage.htm.

CHAPTER 3

APPLICATION ARCHITECTURES



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Give taxonomy of applications and hosted servers
2. Focus on integration of applications
3. Understand markup languages: HTML and XML
4. Focus on web servers
5. Explain middleware
6. Describe database access
7. Identify network architecture considerations
8. Perform clustering

INTRODUCTION

The data center is home to the computational power, storage, and applications necessary to support an enterprise business. The data center infrastructure is central to the IT architecture, from which all content is sourced or passes through. Proper planning of the data center infrastructure design is critical, and performance, resiliency, and scalability need to be carefully considered.

Another important aspect of the data center design is flexibility in quickly deploying and supporting new services. Designing a flexible architecture that has the ability to support new applications in a short time frame can result in a significant competitive advantage. Such a design requires solid initial planning and thoughtful consideration in the areas of **port density**, access layer uplink bandwidth, true server capacity, and oversubscription, to name just a few.



The data center network design is based on a proven layered approach, which has been tested and improved over the past several years in some of the largest data center implementations in the world. The layered approach is the basic foundation of the data center design that seeks to improve scalability, performance, flexibility, resiliency, and maintenance.

The purpose of Data Centers is to host application environments and the servers that support them. Enterprise applications range from Internet portals to e-commerce, customer relationship management (CRM), collaboration, enterprise resource planning (ERP), and more.

Enterprises either develop these applications internally and run them on application servers such as IBM WebSphere, BEA Weblogic, or Oracle 9i, or they use commercial applications made by vendors such as SAP, Oracle, and Siebel. Each application is unique in its purpose, architecture, and associated traffic patterns, yet there are common elements to both homegrown and commercial applications.

For example, a significant number of today's applications use a web-based front end, which implies the following:

- The client is a web browser such as Microsoft Internet Explorer or Netscape Communicator.
- The client and the server use HTTP as the protocol for communicating.
- The data transferred by HTTP is in HTML format.
- The server has a process that understands HTTP and can exchange data with the application and with the client.

Because the design of a Data Center results from understanding the application characteristics, this chapter provides a high-level overview of today's applications.

The following list highlights key reasons why it is important to understand application architectures for a successful Data Center design:

- Upgrades to new versions of a given application might require a different network design. For example, an application that once ran the presentation logic (web front end) and the business logic (application middleware) on the same server might migrate to a tiered model where the presentation logic and the business logic run on separate servers.



- The need to integrate applications (a process that is referred to as EAI) might require the creation of new traffic paths between servers that were previously isolated. For example, you can address the need to web-enable legacy applications running on mainframes with middleware software running on application servers. This solution requires a new communication path between application servers and mainframes in addition to or in replacement of the direct connectivity between terminal clients and mainframes.
- The security design of a Data Center requires understanding the application environment to create security zones and to deploy IDS signatures in a meaningful way.
- The need to scale the performance of application servers might require hardware load balancing and SSL offloading. Taking full advantage of hardware load balancing requires also understanding how web and application servers are deployed.
- The need to deploy clustered applications for disaster recovery requires data replication, which in turns requires understanding how applications store data.
- The need to exchange information with business partners on extranet environments requires the right network design and security considerations to connect the two networks and to allow a secure communication exchange between them.

TAXONOMY OF APPLICATIONS AND HOSTED SERVERS

The main goal of Data Centers is to host applications that the end user invokes remotely to retrieve data or perform some task, such as placing orders, making reservations, communicating with a remote party, or retrieving documentation.

The many categories of enterprise applications include portals, e-commerce applications, ERP, CRM, supply-chain management, and collaboration tools.

"I'm a programmer. I like programming. And the best way I've found to have a positive impact on code is to write it."

—Robert C. Martin

The purpose of portal applications is to operate as the gateway to various information sources and tools tailored to a specific community of users. Portals are web pages hosted on web servers: they provide information in the form of HTML pages transmitted on top of HTTP. HTML pages in turn support a number of other objects, such as Word or PDF documents, other web pages, and multimedia content via anchors. (Read the section "HTML.")



E-commerce and business-to-business applications make it possible to browse catalogs and place orders securely. The user interface for these applications is still built in HTML format and carried on HTTP. Transactions are secured with SSL.

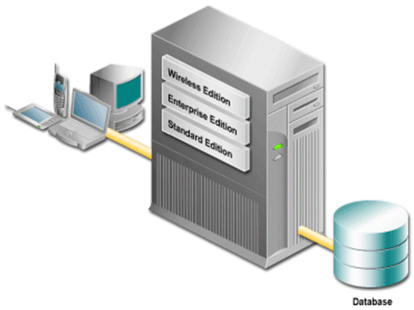
Collaborative applications integrate the communication between remote parties by using e-mails, document sharing, online messaging, and conferencing.

From an architectural point of view, applications execute on a number of servers, which provide the user interface (web servers), the application logic (application servers), and a system to save and retrieve data (database servers); for more information, read the section, "Multitier Applications."

Besides these servers, several other services support applications and network management: they include file servers, Domain Name System (DNS) servers, directory servers, RADIUS servers, and Certificate Authority (CA) servers.

This list summarizes the servers that you can typically find in a Data Center:

- ***Web servers***—These servers provide the presentation function for most applications. The simplest use of a web server is as a portal for a company on the Internet. For more information, read the section, "Web Servers." Apache, Netscape Enterprise Server, and Microsoft Internet Information Server (IIS) are examples.
- ***Application servers***—These servers execute the business logic function of a multitier application. For more information, read the section, "Server-Side Programming." Examples are BEA Weblogic, IBM WebSphere, Oracle 9i, Apache Tomcat, and Sun ONE application servers.



- ***Database servers***—These servers store and manage records of data. One of the key functions of a database server is maintaining structured tables and returning the content in the order and format that is requested by the client application. For more information about database servers, read the section, "Database Access." Examples of database servers include Oracle 9i, IBM databases (DB2, Informix, IMS, and U2), and Sybase.
- ***E-mail servers***—Electronic mail is probably the most popular collaboration application on the Internet, together with HTTP applications. The architecture of e-mail applications usually consists of three components—the user agent on the client's desktop, a mailbox that stores e-mails and interacts with the user agent, and a mail server. The protocols used by these components are the Post Office Protocol (POP3, RFC 1939), the Internet Message Access Protocol (IMAP, RFC 3501), and the Simple Mail Transfer Protocol (SMTP, RFC

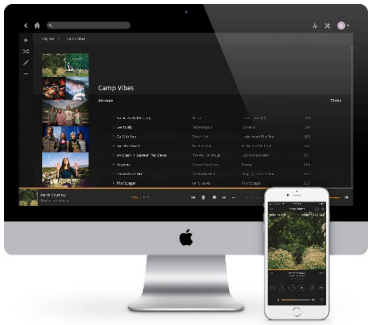
2821). Examples of e-mail servers are Sendmail Inc., Microsoft Exchange, Lotus Notes and Domino, Oracle Collaboration Suite, and Sun ONE messaging servers.



- ***File servers***—These servers provide shared remote storage for other servers in the Data Center and users on the network. The protocols used to access files on file servers are either the Network File System (NFS, RFC 3530) protocol (for UNIX-based operating systems) or the Common Internet File System (CIFS) protocol (for Microsoft Windows applications). Examples of file servers include Network Appliance Filers, EMC Celerra, IBM TotalStorage, and Microsoft Windows Storage Server 2003.
- ***Directory servers***—These servers provide a database in a predefined format (a subset of International Organization for Standardization [ISO] X.500) that stores enterprise information about users, printers, servers, and digital certificates. The Lightweight Directory Access Protocol (LDAP, RFC 3494) is used by client applications (such as servers executing middleware software or authentication servers) to consult directory servers. Examples of directory servers include the Microsoft Active Directory, Novell eDirectory, Sun ONE directory server, Oracle Internet Directory, and IBM **directory server**.
- ***DNS servers***—DNS servers, among other services, provide the translation of domain names into their associated IP addresses. "DNS Essentials and Site Selection Considerations," describes the DNS protocol

and its uses. Examples of DNS servers include BIND, Novell DNS server, Microsoft DNS server, and Cisco Network Registrar.

- ***DHCP servers***—Dynamic Host Configuration Protocol (DHCP, RFC 2131) servers assign IP addresses dynamically to requesting clients. Typical clients are desktop machines in an enterprise network. Examples of DHCP servers include the Internet Software Consortium (ISC) DHCP server, Novell DHCP server, Microsoft DHCP server, IBM DHCP server, and Cisco Network Registrar.
- ***RADIUS servers***—Remote Access Dial-In User Service (RADIUS, RFC 2865) servers—and in general, authentication, authorization, and accounting (AAA) servers— authenticate network administrators, dial-up users, virtual private network (VPN) clients, Internet users (with HTTP cut-through proxy or authentication proxy), and desktop users (with IEEE 802.1x). Examples of authentication servers include CiscoSecure Access Control Server (CSACS) and Microsoft Windows 2000 Internet Authentication Server (IAS).
- ***CA servers***—CA servers are required to build the public-key infrastructure (PKI) for the distribution of public keys when deploying SSL or IP Security (IPSec) applications. E-commerce applications (i.e. applications using HTTP on SSL), IPSec VPNs, SSL VPNs, TN3270 on SSL, and identity-based network access (802.1x authenticated desktops) are examples of applications that require a CA server. CA servers can be hosted in-house or outsourced. Examples of CA servers that you can use to build an in-house PKI are Entrust servers, Sun ONE certificate servers, and Microsoft Windows CA servers.
- ***Streaming servers***—Streaming servers allow the distribution of Video on Demand or live events to desktops in the enterprise network or to Internet clients. Streaming servers include Cisco IPTV, Apple QuickTime Streaming, Microsoft Windows Media, and Progressive Real servers.



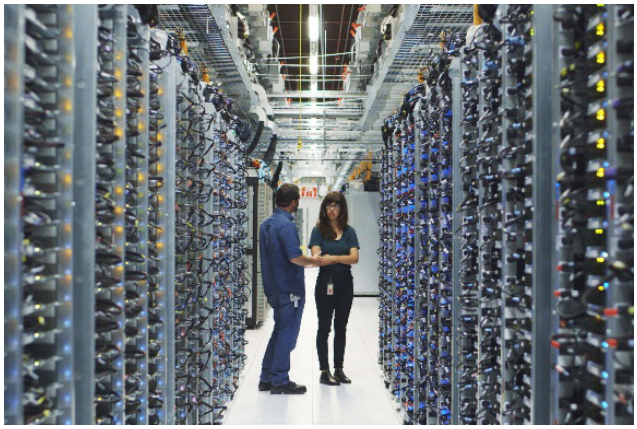
- ***TN3270 servers***—TN3270 servers provide client access to mainframes' Systems Network Architecture (SNA) applications via Telnet. A TN3270 server is a computer that implements both a TCP/IP stack and an SNA stack. The software running on the TN3270 server translates the characters from the format used on terminals to the ASCII format and vice-versa. It converts clients' characters in SNA data streams. Several standards, such as RFC 1576 and RFC 2355, describe how the TN3270 implementations work.

INTEGRATION OF APPLICATIONS

Data Centers typically host servers running several different operating systems and applications based on diverse software architectures.

> *"All race conditions, deadlock conditions, and concurrent update problems are due to mutable variables."*
>
> **—Robert C. Martin**

The diversity of hardware and software platforms in a single Data Center is the result of many factors, which include the rapid evolution of application technologies, the consolidation of server farms from companies that have been merged, and simply the parallel and independent implementation of software without a plan to share data across multiple systems.



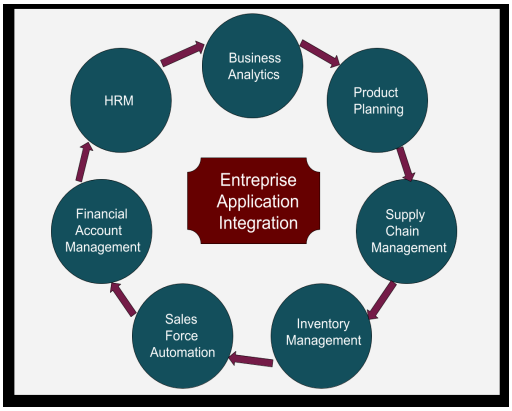
This diversity of applications comes with a management burden and the need to share the same data. These challenges are the reason for the current trend of simplification and integration. This section explains what the Enterprise Application Integration (EAI) is, which software technologies have emerged to make the integration possible, and what is the likely impact on the network traffic patterns.

Enterprise Application Integration

Enterprise business-critical applications such as databases, reservation systems, and partsordering tools have existed long before the Internet, and they were originally developed to execute on systems such as IBM mainframes.

In recent years, enterprises have decided to let remote internal users, business partners, and online customers access their applications. Rewriting applications to make them fully webbased poses a risk to stability, complexity, and availability, which have long been solved in the existing software.

The solution is to web-enable existing applications. This task is not just about placing a web server in front of a mainframe but providing an easy-to-use interface and consolidating and automating business processes. This operation can be complex because a business process requires several different applications, which might not be designed to interoperate with each other. The integration process is typically referred to as Enterprise Application Integration (EAI).



The software design model that simplifies the integration and development of applications is called the n-tier model, and the software that glues these servers and applications together is called middleware.

Middleware can translate HTTP requests into the specific protocols of legacy applications. An example of this approach comes from the technologies that enable the integration of IBM Customer Information Control System (CICS) applications. IBM's redbook Revealed! Architecting Web Access to CICS (http://www.redbooks.ibm.com/) provides an excellent reference.

A higher integration between applications requires the use of distributed object technologies, such as Enterprise Java Beans (EJB), Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA), or Web Services. This topic is described in the section, "Middleware."

Network Design Implications of EAI

EAI has an important implication for the network design. The immediate consequence is that servers which were previously isolated need to communicate. This step requires creating new communication paths and security zones, which translates into Layer 2, Layer 3, and security considerations.

The network design that matches the software n-tier model segregates server farms of different types to provide isolation from possible attacks. You can design this segregation with VLANs and access control lists (ACLs) on routers or enforce it with firewalls.

The network designer needs to assess the impact of server-to-server protocols and database access protocols. Remote Procedure Calls (RPC), Java Remote Method Invocation (RMI), Object RPC (ORPC), Internet Inter ORB (Object Request Broker) Protocol (IIOP), Simple Object Access Protocol (SOAP), and various SQL transport protocols might open dynamically negotiated ports. Addressing this risk is often a challenge in designing ACLs, or it can be as easy as enabling the appropriate fixup on a firewall.



The other implication of application integration is the increased use of web servers for the presentation of the user interface. User interfaces are built as HTML pages carried in the HTTP payload.

The use of web-based applications increases the number of HTTP requests, and the use of web-associated technologies increases the number of DNS requests. Client machines sending HTTP requests to servers initially send DNS requests by indicating the server name in order to determine its IP address. This step is necessary for a user requesting a web page from a browser as well as a Java application calling a remote method. The higher number of DNS and HTTP requests (HTTP uses TCP, thus increasing the number of TCP connection requests) drives the need for scalability and load distribution between a pool of servers.

Permitting access to internal applications from business partners also requires careful design in the extranet. In this environment, network address translation (NAT) is fundamental to avoiding conflicting private IP addresses. (Both companies might use the same private address range.) NAT is a Layer 3

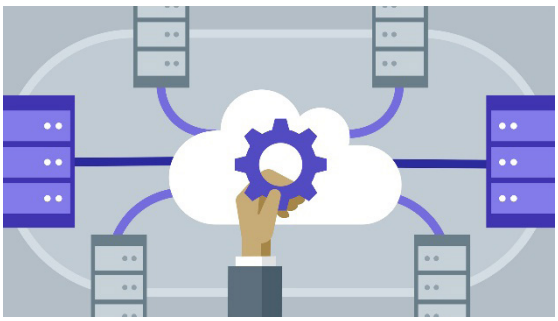
function that needs to be application-aware.

Another, less apparent, implication of EAI is the need to provide application functions from network devices. Consider the various operating systems and server platforms in a Data Center and the need to provide similar services to these applications, such as load distribution, server health monitoring, encryption, and remote access. By providing these application services on network devices, it is possible to ease the integration and the consolidation of multiple server platforms. For example, a network SSL device can decrypt traffic for UNIX-based servers as well as Microsoft Windows servers. The SSL configuration is unique across multiple servers. The same concept applies to the use of load balancers or VPN concentrators.

The following section discusses a model to build open application environments that can easily accommodate the need to integrate various hardware and software platforms, besides being scalable and resilient to failures.

Multitier Applications

Most enterprise applications today are developed according to the multitier or n-tier model. Applications developed to follow the n-tier model are referred to as multitier applications as the various application functions are provided by different server tiers. According to this model, the application functions are divided into the following software tiers:



- *The client tier*—The client software (usually called the user agent) allows the interaction between the user and the application. With web-based applications, the client tier (usually a browser) renders the user interface (which

is generated by the presentation tier) by interpreting the HTML, Java applets, and ActiveX controls.

- *The presentation tier*—This software provides the visualization functions for the application. The presentation tier comprises static objects such as images, form fields to receive the client's input, and dynamically generated objects to display the results of the computation of the application tier. On web-based applications, the presentation tier is implemented by web servers.
- *The application tier*—This software provides the business logic. The application tier receives procedure invocations from the presentation tier, to which it returns the result of the computation. The application tier also communicates with the database tier to store and retrieve data. You implement the application tier either with scripting languages on web servers or on application servers. The typical technologies are CGI; ASP; Java Server Pages (JSP); Java servlets; or, for more complex applications, object-oriented middleware such as EJB, DCOM, and CORBA.
- *The database tier*—This software stores application data, such as catalogs and user information. The software that implements the business logic connects to database servers to retrieve and save data.

Some of these functions are provided either on a single server machine or on multiple machines that communicate over the network.

How the functions are separated and whether they execute on multiple machines depends on how the vendor architected the application (if the tiers are modular enough to be placed on different servers) and the scalability and resiliency requirements of the application. Figure 1 shows some possible multitier Data Center designs.

Part a of Figure 1 shows a two-tier model. The PC is the client and implements most of the processing logic, which is why the client software is called a thick client. The PC connects to a remote database server to retrieve data.

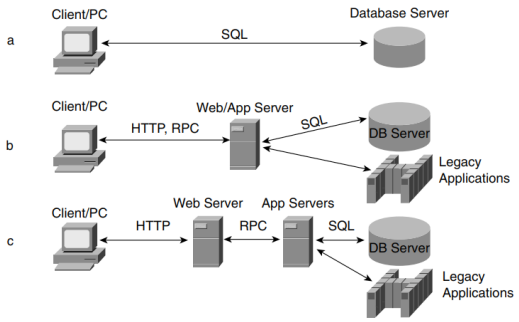


Figure 1: The n-Tier Model.

Part b of Figure 1 shows a three-tier architecture. The client mainly implements the presentation logic; the application processing happens on the application server, which in turn retrieves data from a database server. The software on the PC is considered a thin client, which means that it does not perform much processing besides rendering the information returned by the server. Part b of Figure 1 also shows that the application server can provide the gateway function with legacy applications.

Part c of Figure 1 shows a four-tier architecture. The client has not changed, but the functions of the application servers are separated in a presentation function, typically implemented by web servers, and an application logic function, implemented on application servers. The application servers retrieve data from database servers. The application servers also provide access to legacy applications from web-based applications.

MARKUP LANGUAGES: HTML AND XML

The web browser is the ubiquitous client software in use today. Browsers send HTTP requests (typically HTTP GET and POST requests) to web servers.

> *"The perfect kind of architecture decision is the one which never has to be made.."*
>
> **—Robert C. Martin**

Clients might request static content by sending HTTP GET requests for such objects as /index.html and /image.gif, or they might send variables to the remote application by appending variable names and values to HTTP POST or GET requests.

Web and application servers return HTTP responses with HTML data. HTML describes how a browser should display data, which is why HTML is suitable for formatting the user interface of an application.

One challenge with using HTML to render documents is that users might retrieve information by using clients other than web browsers. These include wireless phones, handheld devices, and IP phones. Formatting documents for these clients is different from formatting for desktop computers because of the differences in the screen size and the capabilities of the software that displays the text.



XML is a markup language that is used by web-based applications to describe data without formatting information. An XML document can be formatted differently according to the client that is requesting it.

Besides providing the functions just described, XML is becoming a key component in building business-to-business and e-commerce applications because it provides a standard way to describe data. In fact, you can use XML in conjunction with SOAP to enable the communication between remote servers of different operating systems on top of HTTP.

Load balancing, SSL offloading, caching, and content-transformation devices are designed to integrate with web-based applications. For example, you can design the HTML data and the forwarding decisions of a load balancer to interoperate: the anchors in the HTML code can be matched by the load balancer, which is useful for session persistence. As another example, a load balancer can be dynamically configured by an application by means of XML messages.

Knowing basic concepts about HTML and XML is necessary to understand the functionalities of network products that are HTTP and SSL aware, and as a consequence, it helps you take advantage of their functions and integrate these products in a Data Center environment.

HyperText Markup Language or HTML

HTML describes how a browser should display text on a screen. Servers use HTML to build the user interface to applications in the Data Center.



HTML provides the following functionalities:

- Text and paragraph formatting
- Inclusion and formatting of images
- Linking to other documents or web pages
- Creation of tables
- Creation of forms
- Creation of image maps

Suppose that the user enters the URL http://www.example.com/ in a browser. The browser resolves the DNS name www.example.com to the IP address, such as 192.0.2.80, and it sends an HTTP request to 192.0.2.80. The web server returns the content of the local file index.html. The HTTP response includes the HTML document. (The HTTP headers indicate Content-Type: text/html.)

Example 1 shows a simple HTML document. The <html> and </html> tags delimit the document. The tags surround the document title.

Example 1: Example of HTML Code from index.html

```
<html>
<head>
<title>Personal Page</title>
</head>
<body>
<h2><img src="images/bullet.gif"> Data Center Design </h2>
<p> Click on the following links to download the documents </p>
<p><a href="http://www.example.com/document1.zip"> Design Best Practices </a></p>
<p><a href="http://www.example.com/document2.zip"> Load balancing overview </a></p>
</body>
</html>
```

The main portion of the document constitutes the body, which is enclosed in the <body> and </body> tags. Within the body, the and tags delimit the paragraphs. An example of a paragraph from Example 1 is " Click on the following links to download the documents ". The browser simply displays the sentence as a separate paragraph. Figure 2 shows how the browser renders the HTML code of Example 1.

Notice the bullet image at the left of the heading "Data Center Design." You insert images by using the <img> element, which specifies the source for the picture. When the browser reads the tag, it sends an HTTP request for the image. The web server sends an HTTP response with Content-type: image/gif and the image data. The browser displays the image according to what the HTML code specifies.

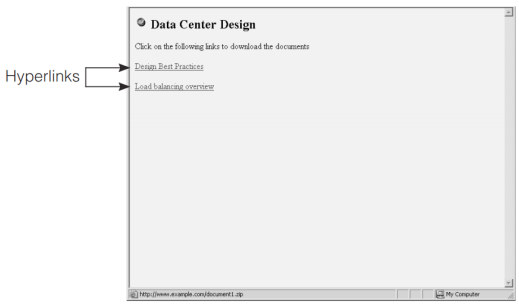


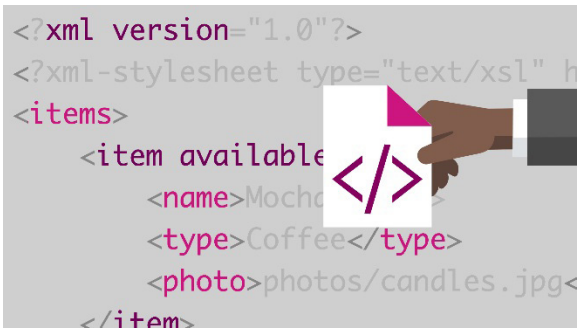
Figure 2: Rendering of a Simple Web Page on a Browser.

Special text visible in Figure 2 appears in bold and underlined format. The user can click on this text, and depending on the content, the browser either displays a new page or downloads the content referenced by this text.

In Example 1, it is possible to read the URL associated with the text visible in Figure 2. The text displayed by the browser, "Design Best Practices," is associated with the URL http://www.example.com/document1.zip. You make this association by using the <a href> tag, also called an anchor. When the user clicks the "Design Best Practices" text, the browser resolves www.example.com to its IP address, sends an HTTP GET to the IP address, and retrieves the file document1.zip using HTTP.

Extensible Markup Language or XML

XML is a data-description language. Whereas HTML describes how some text should appear on a computer screen, XML describes the type of data present in the text. Think of a simple analogy with the records of a database. The fields in the record have a name that describes the field type (such as the name, address, and phone number of a customer).



An XML document includes the field descriptions (such as <name>,<address> , and <phone number>) and their values. The application that uploads the XML file can exchange this information with another application, or it can format the text for a client. This flexibility makes XML suitable for server-to-server communication, where the presentation function is not required, and for user interfaces that need to be displayed on multiple client types: PCs, wireless devices, and handhelds.

Example 2 shows a simple XML document. The tags <title> and </title> include the title of the book, and the tags <author>and </author> include information about the authors, which in this case is the names included in the tags <name>and </name>.

Example 2: Example of an XML Document, book.xml

```
<?xml version = "1.0"?>
<book>
   <title> Data Center Fundamentals </title>
   <subtitle> Fundamental Protocols, Layer 2, Layer 3, Security and Load Balancing
   </subtitle>
   <author id="1">
     <name> Mauricio Arregoces </name>
   </author>
   <author id="2">
     <name>Maurizio Portolani </name>
   </author>
   <editor>Cisco Press</editor>
</book>
```

As it emerges from Example 2 in XML, the tags describe the type of data included in the tags. For the purpose of this book, this description of XML is sufficient; for more information, refer to the URL http://www.w3c.org/XML/. What is really important to understand in this section is the use of XML in a Data Center.

An XML document can be rendered differently based on the user agent that is requesting it. An application server can generate data in XML format. An XML parser on the server loads the XML document into memory; transcoder software provides the formatting for the XML document based on the user agent that is requesting it. If the user agent is a web browser, the document is formatted in HTML; if it is a handheld device, the document is formatted with compact HTML (cHTML); and if it is a wireless phone, the document is formatted with the Wireless Markup Language (WML).

Another important application of XML is facilitating the exchange of data between servers. SOAP makes it possible to invoke remote calls and to exchange messages between servers of different operating systems. SOAP relies on XML for the description of data. SOAP traffic is carried on HTTP.

User Agents

The client side of an application is software that runs on a personal computer, a workstation, a personal digital assistant

(PDA) device, or a web-enabled phone and uses the network to send requests to a server in a Data Center. Client applications, also called user agents, are categorized as follows:

- *Thick clients*—Clients that perform significant amounts of processing and access servers mainly to retrieve data.
- *Thin clients*—Client applications that mainly display the results of processing done on servers. A thin client deals with the presentation layer of the application. A web browser is an example of a thin client.



Web browsers, e-mail clients, and streaming players are examples of client software. Web browsers implement presentation functions based on the HTML code sent by the server and basic functions such as data validation and confirmations based on code sent by the server and executed locally on the client. Among the key technologies used to implement these functions, you should be familiar with the following: helpers, plug-ins, JavaScript, applets, and ActiveX controls. Most vendors' applications rely on one or more of these technologies for functions implemented on the client side.

Browsers

Web browsers are the most popular Internet client. A web browser is a client application that communicates with a web server, and its main purpose is to render HTML files on a computer screen. The user interacts with the browser either by entering URLs into the Address bar of the browser or by clicking the hyperlinks of HTML documents.

In addition to HTML, a browser renders several multimedia types. When a specific media type is not supported natively in the browser, helper programs and plug-ins help with this task.

The browser features that interest the network designer the most are

- The support for HTTP 1.1
- How browsers handle the resolution of domain names
- The support for SSL
- The use of cookies for HTTP session-state management
- The use of proxy servers for Internet access

The most popular browsers at the time of this writing are Microsoft Internet Explorer and Netscape Communicator

Helpers and Plug-Ins

The browser launches a helper application when it receives a content type that it cannot natively display. For example, a browser that requests a video and audio stream launches a player, which in turn contacts the streaming server.



Figure 3 shows the helper applications for Netscape. For each Multipurpose Internet Mail Extensions (MIME) type, there is a handler, which is the **helper application**. For example, Figure 3 shows that the Content-type video/x-ms-wmx is handled by Microsoft Windows Media Player.

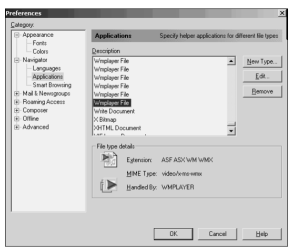


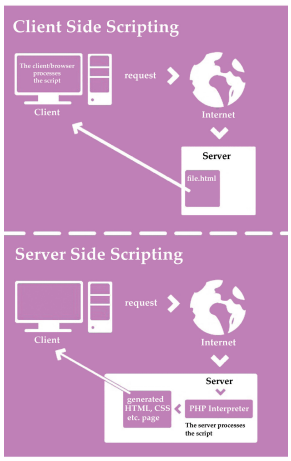
Figure 3: Helper Applications.

Plug-ins provide a more integrated solution than helpers do. For example, a browser that requests a streaming content can load the player plug-in into its memory instead of launching it as an external application. Plug-ins applications run from the browser memory, and consequently, the content usually appears within the window of the browser.

With Netscape, the plug-in files are usually located in the directory C:\Program Files\ Netscape\Communicator\Program\ Plugins. These files are mainly dynamic link libraries (DLLs) and are linked dynamically when the browser is launched.

Client-Side Programming

A browser is a thin client: it mainly implements the presentation function. Sometimes, an application needs to provide functions dynamically, such as executing simple procedures like data validations. You provide these functions with client-side scripting. Compiled code such as Java applets and ActiveX controls can execute more complex operations.



The main technologies for client-side programming are the following:

- *JavaScript*—JavaScript is a client-side scripting language that is interpreted by the browser. JavaScript provides the HTML code with several functionalities that can be implemented on the client side (in the browser). They include control of the browser windows, graphic capabilities, and mathematical computations.

- *Applets*—Applets are Java applications compiled in bytecode format that can be executed in a Java virtual machine (JVM) on any operating system. Java applets and JavaScript are two different technologies, but they both use Java syntax. A Java applet is a compiled bytecode that runs in the browser's JVM; JavaScript is the source-code part of the HTML code, and it must be interpreted by the browser. Applets are independent of HTML code, which merely references them. Applets are downloaded by the browser from a web server. Applets can connect to the remote server with sockets, with the RMI protocol, and with IIOP. Applets can connect directly to a database with the Java Database Connectivity (JDBC) application program interface (API).
- *ActiveX*—ActiveX means many things, but for the purpose of our discussion, we are interested in ActiveX controls, which are developed by Microsoft and similar to applets. ActiveX controls are binary code downloaded to the client machine by a browser. They execute on the client machine just as applets do. An ActiveX control can open connections to a remote server, just as an applet does. The protocol that is used to invoke remote methods is DCOM.

WEB SERVERS

The first tier of servers for web-based applications consists of web servers. Web servers process HTTP requests and return HTTP responses.

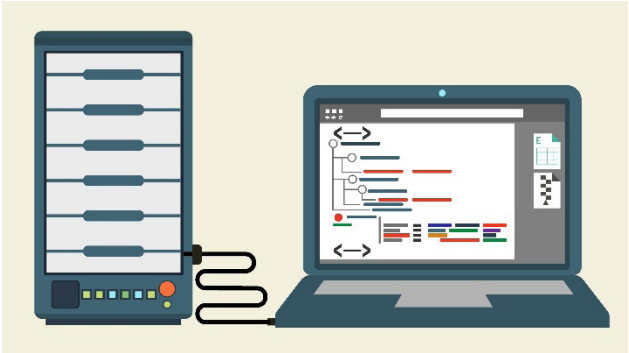
"Just by making the architect role explicit, a team can effectively resolve many technical conflicts."

—Yegor Bugayenko

"Server Architecture Overview," already illustrates how to configure an HTTP server and the strategies used to process requests from multiple users simultaneously with multiprocessing and multithreading.

The most basic HTTP server is a device that accepts HTTP requests from browsers; looks up files on the local disk; and returns text, images, documents. A web server also needs to

be able to terminate SSL: it needs to decrypt secure HTTP (HTTPS) requests, retrieve the requested object, encrypt the HTTP response, and send it back to the browser.



HTTP servers can return a wide variety of Content-types. Examples are text/html, image/ gif, image/jpeg, and video/ quicktime. HTTP uses MIME-like messages to describe the content of HTTP responses so that the browser can correctly visualize it.

"Markup Languages: HTML and XML" provides an example of static HTML content (refer to Example 1). A server machine hosting a web server can also provide dynamic processing by executing applications, connecting to the database servers, and delivering the results to the client in HTML format.

The application technologies include CGI, JSP, and ASP. The section, "Server-Side Programming" provides more detail about these technologies.

Server-Side Programming

Web servers can provide static content, business logic, and database access. The main capability offered by web servers is delivering static content. The business-logic intelligence and the capability to access databases is provided by additional modules, to which the web server hands off specific fields of a URL, or by application servers, to which the web server delegates the processing of the business logic. The case study at the end of the section provides an example of how clients interact with web-based applications by means of HTTP and dynamically generated HTML.

Web Programming Technologies Overview

One of the easiest ways to run business logic on a web server is to execute programs on the web server itself. With server-side programming, the client (the browser, basically) receives HTML code that is the result of executing the program on the server. With server-side scripting, the client cannot see the source code that generates the HTML page (which is different from client-side scripting).

Server-side programming technologies can be categorized as follows:

- *Scripting*—Scripting consists of embedding lines of code in HTML files that are named with the extension of a scripting language, which is interpreted at runtime by the server. Examples of scripting languages are PHP: Hypertext Processor (PHP) and JavaScript.
- *Server-specific APIs*—These APIs are the interface to procedures provided by the web-server software. These APIs are invoked from the HTML code. Examples of this technology are Internet Server API (ISAPI) and Netscape Server API (NSAPI).
- *Precompiled code*—This methodology consists of compiling the software and invoking the compiled code from the web server. Examples of this technology are Java servlets and server-side ActiveX components.

The following list includes the main server-side programming technologies:

- *ISAPI*—These APIs are specific to the Microsoft IIS web server and can be invoked from within the HTML code.
- *NSAPI*—These APIs are specific to the Netscape web server and can be invoked from within the HTML code.
- *PHP*—This scripting language can be embedded into the HTML code and is interpreted at runtime; the web server has a built-in interpreter for PHP. The delimiters enclose the scripting code. The files containing PHP code have the .php extension.
- *Server-side JavaScript*—Server-side JavaScript is equivalent to client-side JavaScript, except that the server-side script is delimited by the tag instead of

- *Server-side ActiveX components*—When developing ASP pages, it is possible to include ActiveX server components written in a programming language that can differ from the language used for scripting. The compiled code can be called from the ASP page.

Server-side programs can run from the memory space of the web server—with the risk that incorrect programming can affect the web server itself—or they can run as a separate process. Comparing CGI and Java servlets helps you understand the implication of either architecture.

CGI is one of the first technologies to be adopted for **server-side programming** of webbased applications. CGI programs spawn separate processes for each request. The life of the program is kept short, which minimizes the impact of code that could be buggy on further requests.

This behavior provides robustness for CGI applications but is also subject to the following limitations:

- *Performance*—CGI creates processes for each client request even when multiple requests come from the same client.
- *Complex data persistence*—Creating separate processes for requests that belong to the same client session requires CGI to continuously save on the disk temporary data that must be shared.
- *Complex handling of database server connectivity*—Multiple requests from the same client that require access to the database imply closing and opening of new database connections because processes are created and killed for each connection.

The preceding list highlights some of the reasons that brought about Java servlets. Unlike CGI, servlets use multiple threads instead of processes to handle multiple clients requesting the same servlet. Because of the threaded architecture, data persistence for a returning client is typically easier to manage than in CGI.

Java servlets and JSP execute in a JVM, which is always running. JSP and servlet code is only compiled the very first time that it is requested.

Each approach has pros and cons: running scripts in the web-server memory space makes the application faster, but

executing scripts in a separate process makes the application more robust.

Example 3: Example of HTML Code Dynamically Generated by a Servlet

```
<html>
<head><title> Simple example of HTML code </title></head>
<body>
<p> Your choice is Rock </p>
<p><a href="SessionTracking?musicpreference=classical" > Click here to select the
  classic collection </a></p>
<p><a href="SessionTracking?musicpreference=rock" > Click here to select the rock
  collection </a></p>

<form action="SessionTracking" method = GET>
<p> Fill in with your music preference </p>
<input type=text name=musicpreference>
<input type=submit value="click here to submit">
</form>

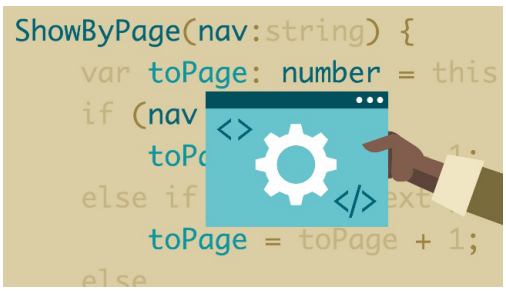
</body>
</html>
```

MIDDLEWARE

Multitier web-based applications require the integration between multiple server tiers, some of which could come from legacy systems. Middleware is the technology that allows this integration because it lets application software running on diverse operating systems communicate.

"Every program attempts to expand until it can read mail. Those programs which cannot so expand are replaced by ones which can."

—Jamie Zawinski



Middleware functions are often categorized as follows:

- *Object middleware*—This category allows the creation of distributed object-oriented applications. CORBA-based brokers, RMI, ActiveX/DCOM, and SOAP belong to this category.

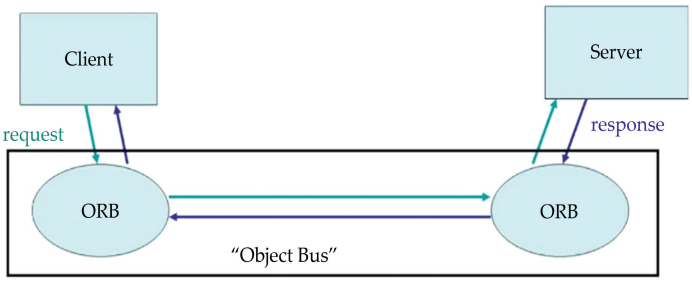
- *Transaction middleware*—This middleware updates several databases and resources with Atomic, Consistent, Isolation, Durable (ACID) transactions; that is, if the transaction does not succeed, a rollback procedure is performed. Transaction Processing (TP) monitors such as IBM CICS provide these functions.
- *Messaging middleware*—Messaging is typically used for asynchronous communication. Integrating applications across a WAN is one example of using messaging middleware. IBM WebSphere MQ and BEA MessageQ are examples of messaging products.
- *Database middleware*—JDBC and Open Database Connectivity (ODBC) interfaces and database management systems (DBMSs) are examples of this middleware type. DBMS solutions typically provide a number of services such as support for load distribution, redundancy, and security.

Each category of middleware can exist as a software product or is part of a software product that implements multiple functions.

The most widely used middleware software environments today are

- *Java 2 Enterprise Edition (J2EE)*—Sun Microsystems's architecture for multitier applications is based on Java. Among its assets is the portability of the architecture across multiple operating systems because it is based on Java and because it is compatible with most web servers.
- *Microsoft .NET*—Previously called Distributed internetwork Application Architecture (DNA), Microsoft .NET is Microsoft's architecture for multitier applications. .NET executes on Windows operating systems (such as Windows 2000 Server and Windows XP). Among the advantages of this architecture is the number of different languages supported.
- *CORBA*—Middleware that is compliant with CORBA lets you develop applications in a number of different languages and permits communication between computers running different operating systems.

Today's applications, whether commercial or home-grown, typically rely on the J2EE technology, Microsoft .NET, or CORBA.



J2EE-based software is characterized by the use of applets for client-side programming, servlets, and JSP for server-side programming and uses JDBC as the interface for database access. The J2EE technology for distributed object-oriented computing is based on EJB.

Microsoft .NET software relies on ActiveX controls for client-side programming, ASP for server-side programming, and ODBC and ActiveX Data Objects (ADO) as the interface for database access. The .NET technology for distributed object-oriented computing is based on DCOM objects.

CORBA is an open architecture that provides transport for the communications between objects in the network and also provides services such as life cycle for the objects, persistence, naming for remote objects, transaction services, and security. You can write the applications in several languages, including Java and C++. Notice that CORBA is not a product but a specification maintained by an organization called the Object Management Group (OMG).

Table 1 maps the client and server functions previously examined in this chapter to the J2EE and .NET architectures.

Table 1: Main Software Architectures for Web-Based Applications and Associated Technologies

	Client Application	Scripting Language	Database Interface	Components	Transport Protocol
J2EE	Browser, Java applet	JSPs and servlets	JDBC	EJB	RMI
.NET	Browser, ActiveX control	ASPs	ODBC	DCOM objects	ORPC

Components: EJBs and DCOM

Among the key software characteristics of the J2EE and .NET technology is that they allow the development of object-oriented distributed applications, which are based on components. Components are small binary applications that can be used by several other applications. Java Beans and Component Object Model (COM) objects follow this paradigm.

COM and Java Beans are locally available components. The logical evolution of this paradigm is to make the components available across the network. Remotely accessible Java Beans are called EJB, and remotely accessible COM objects are called DCOM objects.

In multitier environments, remote objects typically implement the business logic and run on a separate server from the one that provides the presentation logic.

In a J2EE-based application, Java servlets can provide the presentation functions and EJBs can provide the business logic and interface with the database server. The **servlet container** communicates with the EJB container with the RMI protocol.

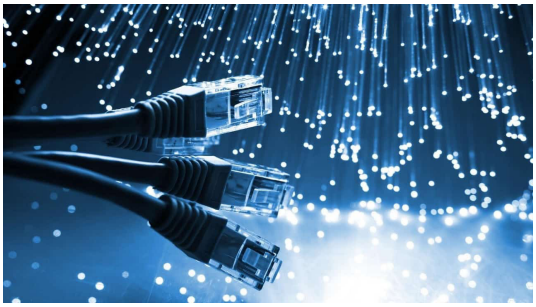
A .NET application can use ASP for presentation purposes and execute the business logic on a DCOM component. The communication between the IIS server and the remote component uses the ORPC protocol.

Network Traffic Patterns: RPC, RMI, ORPC, IIOP

RPCs do not qualify as remote-object technology, yet they provide a simple reference to understanding the network characteristics of such technologies as RMI, ORPC, and IIOP.

An RPC is a procedure invoked from one computer and executed on a different computer. The client computer calls a procedure on the remote computer, on which the procedure waits for requests; once it receives one, it executes the call and returns the results to the client.

The sequence assumes that the client already knows which server can execute the RPC and which protocol and port it is listening to. A mechanism called binding takes care of finding this information.



When the client needs to execute a remote call, it sends a query to a well-known port used by RPC, TCP port 111 (the software service listening on this port is called the portmapper). The server receives the query with the ID of the procedure that the client needs to invoke. The client also specifies the transport protocol, either TCP or User Datagram Protocol (UDP). The server responds with the port number to which the client should connect.

The presence of a dynamically negotiated port can be a challenge when RPC goes through a firewall unless the firewall explicitly supports the RPC protocol. The reason is that when the client and servers try to connect on the dynamically negotiated port, the firewall, not being configured to allow that port, drops the connection. An RPC-aware firewall instead can spoof the negotiation and open the port that was negotiated. Cisco IOS Context Based Access Control (CBAC), Cisco PIX Firewalls, and the Firewall Services Modules (FWSM) have support for RPC.

RMI is a set of APIs that allow applets or servlets to call remote methods. In RMI, the client-side machine sends a request to TCP port 1099 (the software service listening on this port is called the registry). After contacting the server, the client receives the information about the remote object, which includes the hostname and port number of the computer to which it must connect.

Example 4 shows the packet that the server sends to the client, when the server IP address is 10.20.20.10 and the port communicated to the client is 0x08ae, which is 2222 in decimal notation. The use of dynamically negotiated ports is typically a problem with firewalls as described for RPC, unless the firewall is RMI aware.

Example 4 also highlights the presence of an embedded IP address. This embedded address can be a problem if the server IP address is subject to NAT. A firewall between the client-side machine and the server might not have the capability to translate embedded IP addresses for this specific protocol.

Example 4: An RMI Packet

```
Java RMI
    Input Stream Message: ReturnData
    Serialization Data
Java Serialization
    Magic: 0xaced
    Version: 5
[...]
00a0  2e 72 6d 69 2e 73 65 72 76 65 72 2e 52 65 6d 6f   .rmi.server.Remo
00b0  74 65 4f 62 6a 65 63 74 d3 61 b4 91 0c 61 33 1e   teObject.a...a3.
00c0  03 00 00 70 78 70 77 34 00 0a 55 6e 69 63 61 73   ...pxpw4..Unicas
00d0  74 52 65 66 00 0b 31 30 2e 32 30 2e 32 30 2e 31   tRef..10.20.20.1
00e0  30 00 00 08 ae 00 00 00 00 00 00 00 00 19 29   0.............)
```

DCOM is a library that allows communication among components in the network. With DCOM, a client computer can invoke methods on a remote object. DCOM uses ORPC as the wire format.

In DCOM, a client contacts the server on the TCP or UDP port 153 and obtains the reference to a dynamic port for invoking the method for the remote object.

Just like RPC and RMI, using DCOM through a firewall can be difficult because of dynamically negotiated ports. The section, "Using RMI and DCOM Through a Firewall" provides additional details on how to pass these protocols through a firewall.

CORBA can locate remote objects by means of the ORB, which is software that resides in each server and client in the network. The transport protocol used in the context of CORBA is IIOP. With IIOP, a client uses TCP port 683 to connect to the naming service (the equivalent of the registry in RMI) on the server. The server provides the client with the port number where the object is listening for incoming requests. This process can present a problem with firewalls unless the firewall can understand IIOP. Like RPC, RMI, and DCOM, IIOP might not work through a firewall in presence of NAT.

DATABASE ACCESS

Databases provide an organized collection of data. Application servers provide processing and storing of data from and into database systems.

"The component structure cannot be designed from the top down. It is not one of the first things about the system that is designed, but rather evolves as the system grows and changes."

—Robert C. Martin

The database servers are typically separated from the application servers, or from the PCs that have direct access, by firewalls. This separation requires configuring the firewall to allow Layer 4 protocols and ports that are specific to the installed database system. This section explains where to find the information about the traffic characteristics of databases.



SQL enables you to manage access to the database system by giving you the ability to retrieve specific information or save new information. SQL is an American National Standards Institute (ANSI) and Open Systems Interconnection (OSI) standard in database querying, and it is used by commercial DBMSs, such as Microsoft SQL Server, Oracle, and Sybase, as well as open source DBMSs, such as mSQL or MySQL.

Each DBMS product has different access procedures. A driver installed on the application server provides access to the database. For example, if the application server uses servlets, you would install a JDBC driver provided by the DBMS vendor.

Each programming language accesses databases through a database interface. Good database interfaces make it possible to write applications almost independently of the DBMS they interface to.

Commonly used interfaces are

- JDBC—The interface for Java
- DBI—The interface for Perl
- dbx—The PHP interface
- ODBC—A Microsoft library for applications written in C that is based on SQL
- ActiveX Data Objects (ADO)—Another Microsoft interface to access databases with object-oriented software

Example 5 shows what the JDBC interface looks like.

Example 5: Database Access with Java

```
DriverManager.getConnection("jdbc:mysql://10.10.20.14:3306/ski", "username",
  "password");
```

To find out which ports need to be open on a firewall, you consult the documentation of the database vendor.

The Layer 4 port can also be configurable in the application code, as Example 5 shows.

NETWORK ARCHITECTURE CONSIDERATIONS

Application architectures dictate several aspects of the network design. The section, "Integration of Applications" already outlined some of the design implications of integrating applications and how network devices can help. This section provides additional network design considerations that are driven by application requirements. The two main areas of interest are high availability and security.

"Just because architecture is supposed to be stable, it does not mean that it should never change."

—Sinclair Lewis

High availability is provided by clustering, which means executing the same application on multiple servers while providing a single view of the system to the user. How you implement clustering depends on the application.



In some cases, you implement clustering by performing server load balancing either in hardware or in software, we call this approach simply load balancing. Clustering also refers to the combination of load balancing traffic to a number of servers and the capability of these servers to share resources like a storage disk subsystem; in this section, we call this approach clustering.

Security refers to the need to segregate server farms by using firewalls and at the same time to open ports on the firewalls to let the legitimate server-to-server traffic pass. You provide this security by enabling application-specific fixups or by configuring ACLs.

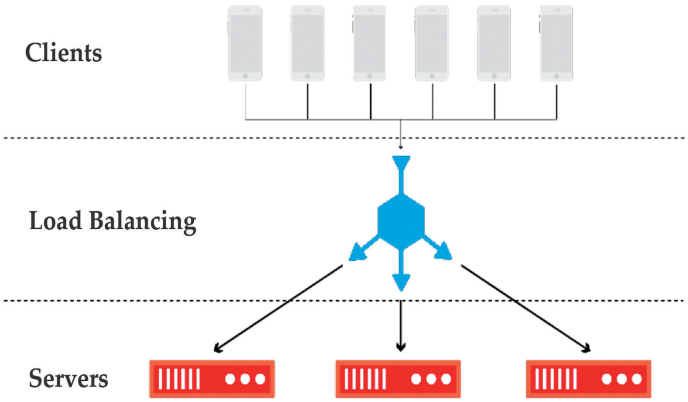
Load Balancing

Server load balancing is balancing traffic load across a group of servers. The load balancer is in charge of distributing the load and ensuring the recipients of the load are available.

Load balancing can be performed by software running on the servers or by hardware devices that are in the path between the client and the server farm.

It is out of the scope of this book to describe in detail how software load balancing works, but it is important to understand that software load balancing is subject to scalability limitations.

Hardware load balancing is a more efficient mechanism for load distribution and high availability because it forwards each client request to only one server in the server farm, thus preserving the LAN bandwidth and optimizing the server-farm performance.



Load balancing is typically applied to these server categories: web servers, application servers, e-mail servers, DNS servers, RADIUS servers, streaming servers, and TN3270 servers.

You can configure load balancers to load-balance almost every protocol used by the applications, but they also have additional intelligence for certain protocols. HTTP is the protocol that load balancers understand better. In addition to HTTP, Cisco load balancers provide special support for SSL, DNS, FTP, POP2, POP3, SMTP, IMAP, streaming protocols, RADIUS, and the Wireless Session Protocol (WSP).

A load balancer can operate on an unlimited number of protocols. You can configure load balancing for a given application if you know the following:

- *The Layer 4 protocols and ports used by the application*—(Often referred to as the port mappings.) If the port mappings are well known, you can configure a load balancer to intercept only the protocol/port pairs of interest.
- *Whether or not the application negotiates dynamic ports*—If the application negotiates Layer 4 ports dynamically, these ports cannot be specifically configured on the load balancer. The solution to this problem is to use a wildcard Layer 4 port.
- *Whether or not the application uses embedded IP addresses*—Load balancers perform NAT, which means that they hide the real address of the servers (which could be a private IP address) by translating it to a different IP address (which could be a public IP address). This mode of

operation is referred to as directed mode. If the protocol used by the application carries IP addresses embedded in the payload of TCP or UDP, the load balancer might not be capable of translating these embedded IP addresses. The solution is to use the load balancer in dispatch mode. Protocols such as FTP or the streaming protocols often do not require dispatch mode because they are automatically translated by the load balancer.

All the protocol/port mappings (including mappings with wildcards) of a given application need to be grouped for the purpose of session persistence.

CLUSTERING

Clustering indicates the use of multiple servers that operate as a single device. Clustering provides high availability and load distribution. Business-critical applications are always deployed on clustered servers.

"Evolving a widely reused resource also requires coordination because changes must be compatible with all existing systems or users. Such coordination can slow down innovation... Some digital companies have even begun to explicitly favor duplication because their business environment rewards economies of speed."

— Gregor Hohpe

For some applications, such as database servers, e-mail servers, and file servers, clustering requires sharing the storage system so that when one element in the cluster fails, the remaining servers have access to the data that was previously processed by the failed device.

Figure 4 shows the topology of a cluster of two servers.

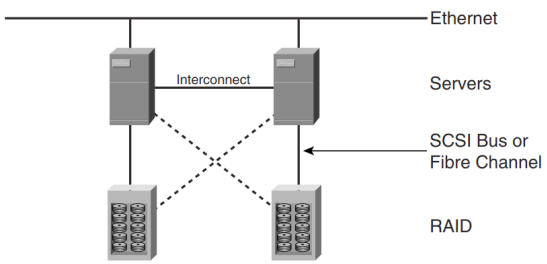


Figure 4: A Two-Node Cluster.

The connectivity between the servers and the storage systems is provided by the I/O channel, which can be a Small Computer System Interface (SCSI) bus, Fibre Channel, Enterprise System Connection (ESCON), or Fibre Connectivity (FICON).

As shown in Figure 4, clustered topologies are characterized by the following:

- A connection to the Ethernet network on a LAN where the servers receive client requests.
- An interconnect cable (or network) that connects the servers directly. The servers use this connection to monitor each other and sometimes to transfer data as well. In most cases, the interconnect can be an Ethernet crossover cable or an Ethernet switch.
- A storage system such as a Redundant Array of Independent Disks (RAID) connected to the servers in the cluster via a SCSI bus or a Fibre Channel network.

Cluster Models

If you want to configure servers to form a cluster, the server needs to run software that implements the clustering functions. Examples are Veritas Cluster Server, HP TruCluster Server, Microsoft Windows NT/2000 Cluster Service (MSCS), Solaris Sun Cluster, HP OpenVMS, and IBM Parallel Sysplex.

The clustering software can support these clustering options:

- *Active/standby two-node cluster*—This cluster consists of two servers attached to a RAID. One server is active and processes the clients' requests. Data is continuously mirrored from the disks assigned to the primary server to the disks assigned to the backup server. When the primary server fails, the backup server takes over with the same IP address and provides access to the same data that was available on the primary server.
- *Active/active shared nothing*—This architecture consists of a pair of servers, both attached to RAID systems of disks. Even if the disks are physically connected to both servers, each array is owned by one server. Figure 4 shows this type of topology. If a server fails, the other server takes over the operations of the failed server and

uses the extra storage connection to mount the volumes previously controlled by the failed server.

- *Active/active shared everything*—This architecture consists of servers connected to the same storage system where the servers can access the same files. A locking system provided by the cluster software prevents concurrent access to a given file. Active/ active shared everything clusters are typically used to build clusters with multiple servers. Figure 5 shows a cluster of servers connected with Fibre Channel to a shared storage system.

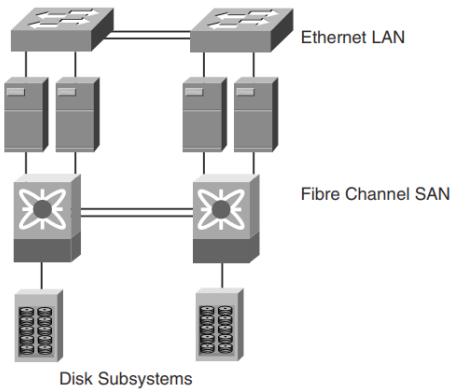


Figure 5: Fibre Channel Storage Cluster.

Key network considerations for integrating clusters into the Data Center are the following:

- The traffic exchanged among the servers in the cluster for monitoring purposes can be multicast or unicast. Unicast is preferable because it preserves the LAN bandwidth.
- Active/active shared everything topologies need to replace the interconnect cable with one or more switches in general with a LAN segment.
- Active/active shared everything topologies call for the use of a SAN to connect servers to storage devices, as shown in Figure 5.

Geographical Clustering

Geographical clustering consists of clustering servers that are located at different geographical facilities.

The number-one requirement of geographical clustering is that the data available in one Data Center must be replicated at the remote Data Center. The second requirement is that the network must be capable of forwarding client requests to the servers in either Data Center location.

The data replication for cluster environments can take advantage of technologies such as disk replication or host-based replication:

- *Disk replication*—This technique replicates data at the block level. When the servers of the cluster in the main Data Center write to the disk array, the disk-replication software generates the I/O write to a disk array in the remote Data Center. With this mechanism, the servers are unaware that the replication is taking place because the operation is performed by the disk-management software. Examples of this type of software are EMC Symmetrix Data Remote Facility (SRDF), IBM Peer-to-Peer Remote Copy (PPRC), HP Data Replication Manager (DRM), and Hitachi Truecopy.
- *Host-based replication*—This technique replicates data at the file-system level. Between the file system and the disk driver is the volume manager, which is software that manages the disks and creates a logical view of the physical devices for the use of the file system. With host replication, the volume manager on the server duplicates the writes to the storage system. Examples of this software are Microsoft Windows Local Disk Manager (LDM) and Veritas Volume Manager.

Designing a cluster solution requires validating which type of technology is supported by the clustering software and whether the replication technology is synchronous or asynchronous. Synchronous replication means that the server which is performing the write to disk waits for an acknowledgment of the success of the write. Asynchronous replication means that the server does not have to wait for an acknowledgment.

Synchronous technologies are suitable for low-latency networks such as a LAN or when Data Centers are no more than 100 kilometers (km) away. In all other cases, it is preferable to use asynchronous replication. The software typically provides the option to choose either mechanism.

The network design for data replication needs to consider the use of a SAN for the connectivity of the servers to the storage system and the need to connect Data Centers either via the WAN or with metro-optical technologies such as dense wave division multiplexing (DWDM) and coarse wave division multiplexing (CWDM). You can extend the SAN geographically with the following technologies:

- *iSCSI*—Internet SCSI consists of sending SCSI commands on IP. If you install an iSCSI driver on the server, the volume manager can send SCSI commands to the Ethernet network interface card (NIC). You can use iSCSI for host-based replication to send write commands to a remote storage system through the WAN.
- *Fibre Channel over IP (FCIP)*—This technique consists of tunneling Fibre Channel through the IP network by using routers with a Fibre Channel Port Adapter or a Fibre Channel switch/director with Gigabit Ethernet ports and FCIP capability (such as the Cisco MDS9000 family).
- *SAN extension via CWDM or DWDM*—This technique consists in using CWDM or DWDM to provide Layer 1 connectivity for the SAN between two Data Centers. With DWDM, it is possible to carry Fibre Channel, ESCON, FICON, and Gigabit Ethernet between Data Centers on the same fiber. Figure 6 depicts the use of DWDM equipment to multiplex the storage transport protocols into one fiber and demultiplex them at the remote Data Center.

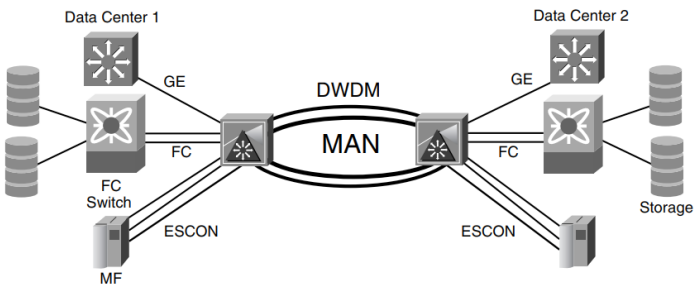


Figure 6: Using DWDM to Connect Distributed Data Centers.

You need to combine the data-replication technologies with a clustering technology that provides a single image of the system of servers. Depending on the clustering software, it might be a

requirement that the servers of either Data Center location be on the same LAN. Servers that belong to the same cluster should be able to exchange traffic at Layer 2 between Data Centers. You can use metro-optical technologies such as CWDM or DWDM to extend the LAN by carrying Gigabit Ethernet to the remote Data Center, as depicted in Figure 6.

When you choose the transport technology that connects the Data Centers, you should consider the data-replication requirements as well as the possible need to extend the local subnet. Pay careful consideration to stretching a local subnet over the WAN because WANs are built using a hierarchical approach to the IP addressing scheme and the interior gateway protocols (IGPs).

It is out of the scope of this book to describe each transport technology, but as a reference, this list describes the typical options:

- *WAN connectivity*—A typical option for Data Center connectivity over the WAN is to use FCIP over a service provider or enterprise synchronous optical network/synchronous digital hierarchy (SONET/SDH) transport. Replication using an FCIP transport can be synchronous over shorter metropolitan distances with low-latency links but is typically asynchronous over longer distances to avoid impacting application performance from the increased latency.
- *Dark Fiber*—Enterprises can connect Data Centers via dark fiber-enabling transport of Gigabit Ethernet or Fibre Channel or further aggregation through deployment of CWDM, DWDM, Dynamic Packet Transport/Resilient Packet Ring (DPT/RPR), or SONET/SDH.
- *Wavelength services*—In some areas, enterprises can connect Data Centers by leasing wavelength services from a service provider. From a technology standpoint, this choice equates to using a channel from a DWDM network. It enables Gigabit Ethernet, RPR, Fibre Channel, or SONET/SDH connectivity between Data Centers

When designing the Data Center connectivity for high availability, one of the key considerations is the distance between the Data Centers. When you choose the distance, consider

3G E-LEARNING

whether both Data Centers are within the threat radius of a potential disaster.

The achievable distance depends on the Layer 2 transport technology itself. It also depends on the latency that can be tolerated by the applications that perform the data replication and the applications that perform the disk writes.

Security

The security technologies that are typically used in a Data Center. The deployment of these technologies varies based on whether they are applied at the Internet Edge or in the intranet server farm

This list summarizes the aspects of the security design that relate to the architecture of the application environment of the Data Center:

- *Security zones*—Server farms are assigned to different virtual LANs (VLANs), and firewalls separate these VLANs and server farms. Servers with similar security requirements are connected on the same LAN and are kept separated from servers that have different requirements. For example, in a multitier application architecture, web servers and database servers are on separate VLANs, and they communicate through a firewall.
- *Port mappings*—Knowing which applications are used in the Data Center lets you decide which protocol/port pairs to open on a firewall. For those applications that open dynamic ports, the best option is to enable fixups on the firewall. A fixup is a feature tailored to a specific application, which can listen to the negotiation of dynamic ports between a client and a server and open these ports for the specific flow. This arrangement limits the exposure of the servers to attacks that can take advantage of services which are not in use but have not been disabled on the servers.
- *Signatures*—IDS sensors can detect attacks of many different types. IDS sensors deployed at the Internet Edge should be configured differently from IDS sensors

deployed in front of the server farm. For the Internet Edge, it is appropriate to identify possible attacks to the client software carried with client-side programming technologies. IDS sensors placed in front of the server farm should use signatures that capture attacks specific to the server and applications in use.

A common challenge when using firewalls in a multitier server farm is protocols that use dynamically negotiated ports and embedded IP addresses. It is easy to use firewalls with these protocols if they are supported as fixups.

Typical fixups available on Cisco firewalls include HTTP, SMTP, Real Time Streaming Protocol (RTSP), FTP, LDAP, Sun RPC, SQL*NET, H.323, Trivial File Transfer Protocol (TFTP), UNIX rlogin, rexec, rsh, and Internet Control Message Protocol (ICMP).

Dynamically negotiated ports and embedded IP addresses are typical of the protocols that are used between web/application servers and distributed components and between web/ application servers and database servers. Some workarounds are necessary to pass these protocols through the firewalls if the firewall does not support them.

Using RMI and DCOM Through a Firewall

When using RMI for remote communication through firewalls, if you open only port 1099 for the connection to the registry, the communication will not work. After contacting the registry for the reference to the object, the actual call is executed on another port. Another possible problem is performing NAT on the firewall because the firewall does not translate the embedded IP address communicated by the registry to the client.

The trace in Example 6 shows a client machine 10.20.20.23 contacting a server machine on port 1099 (packet 1). Packet 10 shows the same client machine contacting the server on port 1067 with another TCP connection. 1099 is, of course, the registry port, and 1067 is the port assigned to the specific object being called. This port is dynamically negotiated, and there is no way for the firewall to know that it needs to open port 1067.

Example 6: RMI Trace

```
Source                Destination           Protocol Info
1  10.20.20.23           10.20.20.10           TCP      2375 > 1099 [SYN]
2  10.20.20.10           10.20.20.23           TCP      1099 > 2375 [SYN, ACK]
3  10.20.20.23           10.20.20.10           TCP      2375 > 1099 [ACK]
4  10.20.20.23           10.20.20.10           RMI      JRMI, Version: 2,
  StreamProtocol
5  10.20.20.10           10.20.20.23           RMI      JRMI, ProtocolAck
6  10.20.20.23           10.20.20.10           RMI      Continuation
7  10.20.20.23           10.20.20.10           RMI      JRMI, Call
8  10.20.20.10           10.20.20.23           TCP      1099 > 2375 [ACK]
9  10.20.20.10           10.20.20.23           RMI      JRMI, ReturnData
10 10.20.20.23           10.20.20.10           TCP      2376 > 1067 [SYN]
11 10.20.20.10           10.20.20.23           TCP      1067 > 2376 [SYN, ACK]
12 10.20.20.23           10.20.20.10           TCP      2376 > 1067 [ACK]
13 10.20.20.23           10.20.20.10           RMI      JRMI, Version: 2,
  StreamProtocol
14 10.20.20.10           10.20.20.23           RMI      JRMI, ProtocolAck
15 10.20.20.23           10.20.20.10           RMI      Continuation
16 10.20.20.23           10.20.20.10           RMI      JRMI, Call
```

There are several workarounds to make RMI work through the firewalls, and they are described at http://java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmi-arch.doc11.html and http://java.sun.com/j2se/1.4.1/docs/guide/rmi/faq.html.

Having a firewall in the path between a DCOM client and a DCOM server can be challenging. DCOM uses the well-known port 135, but the method calls are carried on the top of a dynamically negotiated port. Typically firewalls do not understand DCOM, which means that you need to open a range of ports for DCOM communication.

Another problem is the presence of the server IP address embedded in the response to the client: if the firewall is configured for NAT, this IP is not translated because the firewall does not typically read into the DCOM packets.

The solution to the first problem consists of modifying the configuration on the server machine by using the tool DCOMCFG or by changing the registry file to use a specific range of ports. The registry key is at HKEY_LOCAL_MACHINE\Software\Microsoft\ Rpc\Internet.

By doing this step and by configuring the same range of ports on the firewall, you gain better control on the traffic allowed through the firewall.

IDS Signatures

Network IDS sensors make it possible for the network administrator to monitor the serverfarm subnets for traffic that is trying to exploit server vulnerabilities.

The IDS sensors closer to the servers should capture attacks directed to the servers' operating systems, and you should enable only the signatures that apply to the specific application environment. For example, if the business logic is built with CGI, it makes sense to enable signatures like the following:

- WWW count-cgi overflow
- WWW TEST-CGI attack
- WWW CGI valid shell access
- WWW imagemap.cgi attack
- WWW IRIX infosrch.cgi attack

If the application servers in use are based on servlets, you should enable signatures like the following:

- WWW Sun Java Server access
- *.jsp/*.jhtml Java execution
- Java Web Server command exec

If the web servers are Microsoft IIS servers, you should enable the following signatures:

- WWW IIS view source attack
- IIS MDAC RDS buffer overflow
- IIS 5 Translate: f source disclosure
- IIS executable file command exec

The preceding signatures detect attacks against servers. Other signatures detect attacks against clients. The right place to detect these attacks is at the Internet Edge because the enterprise clients connect to the Internet through the firewalls at the Internet Edge. The signatures that you should turn on detect attacks based on client-side programming technologies such as Java applets, ActiveX controls, and JavaScript. Examples of these signatures are

- ActiveX: Active Setup control file upload bug
- Java hostile applet
- Internet Explorer file access bug

CONCLUSION

The network design of a Data Center is driven by the architecture of the application that needs to be supported. Understanding the application environments used today makes it easier to design the network to provide application high availability and security. The two main software architectures used today are Sun J2EE and Microsoft .NET. Their respective client-side technologies include JavaScript, applets, and ActiveX components. Their respective server-side technologies include Java servlets and ASP. Today's applications, whether developed in-house or commercially, are typically based either on the J2EE or the .NET architecture. These architectures let you build multitier applications. Multitier architectures require specific choices in terms of high availability: some server tiers rely on hardware load balancing, and other server tiers use clustering. You can apply hardware load balancing to a multitude of servers. It is typically necessary to perform hardware load balancing for web and application servers, DNS servers, RADIUS servers, and streaming servers. Session persistence is necessary to optimize the servers' performance. You use clustering to achieve high availability in the deployment of SANs. If the cluster is extended geographically, you need to evaluate the requirements of the applications that write the data as well as the software that performs data replication. These requirements are typically expressed in terms of latency and throughput. The requirements decide the transport technology and the distance between redundant Data Centers. The security design of the Data Center consists of placing firewalls between server tiers and of opening ports based on the applications present in the Data Center. RMI, DCOM, and IIOP are typical protocols used within a Data Center. Sometimes, it is difficult to make these protocols work through a firewall; this chapter references documents that explain how to integrate the applications which use these protocols with the firewalls. Another aspect of the security design consists of enabling IDS signatures in the Data Center and at the Internet Edge. The IDS sensors closer to the servers should capture attacks related to the specific operating systems and application architecture of the server farm. If the server farm is based on the .NET architecture, you should enabled signatures for IIS vulnerabilities; if the server farm is based on J2EE software, you should enable signatures for attacks that exploit servlets, JSP, and Java-based technologies in general.

KEY TERMS

Port density
Session persistence
Server-side programming
Directory server
Helper application
Servlet container

DISCUSSION QUESTIONS

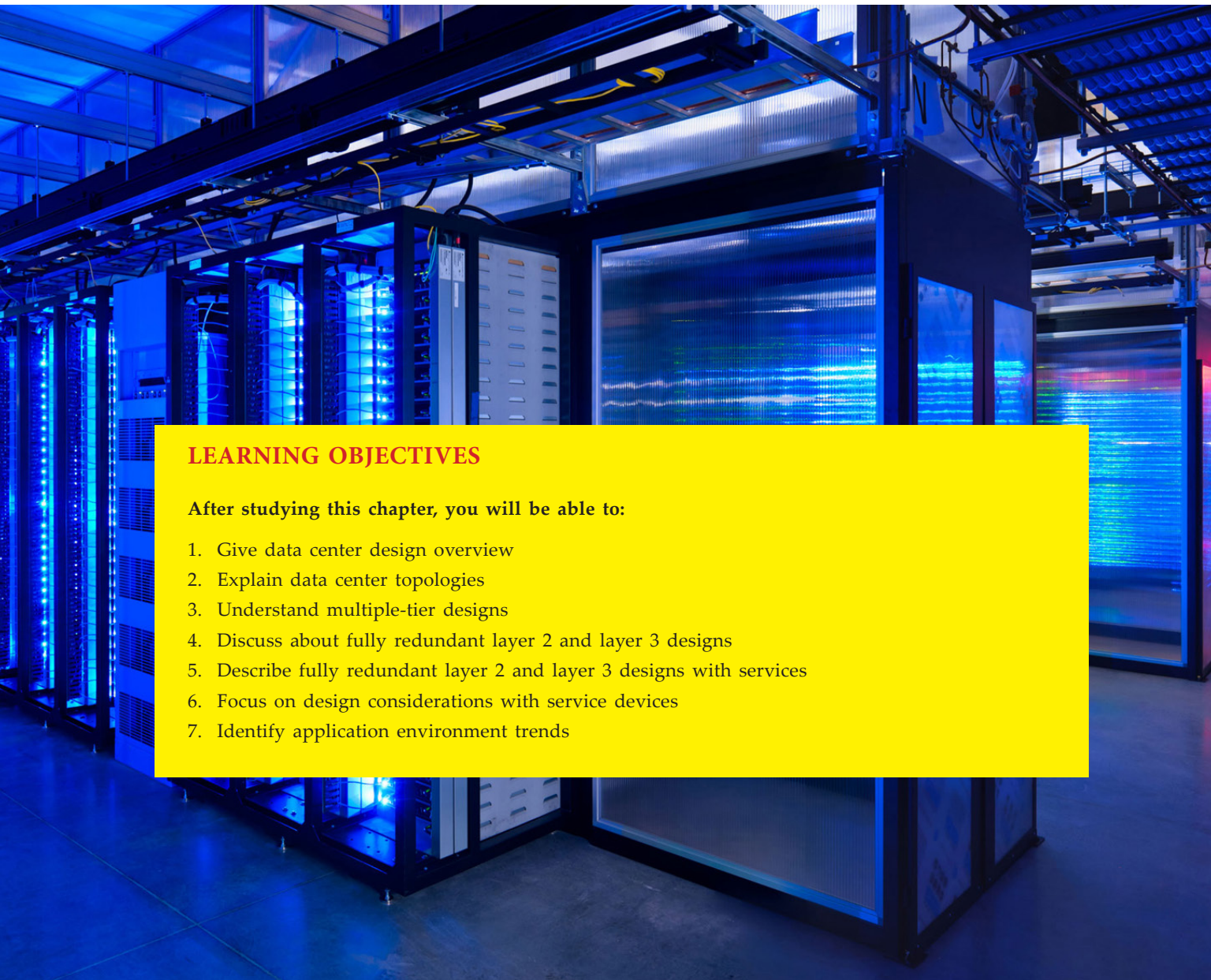
1. Define the applications and hosted servers.
2. Discuss the integration of applications.
3. Explain the markup languages: HTML and XML.
4. What do you understand by web servers?
5. What is middleware? Explain.
6. How to consider network architecture?
7. What is included in application architecture?
8. What is the role of an application architect?
9. What are the three main components of application architecture?
10. What is the difference between software architecture and application architecture?

REFERENCES

1. A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In HotNets, 2011.
2. A. Tootoonchian and Y. Ganjali. Hyperflow: a distributed control plane for openflow. In Proceedings of INM/WREN, 2010.
3. A.Dixit, P. Prakash, Y. C. Hu, R. R. Kompella, "On the impact of packet spraying in data center networks", Proc. INFOCOM, pp. 2130-2138, Apr. 2013.
4. A.Shieh, S. Kandula, A. Greenberg, C. Kim, B. Saha, "Sharing the data center network", Proc. 8th USENIX Conf. Netw. Syst. Design Implement. (NSDI), pp. 23, 2011.
5. C. Demetrescu and G. F. Italiano. A new approach to dynamic all pairs shortest paths. J. ACM, 51(6):968–992, Nov. 2004.
6. C. Guo et al., "SecondNet: A data center network virtualization architecture with bandwidth guarantees", Proc. 6th Int. Conf. Co-NEXT, pp. 15, 2010.
7. C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, M. Handley, "Improving datacenter performance and robustness with multipath TCP", Proc. ACM SIGCOMM, pp. 266-277, 2011.
8. C.-Y. Hong, M. Caesar, P. B. Godfrey, "Finishing flows quickly with preemptive scheduling", ACM SIGCOMM Comput. Commun. Rev., vol. 42, no. 4, pp. 127-138, 2012.
9. Chen et al., "OSA: An optical switching architecture for data center networks with unprecedented flexibility", Proc. NSDI, pp. 18, 2012.
10. Chen et al., "The IBM blue Gene/Q interconnection fabric", IEEE Micro, vol. 32, no. 1, pp. 32-43, Jan./Feb. 2012.
11. Columbus, L. (2017, April 29). Roundup Of Cloud Computing Forecasts, 2017. Retrieved from Forbes: https://www.forbes.com/sites/louiscolumbus/2017/04/29/roundup-of-cloud-computing-forecasts-2017/#41481b5331e8
12. D. Crisan, R. Birke, G. Cressier, C. Minkenberg, M. Gusat, "Got loss? Get zOVN!", Proc. ACM SIGCOMM, pp. 423-434, 2013.
13. D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, "FiConn: Using backup port for server interconnection in data centers", Proc. IEEE INFOCOM, pp. 2276-2285, Apr. 2009.
14. D. Lin, Y. Liu, M. Hamdi, J. Muppala, "FlatNet: Towards a flatter data center network", Proc. IEEE GLOBECOM, pp. 2499-2504, Dec. 2012.

CHAPTER 4

DATA CENTER DESIGN



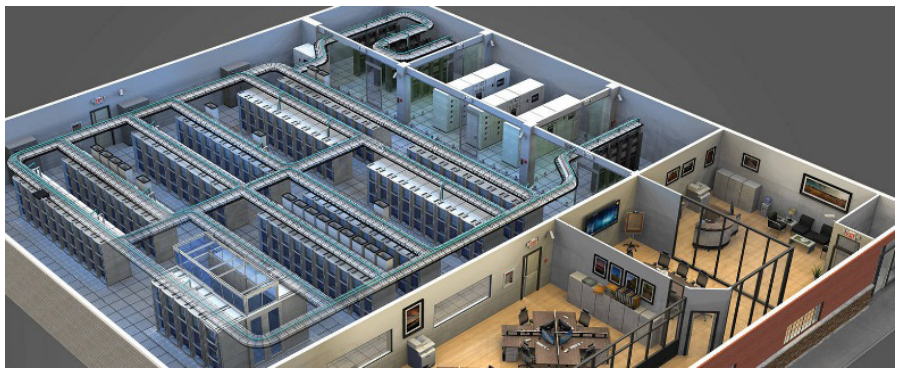
LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Give data center design overview
2. Explain data center topologies
3. Understand multiple-tier designs
4. Discuss about fully redundant layer 2 and layer 3 designs
5. Describe fully redundant layer 2 and layer 3 designs with services
6. Focus on design considerations with service devices
7. Identify application environment trends

INTRODUCTION

The demands on data centers and networks are growing dramatically. Most traditional communications media including telephone, music, film, and television are being reshaped or redefined by the Internet, giving birth to new services such as Voice over Internet Protocol (VoIP) and Internet Protocol Television (IPTV). More and more devices and "things" are being connected to the internet as we evolve towards the Internet of Everything.



With the growing use of streaming video (Netflix, iTunes, YouTube, primetime television, etc.), BYOD (bring your own device), and other connected technologies, the demands on data centers are growing exponentially. Keeping up with the growing demands is becoming a very large problem for the data center manager today. Although important, a good power and cooling strategy doesn't guarantee a good data center design. Even the most efficient power and cooling design doesn't guarantee reliable network performance; a cable could be kinked or unplugged causing costly downtime. A good power and cooling design doesn't simplify the installation and management of the infrastructure. The physical infrastructure can even hinder supplying power where needed and cause the cooling system to be less efficient. Power and cooling strategies won't result in the best optimization of space or have an impact on the whole data center design-build experience. To maximize efficiencies for a data center, five key elements must be considered. This paper will discuss how each of the five elements helps to achieve efficiency throughout the data center life – from design to build to day-to-day operation.

Data Center architectures: scalability, flexibility, and high availability. Data Centers are rapidly evolving to accommodate higher expectations for growth, consolidation, and security. Although the traditional Layer 2 and Layer 3 designs have not changed drastically over the last few years, stringent demands for uptime and service availability, coupled with new technology and protocols, make the design efforts more challenging and demanding.

Demands for scalability, flexibility, and high availability can be summarized as follows:

- *Scalability*—The Data Center must support fast and seamless growth without major disruptions.
- *Flexibility*—The Data Center must support new services without a major overhaul of its infrastructure.
- *High availability*—The Data Center must have no single point of failure and should offer predictable uptime (related to hard failures).

Scalability translates into the capability to sustain rapid growth in performance, the number of devices hosted in the **Data Center**, and the amount and quality of the services offered. Higher performance implies tolerance to very short-term changes in traffic patterns without packet loss and longer-term plans mapping growth trends to the capacity of the Data Center.

Scalability on the number of hosted devices refers to being capable of seamlessly adding more ports for servers, routers, switches, and any other service devices, such as server load balancers, firewalls, IDSs, and SSL offloaders. Higher density also includes slot density because the number of slots ultimately determines the potential growth of the system.

Flexibility translates into designs that accommodate new service offerings without requiring the complete redesign of the architecture or drastic changes outside the normal periods scheduled for maintenance. The approach to flexibility is a modular design in which the characteristics of the modules are known, and the steps to add more modules are simple.

3G E-LEARNING



High availability translates into a fully redundant architecture in which all possible hard failures are predictable and deterministic. This implies that each possible component's failure has a predetermined failover and fallback time, and that the worst-case scenario for a failure condition is still within the acceptable failover limits and is within the requirements as measured from an application availability viewpoint. This means that although the time of failure and recovery of a network component should be predictable and known, the more important time involves the user's perception of the time to recover application service.

Figure 1 presents an overview of the Data Center, which, as a facility, includes a number of the building blocks and components of the larger enterprise network architecture.

The engineering of application environments and their integration to the remaining enterprise network. Different types of server farms support the application environments, yet this book focuses on understanding, designing, deploying, and maintaining the server farms supporting intranet application environments. The actual engineering of the different server farm types—Internet, extranet, and intranet server farms—does not vary much from type to type; however, their integration with the rest of the architecture is different. The design choices that differ for each type of server farm are the result of their main functional purpose. This leads to a specific location for their placement, security considerations, redundancy, **scalability**, and performance. In addition to the server farm concepts, a brief discussion on the types of server farms further clarifies these points.

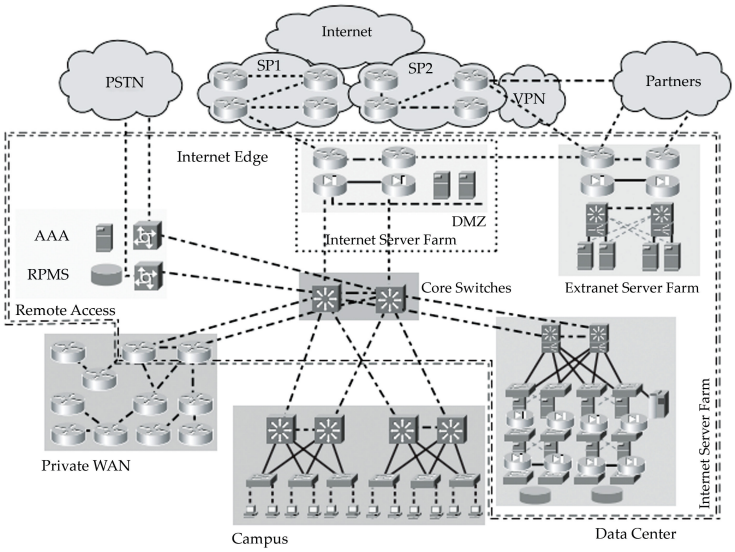


Figure 1: Overview of Data Center Topology.

TYPES OF SERVER FARMS AND DATA CENTERS

As depicted in Figure 1, three distinct types of server farms exist:

- Internet
- Extranet
- Intranet

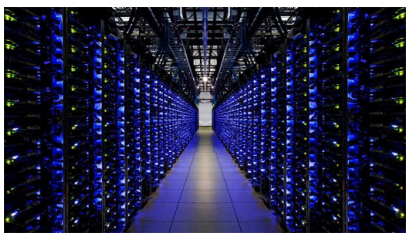
> *"A technology becomes truly disruptive when it drives the marginal cost of something that used to be scarce and expensive to approach zero. Thus, it used to be to deploy software at scale, you had to fund a data center, buy a set of servers, storage, and networking gear, build an in-house IT management capability, and buy an expensive stack of enabling software before you could even get started. Now you can get all that from Amazon or Microsoft on a pay-as-you-grow model."*
>
> **—Geoffrey Moore**

All three types reside in a Data Center and often in the same Data Center facility, which generally is referred to as the corporate Data Center or enterprise Data Center. If the sole purpose of the Data Center is to support Internet-facing applications and server farms, the Data Center is referred to as an Internet Data Center.

Server farms are at the heart of the Data Center. In fact, Data Centers are built to support at least one type of server farm. Although different types of server farms share many architectural requirements, their objectives differ. Thus, the particular set of Data Center requirements depends on which type of server farm must be supported. Each type of server farm has a distinct set of infrastructure, security, and management requirements that must be addressed in the design of the server farm. Although each server farm design and its specific topology might be different, the design guidelines apply equally to them all. The following sections introduce server farms.

Internet Server Farms

As their name indicates, Internet server farms face the Internet. This implies that users accessing the server farms primarily are located somewhere on the Internet and use the Internet to reach the server farm. Internet server farms are then available to the Internet community at large and support business-to-consumer services. Typically, internal users also have access to the Internet server farms. The server farm services and their users rely on the use of web interfaces and web browsers, which makes them pervasive on Internet environments.



Two distinct types of Internet server farms exist. The dedicated Internet server farm, shown in Figure 2, is built to support large-scale Internet-facing applications that support the core business function. Typically, the core business function is based on an Internet presence or Internet commerce.

In general, dedicated Internet server farms exist to sustain the enterprise's e-business goals. Architecturally, these server farms follow the Data Center architecture, "Overview of Data Centers," yet the details of each layer and the necessary layers are determined by the application environment requirements.

Security and scalability are a major concern in this type of server farm. On one hand, most users accessing the server farm are located on the Internet, thereby introducing higher security risks; on the other hand, the number of likely users is very high, which could easily cause scalability problems.

The Data Center that supports this type of server farm is often referred to as an Internet Data Center (IDC). IDCs are built both by enterprises to support their own e-business infrastructure and by service providers selling hosting services, thus allowing enterprises to collocate the e-business infrastructure in the provider's network.

The next type of Internet server farm, shown in Figure 3, is built to support Internet-based applications in addition to Internet access from the enterprise. This means that the infrastructure supporting the server farms also is used to support Internet access from enterprise users. These server farms typically are located in the demilitarized zone (DMZ) because they are part of the enterprise network yet are accessible from the Internet. These server farms are referred to as DMZ server farms, to differentiate them from the dedicated Internet server farms.

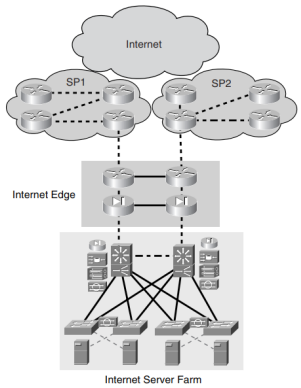


Figure 2: Dedicated Internet Server Farms.

These server farms support services such as e-commerce and are the access door to portals for more generic applications used by both Internet and intranet users. The scalability considerations depend on how large the expected user base is. Security requirements are also very stringent because the security policies are aimed at protecting the server farms from external users while keeping the enterprise's network safe.

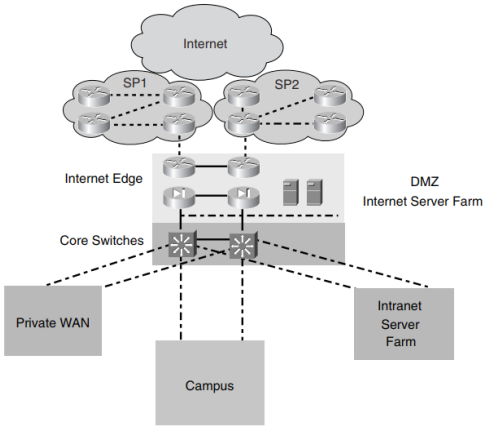
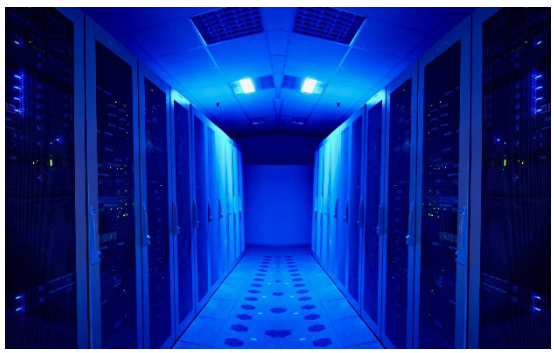


Figure 3: DMZ Server Farms.

Intranet Server Farms

The evolution of the client/server model and the wide adoption of web-based applications on the Internet was the foundation for building intranets. Intranet server farms resemble the Internet server farms in their ease of access, yet they are available only to the enterprise's internal users. As described earlier in this chapter, intranet server farms include most of the enterprise-critical computing resources that support business processes and internal applications. This list of critical resources includes midrange and mainframe systems that support a wide variety of applications. Figure 4 illustrates the intranet server farm.



Notice that the intranet server farm module is connected to the core switches that form a portion of the enterprise backbone and provide connectivity between the private WAN and Internet Edge modules. The users accessing the intranet server farm are

located in the campus and private WAN. Internet users typically are not permitted access to the intranet; however, internal users using the Internet as transport have access to the intranet using virtual private network (VPN) technology.

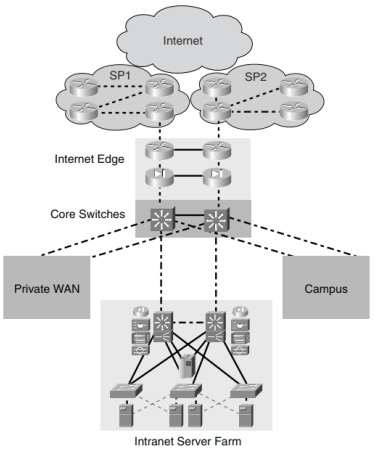


Figure 4: Intranet Server Farms.

The Internet Edge module supports several functions that include the following:

- Securing the enterprise network
- Controlling Internet access from the intranet
- Controlling access to the Internet server farms

The Data Center provides additional security to further protect the data in the intranet server farm. This is accomplished by applying the security policies to the edge of the Data Center as well as to the applicable application tiers when attempting to harden communication between servers on different tiers. The security design applied to each tier depends on the architecture of the applications and the desired security level.

The access requirements of enterprise users dictate the size and architecture of the server farms. The growing number of users, as well as the higher load imposed by rich applications, increases the demand placed on the server farm. This demand forces scalability to become a critical design criterion, along with high availability, security, and management.

Extranet Server Farms

From a functional perspective, extranet server farms sit between Internet and intranet server farms. Extranet server farms continue the trend of using web-based applications, but, unlike Internet- or intranet-based server farms, they are accessed only by a selected group of users that are neither Internet- nor intranet-based. Extranet server farms are mainly available to business partners that are considered external yet trusted users. The main purpose for extranets is to improve business-to-business communication by allowing faster exchange of information in a user-friendly and secure environment. This reduces time to market and the cost of conducting business. The communication between the enterprise and its business partners, traditionally supported by dedicated links, rapidly is being migrated to a VPN infrastructure because of the ease of the setup, lower costs, and the support for concurrent voice, video, and **data traffic** over an IP network.



As explained previously, the concept of extranet is analogous to the IDC, in that the server farm is at the edge of the enterprise network. Because the purpose of the extranet is to provide server farm services to trusted external end users, there are special security considerations. These security considerations imply that the business partners have access to a subset of the business applications but are restricted from accessing the rest of the enterprise network. Figure 5 shows the extranet server farm. Notice that the extranet server farm is accessible to internal users, yet access from the extranet to the intranet is prevented or highly secured. Typically, access from the extranet to the intranet is restricted through the use of firewalls.

Many factors must be considered in the design of the extranet topology, including scalability, availability, and security.

Dedicated firewalls and routers in the extranet are the result of a highly secure and scalable network infrastructure for partner connectivity, yet if there are only a small number of partners to deal with, you can leverage the existing Internet Edge infrastructure. Some partners require direct connectivity or dedicated private links, and others expect secure connections through VPN links. The architecture of the server farm does not change whether you are designing Internet or intranet server farms. The design guidelines apply equally to all types of server farms, yet the specifics of the design are dictated by the application environment requirements.

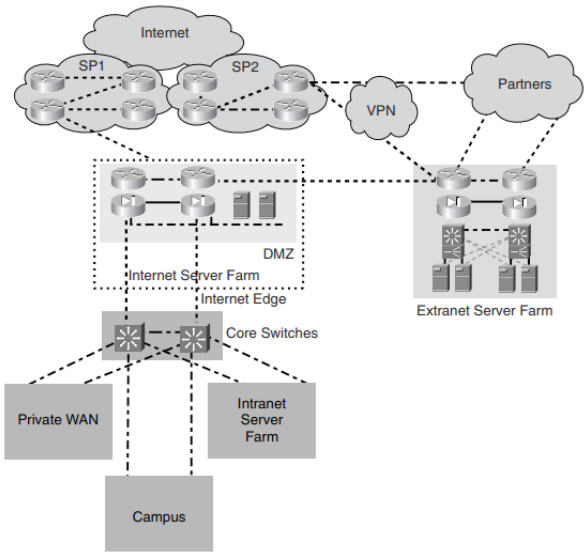


Figure 5: Extranet Server Farms.

Internet Data Center

Internet Data Centers (IDCs) traditionally are built and operated by service providers, yet enterprises whose business model is based on Internet commerce also build and operate IDCs. The architecture of enterprise IDCs is very similar to that of the service provider IDCs, but the requirements for scalability are typically lower because the user base tends to be smaller and there are fewer services compared with those of SP IDCs hosting multiple customers.

In fact, the architecture of the IDC is the same as that presented in Figure 2. An interesting consideration of enterprise IDCs is that if the business model calls for it, the facilities used by the Data Center could be collocated in a service provider Data Center, but it remains under the control of the enterprise. This typically is done to lower the costs associated with building the server farm and reducing a product's time to market by avoiding building a Data Center internally from the ground up.

Corporate Data Center

Corporate or enterprise Data Centers support many different functions that enable various business models based on Internet services, intranet services, or both.

As a result, support for Internet, intranet, and extranet server farms is not uncommon.

This concept was depicted in Figure 1, where the Data Center facility supports every type of server farm and also is connected to the rest of the enterprise network—private WAN, campus, Internet Edge, and so on. The support of intranet server farms is still the primary target of corporate Data Centers.

Enterprise Data Centers are evolving, and this evolution is partly a result of new trends in application environments, such as the n-tier, web services, and grid computing, but it results mainly because of the criticality of the data held in Data Centers.

DATA CENTER TOPOLOGIES

Data Center topologies and, in particular, the server farm topology. Initially, the discussion focuses on the traffic flow through the network infrastructure (on a generic topology) from a logical viewpoint and then from a physical viewpoint.

"[Sovereignty] would break the American monopoly, but it would also break Internet business, because you'd have to have a data center in every country. And data centers are tremendously expensive, a big capital investment."

—Edward Snowden



Generic Layer 3/Layer 2 Designs

The generic Layer 3/Layer 2 designs are based on the most common ways of deploying server farms. Figure 6 depicts a generic server farm topology that supports a number of servers.

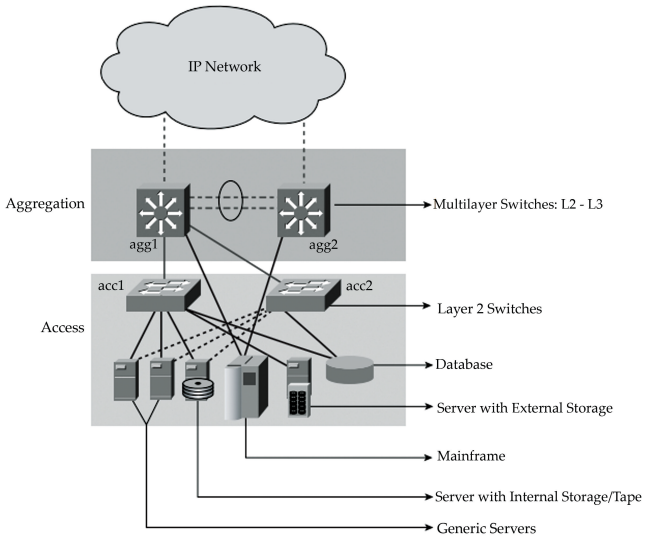


Figure 6: Generic Server Farm Design.

The highlights of the topology are the aggregation-layer switches that perform key Layer 3 and Layer 2 functions, the access-layer switches that provide connectivity to the servers in the server farm, and the connectivity between the aggregation and access layer switches.

The key Layer 3 functions performed by the aggregation switches are as follows:

- Forwarding packets based on Layer 3 information between the server farm and the rest of the network

- Maintaining a "view" of the routed network that is expected to change dynamically as network changes take place
- Supporting default gateways for the server farms

The key Layer 2 functions performed by the aggregation switches are as follows:

- Spanning Tree Protocol (STP) 802.1d between aggregation and access switches to build a loop-free forwarding topology.
- STP enhancements beyond 802.1d that improve the default spanning-tree behavior, such as 802.1s, 802.1w, Uplinkfast, Backbonefast, and Loopguard.
- VLANs for logical separation of server farms.
- Other services, such as multicast and ACLs for services such as QoS, security, rate limiting, broadcast suppression, and so on.

The access-layer switches provide direct connectivity to the server farm. The types of servers in the server farm include generic servers such as DNS, DHCP, FTP, and Telnet; mainframes using SNA over IP or IP; and database servers. Notice that some servers have both internal disks (storage) and tape units, and others have the storage externally connected (typically SCSI).

The connectivity between the two aggregation switches and between aggregation and access switches is as follows:

The connectivity between the two aggregation switches and between aggregation and access switches is as follows:

- EtherChannel between aggregation switches. The channel is in trunk mode, which allows the physical links to support as many VLANs as needed (limited to 4096 VLANs resulting from the 12-bit VLAN ID).
- Single or multiple links (EtherChannel, depending on how much oversubscription is expected in the links) from each access switch to each aggregation switch (uplinks). These links are also trunks, thus allowing multiple VLANs through a single physical path.
- Servers dual-homed to different access switches for redundancy. The NIC used by the server is presumed to have two ports in an active-standby configuration.

When the primary port fails, the standby takes over, utilizing the same MAC and IP addresses that the active port was using.

The typical configuration for the server farm environment just described is presented in Figure 7.

Figure 7 shows the location for the critical services required by the server farm. These services are explicitly configured as follows:

- agg1 is explicitly configured as the STP root.
- agg2 is explicitly configured as the secondary root.
- agg1 is explicitly configured as the primary default gateway.
- agg2 is explicitly configured as the standby or secondary default gateway.

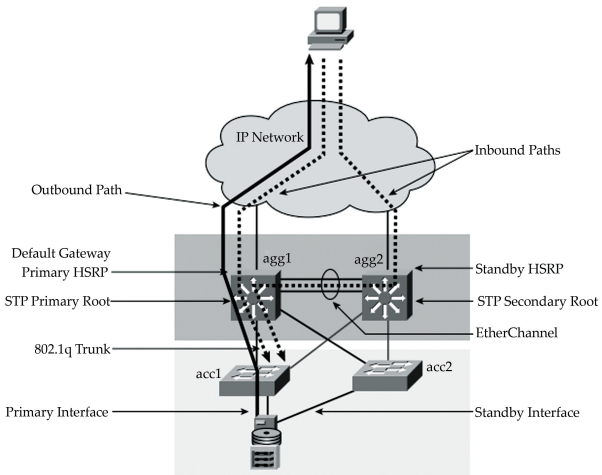


Figure 7: Common Server Farm Environment.

Other STP services or protocols, such as UplinkFast, are also explicitly defined between the aggregation and access layers. These services/protocols are used to lower convergence time during failover conditions from the 802.d standard of roughly 50 seconds to 1 to 3 seconds.

In this topology, the servers are configured to use the agg1 switch as the primary default gateway, which means that outbound traffic from the servers follows the direct path to the agg1 switch. Inbound traffic can arrive at either aggregation

switch, yet the traffic can reach the server farm only through agg1 because the links from agg2 to the access switches are not forwarding (blocking). The inbound paths are represented by the dotted arrows, and the outbound path is represented by the solid arrow.

The next step is to have predictable failover and fallback behavior, which is much simpler when you have deterministic primary and alternate paths. This is achieved by failing every component in the primary path and recording and tuning the failover time to the backup component until the requirements are satisfied. The same process must be done for falling back to the original primary device. This is because the failover and fallback processes are not the same. In certain instances, the fallback can be done manually instead of automatically, to prevent certain undesirable conditions.

The use of STP is the result of a Layer 2 topology, which might have loops that require an automatic mechanism to be detected and avoided. An important question is whether there is a need for Layer 2 in a server farm environment.

The Need for Layer 2 at the Access Layer

Access switches traditionally have been Layer 2 switches. This holds true also for the campus network wiring closet. This discussion is focused strictly on the Data Center because it has distinct and specific requirements, some similar to and some different than those for the wiring closets.

The reason access switches in the Data Center traditionally have been Layer 2 is the result of the following requirements:

- When they share specific properties, servers typically are grouped on the same VLAN. These properties could be as simple as ownership by the same department or performance of the same function (file and print services, FTP, and so on). Some servers that perform the same function might need to communicate with one another, whether as a result of a **clustering** protocol or simply as part of the application function. This communication exchange should be on the same subnet and sometimes is possible only on the same subnet if the clustering

protocol heartbeats or the server-to-server application packets are not routable.

- Servers are typically dual-homed so that each leg connects to a different access switch for redundancy. If the adapter in use has a standby interface that uses the same MAC and IP addresses after a failure, the active and standby interfaces must be on the same VLAN (same default gateway).
- Server farm growth occurs horizontally, which means that new servers are added to the same VLANs or IP subnets where other servers that perform the same functions are located. If the Layer 2 switches hosting the servers run out of ports, the same VLANs or subnets must be supported on a new set of Layer 2 switches. This allows flexibility in growth and prevents having to connect two access switches.
- When using stateful devices that provide services to the server farms, such as load balancers and firewalls, these stateful devices expect to see both the inbound and outbound traffic use the same path. They also need to constantly exchange connection and session state information, which requires Layer 2 adjacency. More details on these requirements are discussed in the section, "Access Layer," which is under the section, "Multiple Tier Designs."

Using just Layer 3 at the access layer would prevent dual-homing, Layer 2 adjacency between servers on different access switches, and Layer 2 adjacency between service devices. Yet if these requirements are not common on your server farm, you could consider a Layer 3 environment in the access layer. Before you decide what is best, it is important that you read the section titled "Fully Redundant Layer 2 and Layer 3 Designs with Services,". New service trends impose a new set of requirements in the architecture that must be considered before deciding which strategy works best for your Data Center.

The reasons for migrating away from a Layer 2 access switch design are motivated by the need to drift away from spanning tree because of the slow convergence time and the operation challenges of running a controlled loopless topology and troubleshooting loops when they occur. Although this is true

when using 802.1d, environments that take advantage of 802.1w combined with Loopguard have the following characteristics: They do not suffer from the same problems, they are as stable as Layer 3 environments, and they support low convergence times.

Alternate Layer 3/Layer 2 Designs

Figure 8 presents an alternate Layer 3/Layer 2 design resulting from the need to address STP limitations.

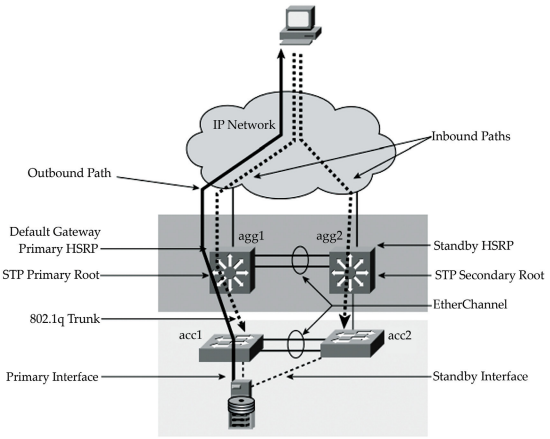


Figure 8: Loopless Topology.

Figure 8 presents a topology in which the network purposely is designed not to have loops. Although STP is running, its limitations do not present a problem. This loopless topology is accomplished by removing or not allowing the VLAN(s), used at the accesslayer switches, through the trunk between the two aggregation switches. This basically prevents a loop in the topology while it supports the requirements behind the need for Layer 2.

In this topology, the servers are configured to use the agg1 switch as the primary default gateway. This means that outbound traffic from the servers connected to acc2 traverses the link between the two access switches. Inbound traffic can use either aggregation switch because both have active (nonblocking) paths to the access switches. The inbound paths are represented by the dotted arrows, and the outbound path is represented by the solid arrows.

MULTIPLE-TIER DESIGNS

Most applications conform to either the client/server model or the n-tier model, which implies most networks, and server farms support these application environments. The tiers supported by the Data Center infrastructure are driven by the specific applications and could be any combination in the spectrum of applications from the client/server to the client/web server/application server/database server. When you identify the communication requirements between tiers, you can determine the needed specific network services. The communication requirements between tiers are typically higher scalability, performance, and security. These could translate to load balancing between tiers for scalability and performance, or SSL between tiers for encrypted transactions, or simply firewalling and intrusion detection between the web and application tier for more security.

"Microsoft is in a court battle with the Department of Justice. The DOJ is saying, "We want information from your data center in Ireland. It's not about a US citizen, but we want it." Microsoft said, "OK, fine. Go to a judge in Ireland. Ask them for a warrant. We have a mutual legal-assistance treaty. They'll do it. Give that to us, and we'll provide the information to you in accordance with Irish laws."

—Edward Snowden

Notice that Figure 9 is a logical diagram that depicts layer-to-layer connectivity through the network infrastructure. This implies that the actual physical topology might be different. The separation between layers simply shows that the different server functions could be physically separated. The physical separation could be a design preference or the result of specific requirements that address communication between tiers.

For example, when dealing with web servers, the most common problem is scaling the web tier to serve many concurrent users. This translates into deploying more web servers that have similar characteristics and the same content so that user requests can be equally fulfilled by any of them. This, in turn, requires the use of a load balancer in front of the server farm that hides the number of servers and virtualizes their services. To the users, the specific service is still supported on a single server, yet the load balancer dynamically picks a server to fulfill the request.

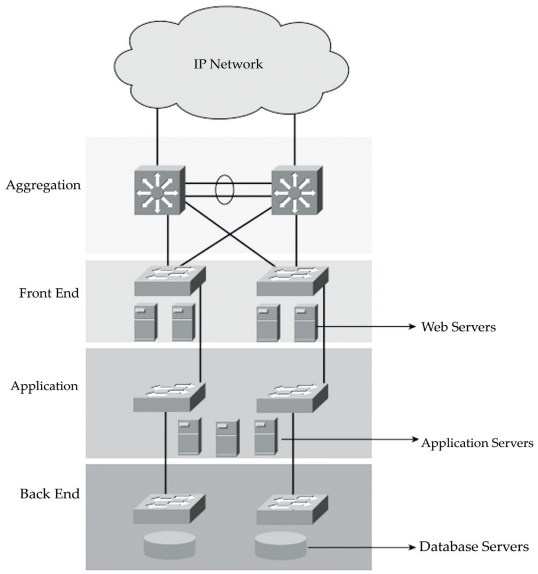


Figure 9: Multiple-Tier Application Environments.

Suppose that you have multiple types of web servers supporting different applications, and some of these applications follow the n-tier model. The server farm could be partitioned along the lines of applications or functions. All web servers, regardless of the application(s) they support, could be part of the same server farm on the same subnet, and the application servers could be part of a separate server farm on a different subnet and different VLAN.

Following the same logic used to scale the web tier, a load balancer logically could be placed between the web tier and the application tier to scale the application tier from the web tier perspective. A single web server now has multiple application servers to access.

The same set of arguments holds true for the need for security at the web tier and a separate set of security considerations at the application tier. This implies that firewall and intrusion detection capabilities are distinct at each layer and, therefore, are customized for the requirements of the application and the database tiers. SSL offloading is another example of a function that the server farm infrastructure might support and can be deployed at the web tier, the application tier, and the database tier. However, its use depends upon the application environment using SSL to encrypt client-to-server and server-to-server traffic.

Expanded Multitier Design

The concept of deploying multiple network-based services in the architecture. These services are introduced in Figure 10 through the use of icons that depict the function or service performed by the network device.

The different icons are placed in front of the servers for which they perform the functions. At the aggregation layer, you find the load balancer, firewall, SSL offloader, intrusiondetection system, and cache. These services are available through service modules (line cards that could be inserted into the aggregation switch) or appliances. An important point to consider when dealing with service devices is that they provide scalability and high availability beyond the capacity of the server farm, and that to maintain the basic premise of "no single point of failure," at least two must be deployed. If you have more than one (and considering you are dealing with redundancy of application environments), the failover and fallback processes require special mechanisms to recover the connection context, in addition to the Layer 2 and Layer 3 paths. This simple concept of redundancy at the application layer has profound implications in the network design.

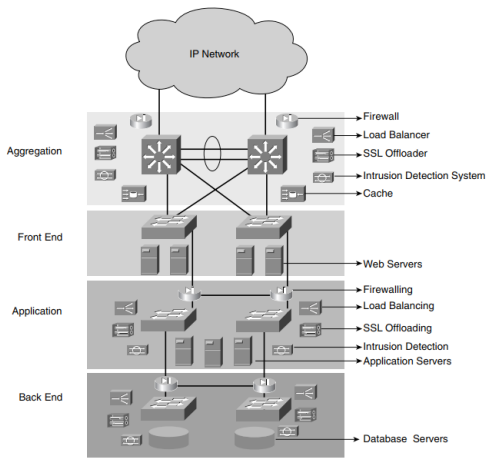


Figure 10: Network Service Icons.

A number of these network service devices are replicated in front of the application layer to provide services to the application servers. Notice in Figure 10 that there is physical separation between the tiers of servers. This separation is one alternative

to the server farm design. Physical separation is used to achieve greater control over the deployment and scalability of services. The expanded design is more costly because it uses more devices, yet it allows for more control and better scalability because the devices in the path handle only a portion of the traffic. For example, placing a firewall between tiers is regarded as a more secure approach because of the physical separation between the Layer 2 switches.

This argument is correct, yet it is likely to be much more related to an existing security policy than a real threat. Having logical instead of physical separation simply requires a consistent application of security policies to ensure that the expanded security zone is as secure logically as it is physically.

This brings the discussion to another alternative of designing the multitier server farm, an alternative in which there is no physical separation, but rather a logical separation between tiers.

Collapsed Multitier Design

A collapsed multitier design is one in which all the server farms are directly connected at the access layer to the aggregation switches, and there is no physical separation between the Layer 2 switches that support the different tiers. Figure 11 presents the collapsed design.

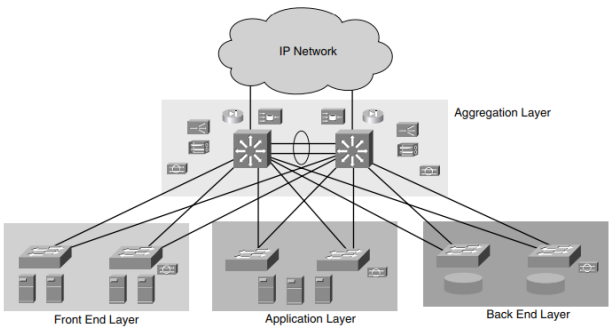


Figure 11: Collapsed Multiple-Tier Design.

Notice that in this design, the services again are concentrated at the aggregation layer, and the service devices now are used by the front-end tier and between tiers. Using a collapsed model, there is no need to have a set of load balancers or SSL

offloaders dedicated to a particular tier. This reduces cost, yet the management of devices is more challenging and the performance demands are higher. The service devices, such as the firewalls, protect all server tiers from outside the Data Center, but also from each other. The **load balancer** also can be used concurrently to load-balance traffic from client to web servers, and traffic from web servers to application servers.

Notice that the design in Figure 11 shows each type of server farm on a different set of switches. Other collapsed designs might combine the same physical Layer 2 switches to house web applications and database servers concurrently. This implies merely that the servers logically are located on different IP subnets and VLANs, yet the service devices still are used concurrently for the front end and between tiers. Notice that the service devices are always in pairs. Pairing avoids the single point of failure throughout the architecture. However, both service devices in the pair communicate with each other, which falls into the discussion of whether you need Layer 2 or Layer 3 at the access layer.

The Need for Layer 2 at the Access Layer

Each pair of service devices must maintain state information about the connections the pair is handling. This requires a mechanism to determine the active device (master) and another mechanism to exchange connection state information on a regular basis. The goal of the dual–service device configuration is to ensure that, upon failure, the redundant device not only can continue service without interruption, but also seamlessly can failover without disrupting the current established connections.

In addition to the requirements brought up earlier about the need for Layer 2, this section discusses in depth the set of requirements related to the service devices:

- Service devices and the server farms that they serve are typically Layer 2–adjacent. This means that the service device has a leg sitting on the same subnet and VLAN used by the servers, which is used to communicate directly with them. Often, in fact, the service devices themselves provide default gateway support for the server farm.

- Service devices must exchange heartbeats as part of their redundancy protocol. The heartbeat packets might or might not be routable; if they are routable, you might not want the exchange to go through unnecessary Layer 3 hops.
- Service devices operating in stateful failover need to exchange connection and session state information. For the most part, this exchange is done over a VLAN common to the two devices. Much like the heartbeat packets, they might or might not be routable.
- If the service devices provide default gateway support for the server farm, they must be adjacent to the servers.

After considering all the requirements for Layer 2 at the access layer, it is important to note that although it is possible to have topologies such as the one presented in Figure 8, which supports Layer 2 in the access layer, the topology depicted in Figure 7 is preferred. Topologies with loops are also supportable if they take advantages of protocols such as 802.1w and features such as Loopguard.

FULLY REDUNDANT LAYER 2 AND LAYER 3 DESIGNS

All the topologies that have been presented are fully redundant. This section explains the various aspects of a redundant and scalable Data Center design by presenting multiple possible design alternatives, highlighting sound practices, and pointing out practices to be avoided.

"Information Age, to the participation age. You get on the Net and you do stuff. You IM (instant message), you blog, you take pictures, you publish, you podcast, you transact, you distance learn, you telemedicine. You are participating on the Internet, not just viewing stuff. We build the infrastructure that goes in the data center that facilitates the participation age. We build that big friggin' Webtone switch. It has security, directory, identity, privacy, storage, compute, the whole Web services stack."

—Scott McNealy

The Need for Redundancy

Figure 12 explains the steps in building a redundant topology.

Figure 12 depicts the logical steps in designing the server farm infrastructure. The process starts with a Layer 3 switch that provides ports for direct server connectivity and routing to the core. A Layer 2 switch could be used, but the Layer 3 switch limits the broadcasts and flooding to and from the server farms. This is option a in Figure 12. The main problem with the design labeled a is that there are multiple single point of failure problems: There is a single NIC and a single switch, and if the NIC or switch fails, the server and applications become unavailable.

The solution is twofold:

- Make the components of the single switch redundant, such as dual power supplies and dual supervisors.
- Add a second switch.

Redundant components make the single switch more tolerant, yet if the switch fails, the server farm is unavailable. Option b shows the next step, in which a redundant Layer 3 switch is added.

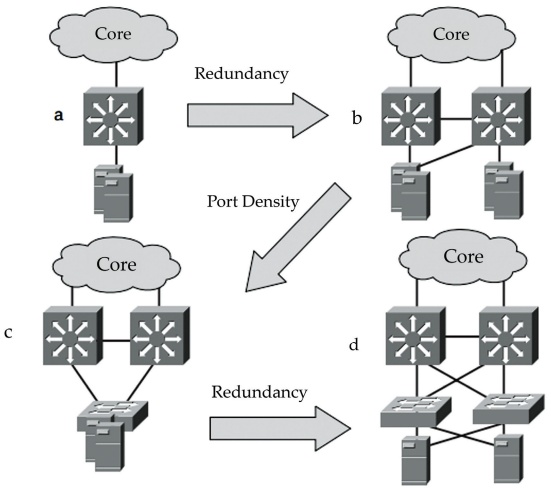


Figure 12: Multilayer Redundant Design.

By having two Layer 3 switches and spreading servers on both of them, you achieve a higher level of redundancy in which the failure of one Layer 3 switch does not completely compromise the application environment. The environment is not completely compromised when the servers are dual-homed, so if one of the Layer 3 switches fails, the servers still can recover by using

the connection to the second switch. In options a and b, the port density is limited to the capacity of the two switches. As the demands for more ports increase for the server and other service devices, and when the maximum capacity has been reached, adding new ports becomes cumbersome, particularly when trying to maintain Layer 2 adjacency between servers.

The mechanism used to grow the server farm is presented in option c. You add Layer 2 access switches to the topology to provide direct server connectivity. Figure 12 depicts the Layer 2 switches connected to both Layer 3 aggregation switches. The two uplinks, one to each aggregation switch, provide redundancy from the access to the aggregation switches, giving the server farm an alternate path to reach the Layer 3 switches.

The design described in option c still has a problem. If the Layer 2 switch fails, the servers lose their only means of communication. The solution is to dual-home servers to two different Layer 2 switches, as depicted in option d of Figure 12.

Layer 2 and Layer 3 in Access Layer

Option d in Figure 12 is detailed in option a of Figure 13.

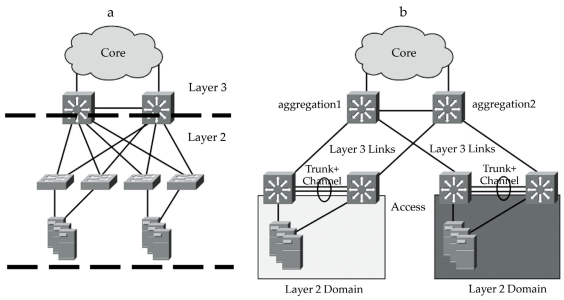


Figure 13: Layer 3 and Layer 2 in the Data Center.

Figure 13 presents the scope of the Layer 2 domain(s) from the servers to the aggregation switches. Redundancy in the Layer 2 domain is achieved mainly by using spanning tree, whereas in Layer 3, redundancy is achieved through the use of routing protocols.

Historically, routing protocols have proven more stable than spanning tree, which makes one question the wisdom of using

Layer 2 instead of Layer 3 at the access layer. This topic was discussed previously in the "Need for Layer 2 at the Access Layer" section. As shown in option b in Figure 13, using Layer 2 at the access layer does not prevent the building of pure Layer 3 designs because of the routing between the access and distribution layer or the supporting Layer 2 between access switches.

The design depicted in option a of Figure 13 is the most generic design that provides redundancy, scalability, and flexibility. Flexibility relates to the fact that the design makes it easy to add service appliances at the aggregation layer with minimal changes to the rest of the design. A simpler design such as that depicted in option b of Figure 13 might better suit the requirements of a small server farm.

Layer 2, Loops, and Spanning Tree

The Layer 2 domains should make you think immediately of loops. Every network designer has experienced Layer 2 loops in the network. When Layer 2 loops occur, packets are replicated an infinite number of times, bringing down the network. Under normal conditions, the Spanning Tree Protocol keeps the logical topology free of loops. Unfortunately, physical failures such as unidirectional links, incorrect wiring, rogue bridging devices, or bugs can cause loops to occur.

Fortunately, the introduction of 802.1w has addressed many of the limitations of the original spanning tree algorithm, and features such as Loopguard fix the issue of malfunctioning transceivers or bugs.

Still, the experience of deploying legacy spanning tree drives network designers to try to design the Layer 2 topology free of loops. In the Data Center, this is sometimes possible. An example of this type of design is depicted in Figure 14. The Layer 2 domain (VLAN) that hosts the subnet 10.0.0.x is not trunked between the two aggregation switches, and neither is 10.0.1.x. Notice that GigE3/1 and GigE3/2 are not bridged together.

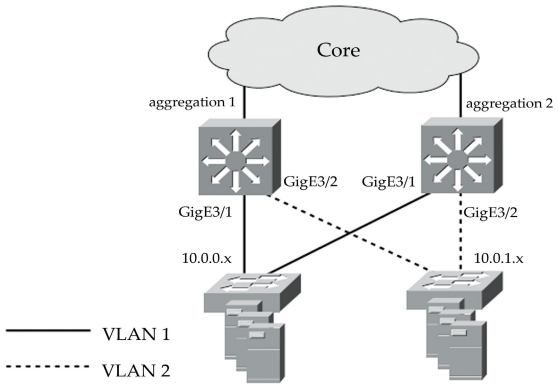


Figure 14: Loop-Free Layer 2 Design.

If the number of ports must increase for any reason (dual-attached servers, more servers, and so forth), you could follow the approach of daisy-chaining Layer 2 switches, as shown in Figure 15.

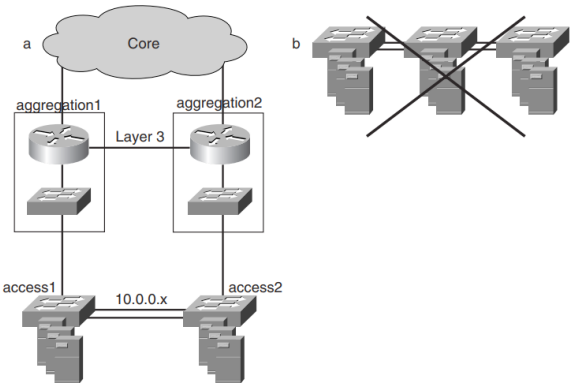


Figure 15: Alternate Loop-Free Layer 2 Design.

To help you visualize a Layer 2 loop-free topology, Figure 15 shows each aggregation switch broken up as a router and a Layer 2 switch.

The problem with topology a is that breaking the links between the two access switches would create a discontinuous subnet—this problem can be fixed with an EtherChannel between the access switches.

The other problem occurs when there are not enough ports for servers. If a number of servers need to be inserted into the same subnet 10.0.0.x, you cannot add a switch between the two

existing servers, as presented in option b of Figure 15. This is because there is no workaround to the failure of the middle switch, which would create a split subnet. This design is not intrinsically wrong, but it is not optimal.

Both the topologies depicted in Figures 14 and 15 should migrate to a looped topology as soon as you have any of the following requirements:

- An increase in the number of servers on a given subnet
- Dual-attached NIC cards
- The spread of existing servers for a given subnet on a number of different access switches
- The insertion of stateful network service devices (such as load balancers) that operate in active/standby mode

Options a and b in Figure 16 show how introducing additional access switches on the existing subnet creates "looped topologies." In both a and b, GigE3/1 and GigE3/2 are bridged together.

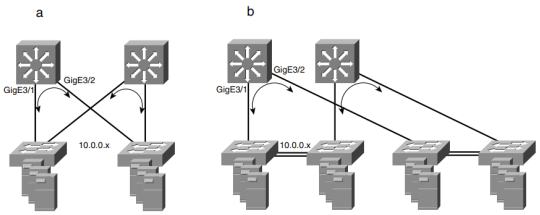


Figure 16: Redundant Topologies with Physical Layer 2 Loops.

If the requirement is to implement a topology that brings Layer 3 to the access layer, the topology that addresses the requirements of dual-attached servers is pictured in Figure 17.

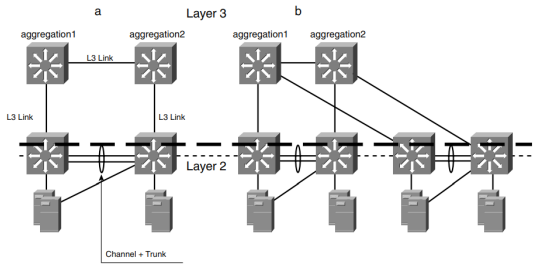


Figure 17: Redundant Topology with Layer 3 to the Access Switches.

Notice in option a of Figure 17, almost all the links are Layer 3 links, whereas the access switches have a trunk (on a channel) to provide the same subnet on two different switches. This trunk also carries a Layer 3 VLAN, which basically is used merely to make the two switches neighbors from a routing point of view. The dashed line in Figure 17 shows the scope of the Layer 2 domain.

Option b in Figure 17 shows how to grow the size of the server farm with this type of design. Notice that when deploying pairs of access switches, each pair has a set of subnets disjointed from the subnets of any other pair. For example, one pair of access switches hosts subnets 10.0.1.x and 10.0.2.x; the other pair cannot host the same subnets simply because it connects to the aggregation layer with Layer 3 links.

So far, the discussions have centered on redundant Layer 2 and Layer 3 designs. The Layer 3 switch provides the default gateway for the server farms in all the topologies introduced thus far. Default gateway support, however, could also be provided by other service devices, such as load balancers and firewalls. The next section explores the alternatives.

FULLY REDUNDANT LAYER 2 AND LAYER 3 DESIGNS WITH SERVICES

The build-out of a fully redundant Layer 2 and Layer 3 topology and considering the foundation of the Data Center, the focus becomes the design issues related to other Data Center services. These services are aimed at improving security and scaling the performance of application services by offloading processing away from the server farm to the network. These services include security, load balancing, SSL offloading, and caching; they are supported by a number of networking devices that must be integrated into the infrastructure following the design requirements.

"Wikipedia was offline after an overheating problem at one of its data centers. It was pretty bad. For a while there, people had nowhere to go for phony, inaccurate information."

—Jay Leno

Additional Services

At the aggregation layer, in addition to Layer 2 and Layer 3, the Data Center might need to support the following devices:

- Firewalls
- Intrusion Detection Systems (IDSs)
- Load balancers
- SSL offloaders
- Caches

It is important to discuss design issues when supporting some of these devices.

Service devices bring their own requirements that could change certain aspects of the design—for instance, the exchange state or status information, the NAT function that they perform on the source or destination IP addresses that forces them to be in the inbound and outbound path, and so on.

Service devices can be deployed using service modules integrated in the aggregation switches or as appliances connected to the aggregation switches. Both deployments require network connectivity and forethought about the actual traffic path.

Firewalls and load balancers may support the default gateway function on behalf of the server farms. Default gateway support traditionally has been provided by the router, so with two additional alternatives, you need to decide which is the default gateway and in which order traffic is processed through the multiple devices. Firewalls and load balancers are capable of providing stateful failover, which is supported by specific redundancy protocols. The protocols, which are specific to the firewalls or load balancers, must be supported by the design. SSL offloaders are typically used with load balancers and require the same considerations, with one exception: They do not support default gateway services.

IDSs are transparent to the design, which means that they integrate well with any existing design. The main consideration with regard to IDSs is their placement, which depends on selecting the location to analyze traffic and the traffic types to be monitored.

Caches, on the other hand, are deployed in reverse proxy cache mode. The placement of the caches and the mechanism for directing traffic to them impact the Data Center design. The options for traffic redirection are the Web Cache Communication Protocol (WCCP) on the Layer 2 or Layer 3 switches, and load balancers to distribute the load among the cache cluster. In either case, the cache or cache cluster changes the basic traffic path to the server farm when in use.

The following section presents the multiple deployment options.

Service Deployment Options

Two options exist when deploying Data Center services: using service modules integrated into the aggregation switch and using appliances connected to the aggregation switch. Figure 18 shows the two options.

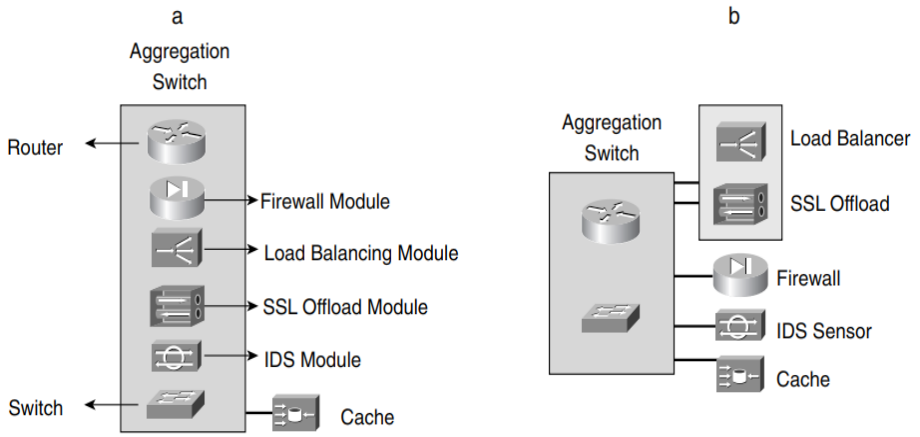


Figure 18: Service Deployment Options.

Option a shows the integrated design. The aggregation switch is represented by a router (Layer 3) and a switch (Layer 2) as the key components of the foundation (shown to the left) and by a firewall, load balancer, SSL module, and IDS module (shown to the right as add-on services). The service modules communicate with the routing and switching components in the chassis through the backplane.

Option b shows the appliance-based design. The aggregation switch provides the routing and switching functions. Other services are provided by appliances that are connected directly to the aggregation switches.

A thoughtful approach to the design issues in selecting the traffic flow across different devices is required whether you are considering option a, option b, or any combination of the options in Figure 18. This means that you should explicitly select the default gateway and the order in which the packets from the client to the server are processed. The designs that use appliances require more care because you must be concerned with physical connectivity issues, interoperability, and the compatibility of protocols.

DESIGN CONSIDERATIONS WITH SERVICE DEVICES

Up to this point, several issues related to integrating service devices in the Data Center design have been mentioned. They are related to whether you run Layer 2 or Layer 3 at the access layer, whether you use appliance or modules, whether they are stateful or stateless, and whether they require you to change the default gateway location away from the router. Changing the default gateway location forces you to determine the order in which the packet needs to be processed through the aggregation switch and service devices.

"I think Unix is a great system - especially for running data centers - because it is very mature, very reliable, very scalable. But when I want to go out and populate small devices, I think Java."

—Bill Joy

Figure 19 presents the possible alternatives for default gateway support using service modules

Figure 19 shows the aggregation switch, a Catalyst 6500 using a firewall service module, and a content-switching module, in addition to the routing and switching functions provided by the Multilayer Switch Feature Card (MSFC) and the Supervisor Module.

The one constant factor in the design is the location of the switch providing server connectivity; it is adjacent to the server farm.

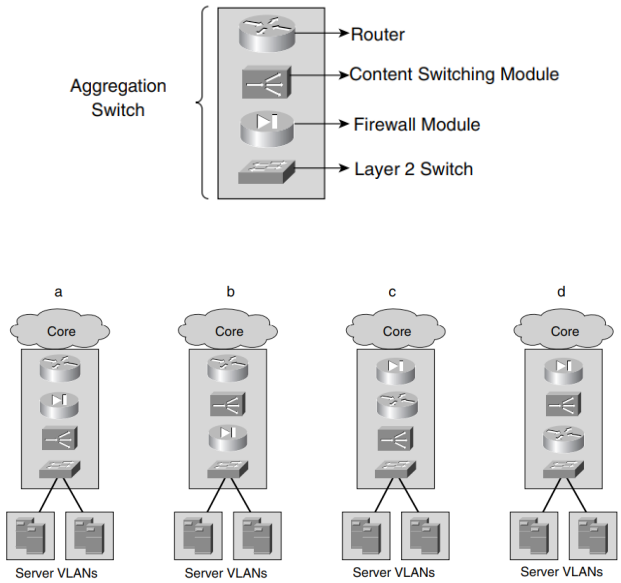


Figure 19: Service Module Interoperability Alternatives.

Option a presents the router facing the core IP network, the content-switching module facing the server farm, and the firewall module between them firewalling all server farms. If the content switch operates as a router (route mode), it becomes the default gateway for the server farm.

However, if it operates as a bridge (bridge mode), the default gateway would be the firewall. This configuration facilitates the creation of multiple instances of the firewall and content switch combination for the segregation and load balancing of each server farm independently.

Option b has the firewall facing the server farm and the content switch between the router and the firewall. Whether operating in router mode or bridge mode, the firewall configuration must enable server health-management (health probes) traffic from the content-switching module to the server farm; this adds management and configuration tasks to the design. Note that, in this design, the firewall provides the default gateway support for the server farm.

Option c shows the firewall facing the core IP network, the content switch facing the server farm, and the firewall module between the router and the content-switching module. Placing a firewall at the edge of the intranet server farms requires the firewall to have "router-like" routing capabilities, to ease the integration with the routed network while segregating all the server farms concurrently. This makes the capability to secure each server farm independently more difficult because the content switch and the router could route packets between the server farm without going through the firewall. Depending on whether the content-switching module operates in router or bridge mode, the default gateway could be the content switch or the router, respectively.

Option d displays the firewall module facing the core IP network, the router facing the server farm, and the content-switching module in between. This option presents some of the same challenges as option c in terms of the firewall supporting IGPs and the inability to segregate each server farm independently. The design, however, has one key advantage: The router is the default gateway for the server farm. Using the router as the default gateway allows the server farms to take advantage of some key protocols, such as HSRP, and features, such as HSRP tracking, QoS, the DHCP relay function, and so on, that are only available on routers.

All the previous design options are possible—some are more flexible, some are more secure, and some are more complex. The choice should be based on knowing the requirements as well as the advantages and restrictions of each. "Integrating Security into the Infrastructure," addresses the network design in the context of firewalls.

APPLICATION ENVIRONMENT TRENDS

Undoubtedly, the most critical trends are those related to how applications are being developed and are expected to work on the network. These trends can be classified arbitrarily into two major areas:

- Application architectures
- Network infrastructure

Application Architecture Trends

Application architecture trends include the evolution of the classic client/server model to the more specialized n-tier model, web services, specific application architectures, the server and client software (operating systems), application clients, the server and client hardware, and middleware used to integrate distributed applications in heterogeneous environments.

The more visible trends of application architectures are the wide adoption of web technology in conjunction with the use of the n-tier model to functionally segment distinct server types. Currently, web, application, and database servers are the basic types, yet they are combined in many ways (depending on the vendor of the application and how the buyer wants to implement it).

This functional partitioning demands that the network be smarter about securing and scaling the tiers independently. For instance, the n-tier model's web tier layer created the need for smaller and faster servers used to scale up the front-end function. This resulted in 1RU (rack unit) servers, which offer adequate performance for web servers at a low cost and minimal infrastructure requirements (power and rack space).

Web services are bringing a service-oriented approach to the use of different and distinct distributed applications that are accessible using standard messages over Internet protocols. Web services rely initially on the transport functions of the network and eventually on using the network as an extension to provide computing capacity to the distributed application environments by offloading tasks to network hardware.

Grid computing is another trend that actually brings the applications and the network closer together by treating the servers as a network of CPUs in which the applications use the most available CPU on the network. Other trends related to grid computing include blade servers as an alternative to 1RU servers, to provide higher CPU density per RU, lower power consumption per server, and an additional benefit of lower cabling requirements. Blade servers are servers on blades (or modules) that are inserted into a chassis, much like network modules or line cards are inserted on a switch chassis. Using blade servers in blade chassis enables you to centralize the

server-management functions (one chassis instead of however many servers are in the chassis), requires less cables (one set per chassis instead of one set per server), and provides higher computing and memory capacity per rack unit.

However, the blade server technology is still young, which explains the variety of flavors, architectures, connectivity options, and features.

An instance of middleware is the software used in the management and control of distributed CPUs in a grid of computers that can be 1RU or blade servers. This specific middleware virtualizes the use of CPUs so that the applications are given a CPU cycle from CPUs on the network instead of through the traditional manner.

Network Infrastructure Trends

The network infrastructure is growing smarter and more application-aware, and it thereby supports application environments both by offloading some computationally intense tasks to the network (typically hardware-based) and by replacing some functions performed by servers that could be better handled by networking devices.

Load balancing is a good example of a function performed by the network that replaces clustering protocols used by servers for high availability. Clustering protocols tend to be software-based, hard to manage, and not very scalable in providing a function that the network performs well using hardware.

Trends such as blade servers bring new design considerations. Most blade server chassis (blade chassis, for short) in the market support both an option to provide redundant Ethernet switches inside the chassis and as an option to connect the blade servers to the network using pass-through links, with the chassis simply providing at least twice as many uplinks as servers in the chassis, to allow dual-homing.

Figure 20 presents both connectivity alternatives for a blade chassis.

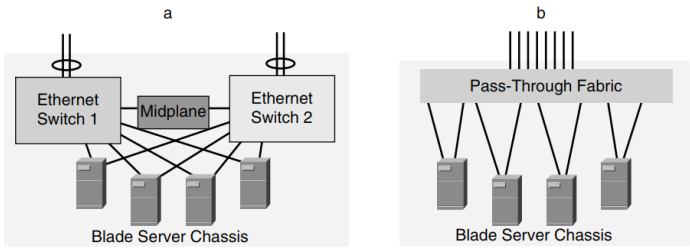


Figure 20: Blade Server Chassis Server Connectivity.

Option a in Figure 20 shows a blade server chassis in which each blade server is connected to each of the blade chassis's redundant Layer 2 Ethernet switches. Each blade chassis's Ethernet switch provides a number of uplinks that can be channeled to the IP network. The number of uplinks is typically smaller than the combined number of links per server, which requires planning for oversubscription, particularly if the servers are Gigabit Ethernet–attached. The midplane is the fabric used for management tasks, that is, control plane traffic such as switch status.

Option b in Figure 20 presents the pass-through option in which the servers are dualhomed and preconnected to a pass-through fabric that provides the connectivity to the IP network. This option does not use Ethernet switches inside the chassis. The pass-through fabric is as simple as a patch panel that conserves the properties of the server NICs, but it also could become a more intelligent fabric, adding new features and allowing blade server vendors to differentiate their products. Either approach you take to connect blade servers to your network requires careful consideration on short- and long-term design implications.

For instance, if the choice is to utilize the redundant Ethernet switches in the blade chassis, you have the following design alternatives to consider:

- How to use the redundant Ethernet switches' uplinks for connectivity
- Whether to connect the blade chassis to the access or aggregation switches
- What level of oversubscription is tolerable

Figure 21 displays two connectivity choices utilizing the uplinks on the redundant Ethernet switches. For redundancy,

two switches are used to connect the uplinks from the blade chassis. Switches A and B, the small clouds in the IP network cloud, provide a redundant network fabric to the blade chassis to avoid single point of failure issues.

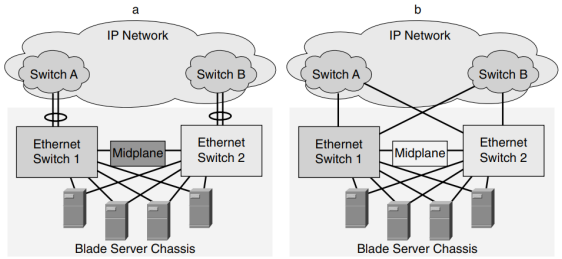


Figure 21: Blade Chassis Uplink Connectivity.

Option a in Figure 21 shows all the uplinks from both blade chassis' Ethernet switches connected to a single switch in the IP network. This allows the uplinks to be channeled. In contrast, option b in Figure 21 shows each blade chassis Ethernet switch connected to each IP network switch, also avoiding a single point of failure. This presents the advantage of having a direct link to either switch A or switch B, thus avoiding unnecessary hops. Additionally, if each blade chassis Ethernet switch supports more than two uplinks, they can also be channeled to switches A and B for greater redundancy and higher bandwidth.

The next step is to determine whether to connect the blade chassis to the access-layer switches, as is traditionally done with servers, or to the aggregation layer switches. Figure 22 displays the connectivity options for the next-hop switches from the blade chassis.

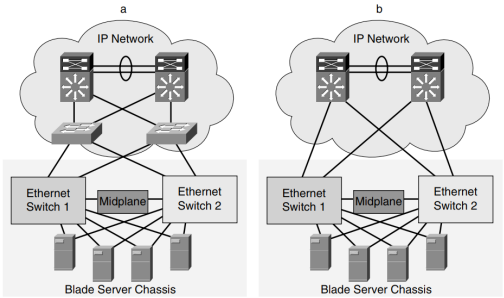


Figure 22: Blade Chassis Next-Hop Switch.

Option a in Figure 22 shows the blade chassis connected to the access-layer switches. This particular design choice is equivalent to connecting Layer 2 access switches to Layer 2 access switches.

The design must take into account spanning tree recommendations, which, based on the topology of option a in Figure 22, are aimed at determining a loop-free topology given the number of Layer 2 switches and the amount of available paths to the STP root and the secondary root from each leaf node. If the blade chassis Ethernet switches support 802.1w, the convergence time stays within two to three seconds; however, if the support is strictly 802.1d, the convergence time goes back to the typical range of 30 to 50 seconds. Other design considerations have to do with whether the midplane is used for more than management and switch-to-switch control traffic communication functions. If for some reason the midplane also is used to bridge VLANs (forward Bridge Protocol Data Units, or BPDUs) the STP topology needs to be considered carefully. The design goals remain making the topology predictable and deterministic.

This implies that you need to explicitly set up root and bridge priorities and analyze the possible failure scenarios to make sure they support the requirements of the applications.

Option b in Figure 22 shows the blade chassis Ethernet switches directly connected to the aggregation switches. This is the preferred alternative because it lends itself to being more deterministic and supporting lower convergence times. Much like in the previous option, if the blade chassis Ethernet switches do not support 802.1w or some of the STP enhancements such as Uplinkfast and Loopguard, the convergence time would be in the range of 30 to 50 seconds.

The topology still needs to be made deterministic and predictable by explicitly setting up root and bridge priorities and testing the failures scenarios.

How to scale the blade server farm is another consideration. Scalability on server environments is done simply by adding pairs of access switches for redundancy and connecting them to the aggregation switches, as shown in option a in Figure 23.

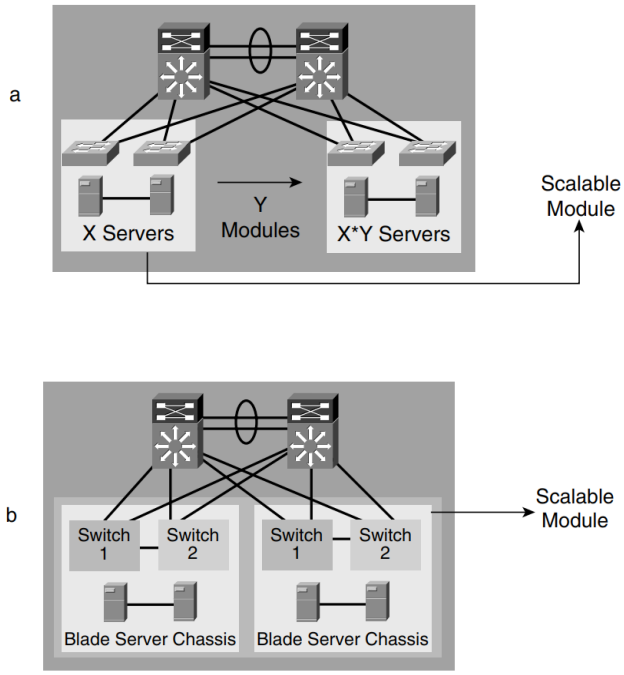


Figure 23: Server Farm Scalability.

If a single scalable server module supports X servers (limited by port density), higher scalability is achieved by replicating the scalable module Y times (limited by slot density in the aggregation switch). The total number of servers could be X * Y. Depending on the access switch port density and the aggregation switch slot density, this could grow to thousands of servers. Scaling the number of blade servers might require a slightly different strategy. Because blade chassis with Ethernet switches are the access layer, the amount of blade server is limited to the number of slots and ports per slot at the aggregation switches. Option b in Figure 23 shows this alternative.

Notice that the scalable module is now the aggregation switch along with a set number of blade chassis. This is because the aggregation switch has a limit to the number of slots that can be used for blade chassis. In addition, line cards used to support blade server uplinks now receive aggregate server traffic, thus requiring less oversubscription. This leads to fewer ports used per line card. So, the total number of blade servers is limited somewhat by the slot and port density. Even though this design alternative is likely to support hundreds of blade servers and satisfy the requirements for a fast-growing server

farm environment, you must have a plan for what to do if you need to increase your server farm beyond what the current design supports. Figure 24 shows this alternative.

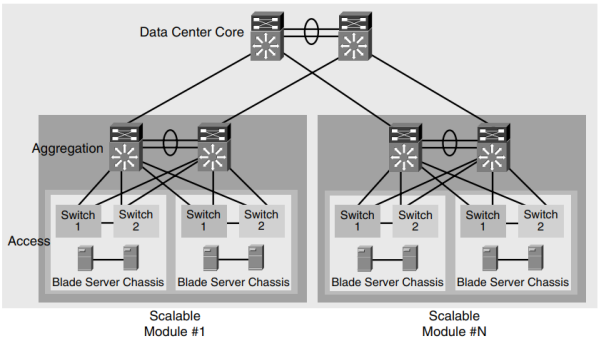


Figure 24: Core Layer Within the Data Center.

Figure 24 introduces a new layer in the Data Center: the core layer. The core layer is used to aggregate as many server blade modules as needed, but the number is limited to the port and slot capacity to the aggregation switches. The pass-through option might not require as much planning because the blade chassis do not have redundant Ethernet switches. The uplinks are connected to the access layer, which is equivalent to current designs in which servers are dual-homed to a redundant set of access switches.

Setting aside the connectivity, port density, slot density, and scalability considerations, other areas, such as oversubscription, uplink capacity, and service deployment options, might require design and testing before the Data Center architecture is established.

Additional trends include the dual-homing of servers, the migration from Fast Ethernet to Gigabit Ethernet, application firewalls, and the use of transparent network service devices. Application firewalls are firewalls that are more in tune with application behavior than ordinary firewalls, thus making the firewalling process more granular to application information in addition to just network or transport layer information. For instance, an application firewall might be capable of identifying not only that a packet is TCP and that the information in the TCP payload is HTTP, but also that the request comes from a

specific high-priority user and is a SQL request for sensitive payroll information, which requires a higher security service level.

Transparent network services include firewalling, load balancing, SSL offloading, and so on. These services are provided by network devices with minimal interoperability issues that leave the existing designs unchanged. These transparent services could apply to traditional network services such as load balancing and firewalling, yet they are implemented to minimize disruption and changes in the application environment. This approach might include using physical devices as if they were distinct logical entities providing services to different server farms concurrently. This implies that the administration of those services, such as configuration changes or troubleshooting efforts, is isolated to the specific logical service. Think of it as a single physical firewall that is deployed to support many server farms concurrently where access to the CLI and configuration commands is available only to users who have been granted access to the specific server farm firewall service. This would appear to the user as a completely separate firewall.

CONCLUSION

Data Centers are very dynamic environments hosting multiple types of server farms that all support key business applications. The design of the Data Center involves a variety of aspects related to how applications are architected, how they are deployed, and their network infrastructure. A sound approach to design involves using a combination of architectural principles, such as scalability, flexibility, and high availability, as well as applying those principles to the requirements of the application environment. The result should be an architecture that meets the current needs but that is flexible enough to evolve to meet the needs of short- and long-term trends.

A solid foundation for Data Center design is based on a redundant, scalable, and flexible Layer 2 and Layer 3 infrastructure in which the behavior is both predictable and deterministic. The infrastructure also should accommodate service devices that perform key functions aimed at scaling or securing application environments. The deployment of service devices

such as firewalls, load balancers, SSL offloaders, and caches requires careful planning. The planning efforts must ensure that the desired behavior is achieved in the following areas: redundancy protocols between service devices, the exchange of connection and session information between stateful devices, the location of default gateway services, and the traffic path through the Data Center infrastructure from device to device. Additional considerations require an architectural approach to deal with the application environment trends and the requirements that are imposed on the network infrastructure.

CASE STUDY

THE WORLD'S SECOND LARGEST TIER IV DATA CENTER

Yotta Infrastructure and Uptime Institute teams collaborated at multiple fronts to mitigate customer risk to the maximum extent.

Overview

Yotta Infrastructure Solutions is a new-age, managed data center provider that caters to the hosting requirements of both wholesale and retail customers. It was conceived to bridge the gap between huge demand and low supply for high-quality, hyperscale data centers in the country. The idea was to build ultra-scalable data center solutions that help companies grow unrestricted and uninterrupted.

The company is driven by its tremendous real estate acumen combined with comprehensive data center services and expertise and unparalleled power capability. It is also a 100% subsidiary of Hiranandani Group, a leading real-estate firm in India with access to large land parcels across the country and world-class construction, power generation, and distribution capabilities. The Yotta Infrastructure team has vast expertise in digital infrastructure design, engineering, execution, and operations. This combination of real-estate expertise and data center execution has enabled Yotta Infrastructure to leverage large economies of scale and build a highly cost-effective proposition while delivering Uptime Institute Tier IV-certified quality to hyperscalers for their wholesale colocation needs and, at the same time, offers quality digital infrastructure to enterprises and governments.

The Yotta Infrastructure Difference

Yotta Infrastructure has always focused on delivering scalability, quality, innovation, and economies of scale ranging from data center infrastructure and operational practices to products and services. And its goal is to be able to deliver better solutions and services than existing players in the country.

"Yotta's vision is to deliver the best quality data center services to its customer at the most reasonable costs. We made this possible via our group capabilities of real-estate ownership, execution acumen, power generation, and distribution capabilities," said Sunil Gupta, co-founder and CEO, Yotta Infrastructure. "Uptime Institute's Tier IV Certification of Constructed Facility further validates our commitment to delivering state-of-the-art IT infrastructure to the nation."

The Uptime Institute Solution

With that in mind, Yotta Infrastructure moved forward working with Uptime Institute on a Tier Certification of Design Documents (TCDD) for the initial pre-construction planning and design of its NM1 facility in Mumbai. Following this, they pursued a Tier certification of Constructed Facility (TCCF) for the data center, which finished in 2021. Yotta NM1 is India's first and only data center to earn Uptime Institute's Tier IV certification and is also the country's largest data center in terms of rack capacity and power offered.

"Any single point of failure in power and cooling systems or various supporting systems (including structural, civil, automation, and others) will not bring down a customer's rack or infrastructure at any point in time. Equally stringent are the criteria that ensure safety if there is an incident of fire at any place within the data center. Full power and cooling will continue to the rack for at least one hour, even during an active fire," said N. K. Jain, chief technical advisor at Yotta Infrastructure.

With this end objective of fault-tolerant design as per global standards, Yotta created a Tier IV facility with just 25% extra CapEx required compared to a Tier III design. Yotta Infrastructure chose unconventional data center designs and went through a highly detailed evaluation from the Uptime Institute team. Achieving Uptime Institute's Tier IV TCDD and TCCF for NM1, while still keeping the space usage and overall costs at an optimized level, required an out-of-the-box design approach. Yotta Infrastructure believes that the efforts have paid off well. In fact, the Yotta NM1 data center won the title of Best APAC Data Center in Datacloud's 2021 Global Awards program.

Yotta Infrastructure and Uptime Institute teams collaborated at multiple fronts to mitigate customer risk to the maximum extent. The first of its kind in the country, the Yotta NM1 data center was a unique project, and the process to complete and certify it combined the extensive domain expertise of both teams and intensive knowledge sharing from start to finish. Uptime Institute's digital infrastructure expertise across geographies and experience undertaking thousands of projects worldwide helped Yotta ensure the innovative project met all necessary requirements.

KEY TERMS

Data center
Scalability
Data traffic
Clustering
Load balancer

DISCUSSION QUESTIONS

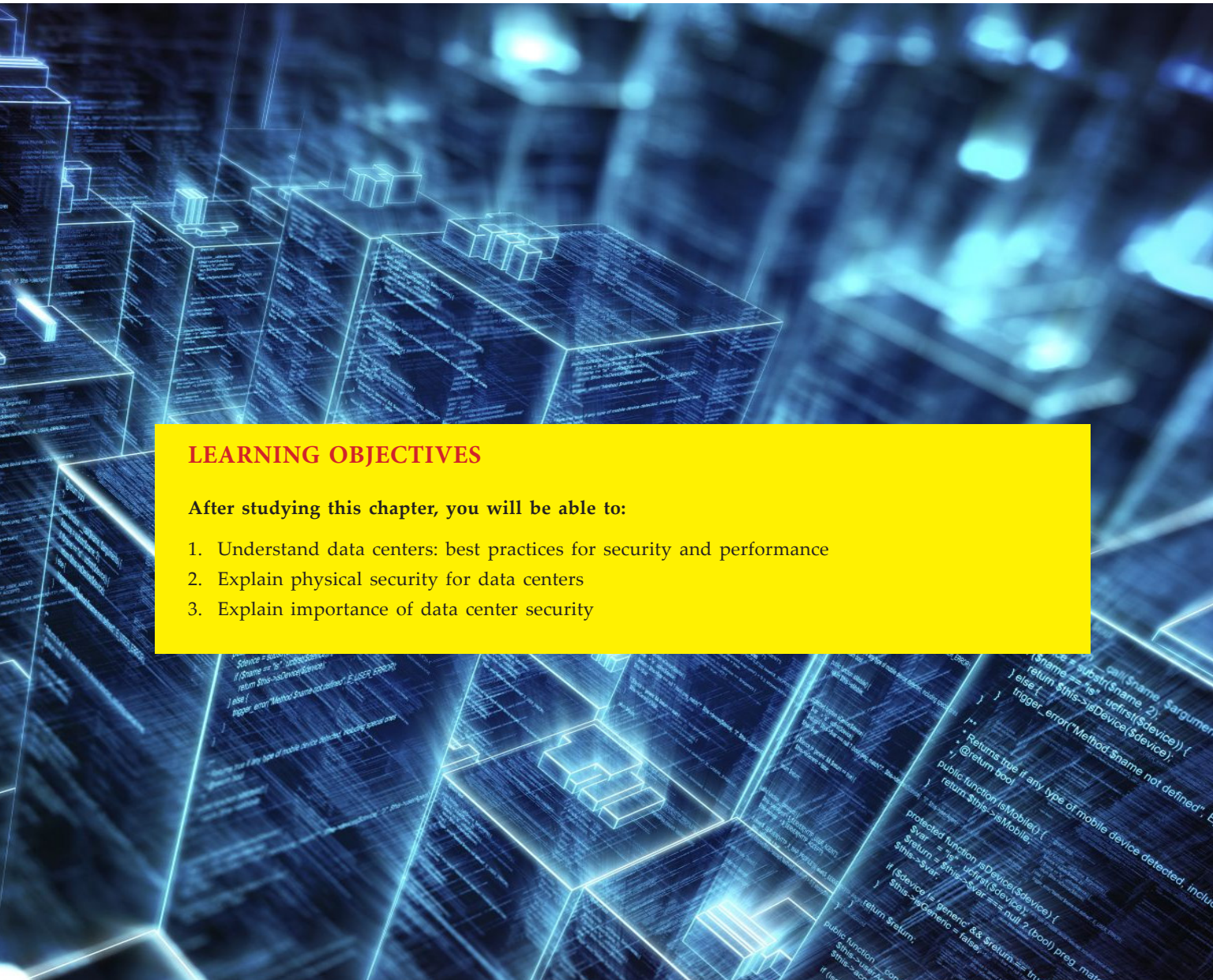
1. What are the types of server farms and data centers?
2. Explain the multiple-tier designs.
3. Discuss about layer 2 and layer 3 designs.
4. Describe the application environment trends.
5. Write short notes on
   - Internet Data Center
   - Corporate Data Center

REFERENCES

1. AVRAMOV L., PORTOLANI M., The Policy Driven Data Center with ACI: Architecture, Concepts, and Methodology, CISCO Press, 2014
2. AZODOLMOLKY S., Software Defined Networking with OpenFlow, Packt Publishing, 2013
3. GENG H., Data Center Handbook, Wiley, 2014
4. GORANSSON P., BLACK C., Software Defined Networks: A Comprehensive Approach, Morgan Kaufmann, 2014
5. HOODA S.K., KAPADIA S., Using TRILL, FabricPath, and VXLAN: Designing Massively Scalable Data Centers (MSDC) with Overlays, CISCO Press, 2014
6. HU F., Network Innovation through OpenFlow and SDN: Principles and Design, CRC Press, 2014
7. LOWE S., MARSHALL N., GUTHRIE F. et al., Mastering VMware vSphere 5.5, SYBEX, 2013
8. MADISETTI V., BAHGA A., Internet of Things (A Hands-onApproach), VPT publisher, 2014
9. MIR N. F., Computer and Communication Networks, Prentice Hall, 2014
10. MORREALE P.A., ANDERSON J.M., Software Defined Networking: Design and Deployment, CRC Press, 2014
11. NADEAU T.D., GRAY K., SDN: Software Defined Networks, O'Reilly, 2013
12. RHOTON J., DE CLERCQ J., NOVAK F., OpenStack Cloud Computing: Architecture Guide, RP publisher, 2014
13. SABOOWALA H., ABID M., MODALI S., Designing Networks and Services for the Cloud: Delivering business-grade cloud applications and services, CISCO Press, 2013
14. SHUKLA V., Introduction to Software Defined Networking - OpenFlow & VxLAN, Createspace, 2013
15. WEN H., TIWARY P.K., LE-NGOC T., Wireless Virtualization, Springer, 2013
16. YELURI R., CASTRO-LEON E., Building the Infrastructure for Cloud Security: A Solutions View, Apress Open, 2014

CHAPTER 5

DATA CENTER SECURITY



LEARNING OBJECTIVES

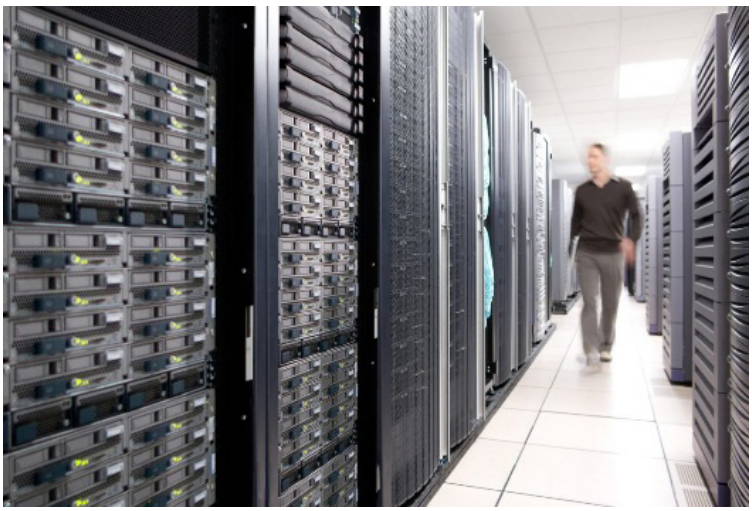
After studying this chapter, you will be able to:

1. Understand data centers: best practices for security and performance
2. Explain physical security for data centers
3. Explain importance of data center security

INTRODUCTION

Data center security is the pursuit of practices that make a data center more secure from a range of different kinds of threats and attacks. The data center, as a major primary resource for companies, deserves this kind of dedicated security effort.

Data centers are integral repositories for business data. Typically, these central IT infrastructures are served by sophisticated networking equipment and resources. They will take in business data from diverse places, and store it for eventual use. Complex designs may use IT tools called middleware to usher data to and from a data center or otherwise guide it within the system. Data center security aims to make that data less accessible to hackers or anyone else who may seek unauthorized access.



One type of data center security is physical security. Experts may recommend various kinds of facilities or site security such as setbacks, landscaping, thick walls, and other aspects of a building that will create physical barriers around the data center. Another main component of data center security is essentially network security. Because data centers are served by networks, security engineers need to plan adequate protections into those network trajectories that run to a data center. That may mean installing firewalls, anti-virus programs, or anything that prevents data breaches or other security issues.



Data center security may also vary according to the type of data center in question. For example, identifying a data center according to one of four tiers shows how fault tolerant that system is and what kind of security it may need. Generally, a lot of experts tend to recommend redundant utilities for data centers, such as multiple power sources, multiple environmental controls and more. Adequate security for a data center may also depend on the level of network virtualization that is used, or on other aspects of complex IT setups.

DATA CENTERS: BEST PRACTICES FOR SECURITY AND PERFORMANCE

As the data center spurs higher levels of business growth and opportunities, it becomes more critical to protect it. Threats are increasing in sophistication and the data center is a high-value target.

The explosive growth of mobile computing requires an open infrastructure with trusted access to data center resources. New technologies such as **virtualization** and cloud computing change how workloads are handled, but the larger "attack surface" creates more vulnerability. Compliance mandates such as PCI DSS impose stringent requirements for policy enforcement and controls. Web 2.0 technologies and e-commerce require security that can scale to protect a mission-critical computing environment. More and more, security is becoming an integral part of robust and thriving data center solutions.

"Cyber-Security is much more than a matter of IT."

—Stephane Nappo



There are four objectives in the design of any high performance data center: Security, availability, scalability, and manageability. Today's business and competitive environment requires that each of these objectives be considered from a complete end-to-end perspective. Clients or users connecting to the data center measure performance by timely access to the desired application data, whether the connection is point to point or via an Internet connection. The user must perceive reasonable response and connection time. Access is no longer measured solely on an arbitrary measure of "network availability" or ping times. This fact does not lesson the goal of developing networks that provide 99.999 percent availability.

Security

Security can no longer be treated as an afterthought when it comes to optimum network design. Trial-and-error networking is not an option, as a single **vulnerability** could compromise the corporate lifeblood, the network. In such a high-stakes environment, only specialized expertise is acceptable. There are challenges in offering encryption, certification, directory, network, and other security components that enable what one would consider a 100 percent secure network. While industry struggles with developing the technology to provide these protective components, the IT manager must still cope on a daily basis to reduce the network's imminent risk.



A complete network security solution includes **authentication**, authorization, data privacy, and perimeter security. Perimeter security is traditionally provided by a firewall, which inspects packets and sessions to determine if they should be transmitted or dropped. In effect, firewalls have become a single point of network access where traffic can be analyzed and controlled according to parameters such as application, address, and user for both incoming traffic from remote users, and outgoing traffic to the Internet.

In general, firewalls are intended to protect resources from several kinds of attacks such as passive eavesdropping/packet sniffing, IP address spoofing, port scans, denial-of-service (DoS) attacks such as SYNchronize-ACKnowledge attack (SYN flooding), packet injection, and application-layer attack.

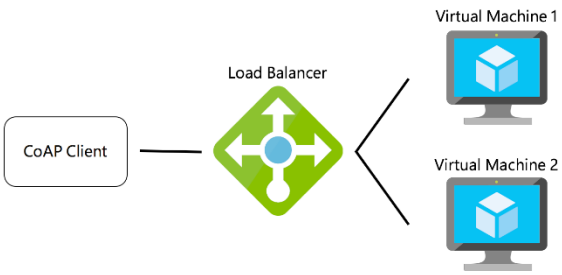
Availability

Availability is generally ensured by the overall network design and implemented in several ways. First, networks are designed with steps to minimize the occurrence of service problems and the time to recover from problems (such as backup recovery policies). Second, high availability must be considered at each layer of the Open Systems interconnection (OSI) reference model, with redundancy and failover provisions made at the physical, data link (for example, Ethernet), network (for example, IP), and application layers. The most effective solutions are those with consistent engineering considerations tightly integrated throughout the data center, rather than those approached with a series of "point-solution" products and techniques.



Scalability

Scalability must be provided in every data center. Server load balancing is the norm, and techniques such as **reverse proxy** caching are often used to offload servers. Server load balancing, like other aspects of data center design, must also be content-aware, preferably using delayed binding, full URL and cookie inspection, and "sticky (server) connections" as part of the logic of choosing a server for each user request. Specific content may be in high demand and considered "hot." Because this content may not be known in advance (such as breaking news stories), advanced data centers should also have the ability to automatically and immediately identify and replicate hot content to overflow or backup servers or cache to ensure the ability to support the increased demand without compromising performance.

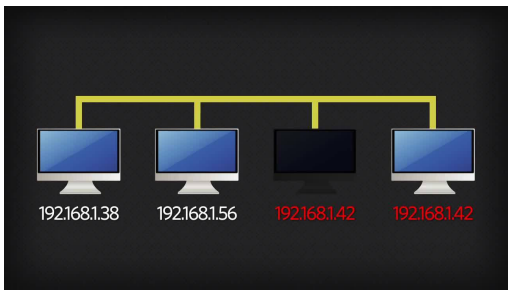


Both the complications and available solutions increase greatly if the data center is part of a geographically dispersed set. The ability to provide content at multiple sites allows the served data to be "closer" to the requesting client, thus providing

faster response but also higher availability (due to partial or complete redundancy of the data). But multidata center design must then include considerations of management of the data content, distribution of updates, synchronization of different sources, proper routing of requests, handling of downed servers, additional security and so on.

Manageability

Manageability means much more than simply knowing if a server or other network element is "up or down." Especially in service provider data centers supporting multiple customers, the ability to assess service levels on a per-customer basis is essential to the offering and administration of service-level agreements (SLAs). Managing IP address assignments, keeping track of network configurations, and not losing site of trouble tickets and alarms often requires use of a mechanized network operations center (NOC) support system. Good manageability tools and qualified personnel supporting the infrastructure translate into lower operations costs, since time is not wasted trying to resolve indications from conflicting management systems and higher customer satisfaction.



PHYSICAL SECURITY FOR DATA CENTERS

The aim of physical data center security is largely the same worldwide, barring any local regulatory restrictions: that is, to keep out the people you do not want in your building, and if they do make it in, then identify them as soon as possible (ideally also keeping them contained to a section of the building). The old adage of **network security** specialists, that "security is like an onion" (it makes you cry!) because you need to have it in

layers built up from the area you are trying to protect, applies just as much for the physical security of a data center.

> *"Passwords are like underwear: don't let people see it, change it very often, and you shouldn't share it with strangers."*
>
> **—Chris Pirillo**

There are plenty of resources to guide you through the process of designing a highly secure data center that will focus on building a "gold standard" facility capable of hosting the most sensitive government data. For the majority of companies, however, this approach will be overkill and will end up costing millions to implement.

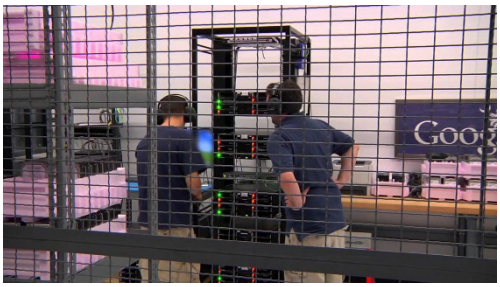


When looking at physical security for a new or existing data center, you first need to perform a basic risk assessment of the data and equipment that the facility will hold according to the usual impact-versus-likelihood scale (i.e., the impact of a breach of the data center versus the likelihood of that breach actually happening). This assessment should then serve as the basis of how far you go with the physical security. It is impossible to counter all potential threats you could face, and this is where identification of a breach, then containment, comes in. By the same token, you need to ask yourself if you are likely to face someone trying to blast their way in through the walls with explosives!

There are a few basic principles that any data center build should follow, however:

- Low-key appearance: Especially in a populated area, you do not want to be advertising to everyone that

you are running a data center. Avoid any signage that references "data center" and try to keep the exterior of the building as nondescript as possible so that it blends in with the other premises in the area.



- Avoid windows: There should not be windows directly onto the data floor, and any glazing required should open onto common areas and offices. Use laminate glass where possible, but otherwise make sure windows are double-glazed and shatter resistant.
- Limit entry points: Access to the building needs to be controlled. Having a single point of entry for visitors and contacts along with a loading bay for deliveries allows you to funnel all visitors through one location where they can be identified. Loading-bay access should be controlled from security or reception, ideally with the shutter motors completely powered down (so they cannot be opened manually either). Your security personnel should only open the doors when a pre-notified delivery is arriving (i.e., one where security has been informed of the time/date and the delivery is correctly labelled with any internal references). Of course all loading-bay activity should also be monitored by CCTV.
- Anti-passback and man-traps: Tailgating (following someone through a door before it closes) is one of the main ways that an unauthorized visitor will gain access to your facility. By implementing man-traps that only allow one person through at a time, you force visitors to be identified before allowing access. And **anti-passback** means that if someone tailgates into a building, it's much harder for them to leave.



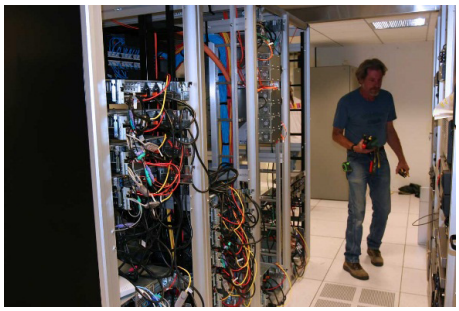
- Hinges on the inside: A common mistake when repurposing an older building is upgrading the locks on doors and windows but leaving the hinges on the outside of the building. This makes is really easy for someone to pop the pins out and just take the door off its hinges (negating the effect of that expensive lock you put on it!).
- Plenty of cameras: CCTV cameras are a good deterrent for an opportunist and cover one of the main principles of security, which is identification (both of a security breach occurring and the perpetrator). At a minimum you should have full pan, tilt and zoom cameras on the perimeter of your building, along with fixed CCTV cameras covering building and data floor entrances/exits. All footage should be stored digitally and archived offsite, ideally in real time, so that you have a copy if the DVR is taken during a breach.



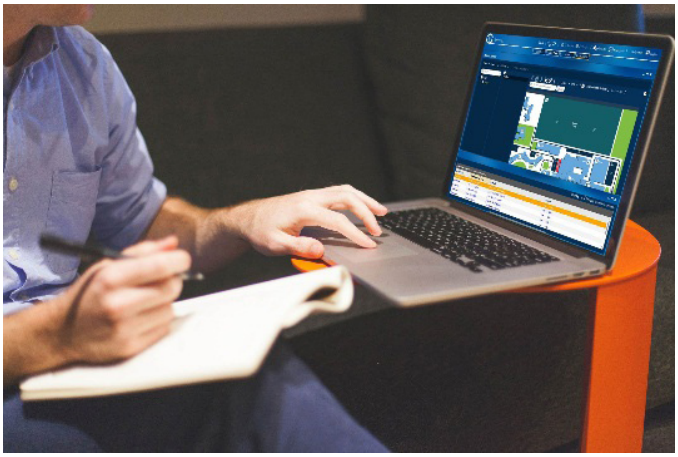
- Make fire doors exit only (and install alarms on them): Fire doors are a requirement for health and safety, but you should make sure they only open outward and have active alarms at all times. Alarms need to sound if fire doors are opened at any time and should indicate, via the alarm panel, which door has been opened; it could just be someone going out for a cigarette, but it

could also be someone trying to make a quick escape or loading up a van! On the subject of alarms, all doors need to have alarms and be set to go off if they are left open for too long, and your system should be linked to your local police force, who can respond when certain conditions are met.

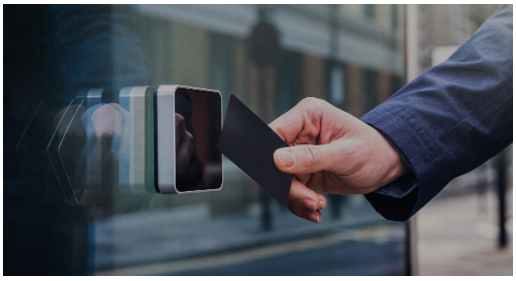
- Door control: You need granular control over which visitors can access certain parts of your facility. The easiest way to do this is through proximity access card readers (lately, biometrics have become more common) on the doors; these readers should trigger a maglock to open. This way you can specify through the access control software which doors can be opened by any individual card. It also provides an auditable log of visitors trying to access those doors (ideally tied in with CCTV footage), and by using **maglocks**, there are no tumblers to lock pick, or numerical keypads to copy.



- Parking lot entry control: Access to the facility compound, usually a parking lot, needs to be strictly controlled either with gated entry that can be opened remotely by your reception/security once the driver has been identified, or with retractable bollards. The idea of this measure is to not only prevent unauthorized visitors from just driving into your parking lot and having a look around, but also to prevent anyone from coming straight into the lot with the intention of ramming the building for access. You can also make effective use of landscaping to assist with security by having your building set back from the road, and by using a winding route into the parking lot, you can limit the speed of any vehicles. And large boulders make effective barriers while also looking nice!



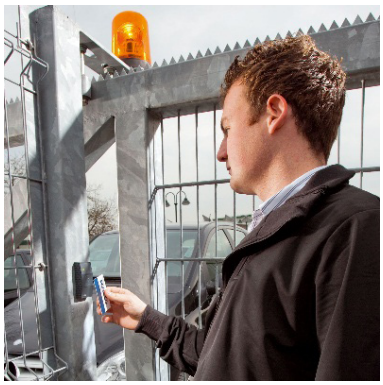
- Permanent security staff: Many facilities are manned with contract staff from a security company. These personnel are suitable for the majority of situations, but if you have particularly sensitive data or equipment, you will want to consider hiring your security staff permanently. A plus and minus of contract staff is that they can be changed on short notice (e.g., illness is the main cause of this). But it creates the opportunity for someone to impersonate your contracted security to gain access. You are also at more risk by having a security guard who does not know your site and probably is not familiar with your processes.
- Test, test and test again: No matter how simple or complex your security system, it will be useless if you do not test it regularly (both systems and staff) to make sure it works as expected. You need to make sure alarms are working, CCTV cameras are functioning, door controls work, staff understands how visitors are identified and, most importantly, no one has access privileges that they should not have. It is common for a disgruntled employee who has been fired to still have access to a building, or for a visitor to leave with a proximity access card that is never canceled; you need to make sure your HR and security policies cover removing access as soon as possible. It's only by regular testing and auditing of your security systems that any gaps will be identified before someone can take advantage of them.



- Do not forget the layers: Last, all security systems should be layered on each other. This ensures that anyone trying to access your "core" (in most cases the data floor) has passed through multiple checks and controls; the idea is that if one check fails, the next will work.

The general rule is that anyone entering the most secure part of the **data center** will have been authenticated at least four times:

1. At the outer door or parking entrance. Do not forget you will need a way for visitors to contact the front desk.
2. At the inner door that separates the visitors from the general building staff. This will be where identification or biometrics are checked to issue a proximity card for building access.



3. At the entrance to the data floor. Usually, this is the layer that has the strongest "positive control," meaning no tailgating is allowed through this check. Access should only be through a proximity access card and all access should be monitored by CCTV. So this will generally be one of the following:

- A floor-to-ceiling turnstile. If someone tries to sneak in behind an authorized visitor, the door gently revolves in the reverse direction. (In case of a fire, the walls of the turnstile flatten to allow quick egress.)
- A man-trap. Provides alternate access for equipment and for persons with disabilities. This consists of two separate doors with an airlock in between. Only one door can be opened at a time and authentication is needed for both doors.

4. At the door to an individual server cabinet. Racks should have lockable front and rear doors that use a three-digit combination lock as a minimum. This is a final check, once someone has access to the data floor, to ensure they only access authorized equipment.

The above is not an exhaustive list but should cover the basics of what you need to consider when building or retrofitting a data center. It is also a useful checklist for auditing your colocation provider if you do not run your own facility.



In short, all physical security comes down to managing risks, along with the balance of "CIA" (confidentiality, integrity and access). It's easy to create a highly secure building that is very confidential and has very high integrity of information stored within: you just encase the whole thing in a yard of concrete once it's built! But this defeats the purpose of access, so you need a balance between the three to ensure that reasonable risks are mitigated and to work within your budget—everything comes down to how much money you have to spend.

Physical Data Center Security Considerations

When IT executives talk about security, it often revolves around defending against **cyber attacks** using clever technology. However, data security is just part of the equation. Physical security -- keeping the bad guys from physically accessing servers -- is essential.



Discussions on physical security generally address three main components: *Something you have, something you know, and something you are*.

The something you know could be a password or passphrase. Your Personal Identification Number for getting cash from your bank's automatic teller machine or your login at work is something you know.

Something you are is a biometric scan, such as a fingerprint or retina scan. New biometric devices promise to add the user's heartbeat to the mix, but that is still in the early stages.

The something you have might be some type of token or card key that contains coded information about who you are and what attributes you have to access secure environments. Generally speaking, these might include a basic card key with a magnetic stripe not unlike your ATM card, a proximity card that only allows access when the card is within a predefined distance from a server, or perhaps a USB key or keyfob that holds information about the owner. Some proximity card readers can screen out unauthorized cards at the reader.



But there are a lot of other ways companies use physical security to protect servers and data. Amazon.com is known for having massive warehouses throughout the country that are disguised as merely large, nondescript buildings. They do not sport the Amazon logo nor have Amazon tractor-trailer trucks pulling up to the loading dock. Essentially these buildings are hiding in plain sight, looking like every other **warehouse**.

Years ago a Bank of America data center in downtown Los Angeles used the same approach. The bank moved in to a large, concrete building that had few windows. Drivers heading into the Downtown area could see the building from the freeway but there were no markings to indicate what the building housed.

Today data centers are situated throughout the United States. Often companies build these facilities several miles outside of cities and away from airports, train tracks, chemical facilities, and of course, large fault lines; Silicon Valley tends to be an exception to the latter, but it is still a good idea to protect the data center from potential man-made and natural disasters.

Protecting the Data Center: Security Considerations

One simple approach to stopping drive-by attackers from breaching a data center is simply to not let the building be seen from the street. A large berm surrounding the building so that it is unseen from drivers or building the facility below street level with trees and shrubbery blocking passersby from seeing the building is a popular approach. Limiting access to the facility is advised, requiring those wishing to enter to pass through a specific gate protected by cameras, locked doors and other security precautions. High fences -- at least 20 feet high, according to the SANS Institute -- and CCTV camera surveillance also are recommended.



But physical security is not only limited to the outside of the building. Data centers need redundant utilities so if one system fails, there is a backup. This includes water, power, telephone lines and air filtration systems. Battery backups also are essential to ensure security systems; **heating, ventilation and air conditioning (HVAC**); and other essential systems continue to operate in case of an area-wide power outage. Today most companies also have redundant Internet access to different physical infrastructures and providers.

Depending on the company's security requirements, there could be restrictions on the removable media allowed in the data center. In order to eliminate the ability for a user to introduce malware into servers, some large banks do not allow users to bring USB connected drives into the building. If an employee or guest needs to bring a PowerPoint file, for example, they load it onto a terminal in the reception area. Data is transferred to a protected virtual machine on a server where it is scanned for malware. The external drive is left at the reception desk until the guest leaves. When the presentation is given, it is pulled from the VM on a network sequestered from the corporate network.



Computer rooms should be on the interior of the building, ideally with no outside walls. If there is an exterior wall, SANS

recommends having barriers on the outside to prevent someone from getting too close to the wall.

There is an old saying that *if you want to break into the back door of a building, drive a car through the front door*. While diversions might work for some cases, experts recommend foot-thick concrete walls and retractable crash barriers at vehicle entry points. Thick, concrete walls work well against diversions, as well as direct attacks.

Speaking of doors, fire doors and other doors that are designed to allow employees to escape during an emergency should not have handles on the outside of the door. When these doors are opened, an alarm should sound and an alert should light up on the console of the security team. Remember that breach protection is not only designed to stop attackers from getting in but also data from going out.



The National Security Agency uses a private cloud for protecting much of its sensitive data. The cloud provides several additional layers of physical and **logical security** to the data than the NSA had when it used only its own data center, according to NSA Chief Information Security Officer Sally Holcomb.

In addition to the traditional physical access controls the NSA uses for individuals entering a restricted building, additional physical security enhancements include limiting who has access to data based on a variety of attributes encoded not only in the data, but also in tokens and other physical devices. Even if a person has a token and attributes that permits them physical access to a given system, the encoded attributes might not allow them access to specific data that resides on the system.

In addition to site and data center physical security, the third leg of the physical security stool is people. As noted, identity, credential and access controls management (ICAM) tools are essential, but there are some other basics that should be addressed. Any vendor with access to the facility (cleaning crew, security guard services, caterers and the like) need to be vetted before access is permitted.



Background and security checks should be required under contract with the service provider and available for immediate review should a breach occur. Visitors should be escorted within secure areas, and restrooms for visitors and contractors should be provided.

There is a maxim that states "Nothing is foolproof in the hands of a fool." Even organizations that have top-shelf data and physical security experts and technology have been breached. This is evidenced by successful attacks on the NSA, various governmental agencies around the world, major financial institutions, and even data security vendors. Virtually every company, regardless of size or perceived importance, likely has something a cyber or traditional thief might want to steal. All this leads to one of the great Irish provers about physical security: "Never bolt your door with a boiled carrot."

Physical Data Center Security Systems

Physical security is a little different from other data center technologies in the most products, for the most part, can be considered vendor neutral -- the types of doors you buy, physical locks, **surveillance** cameras or cages are not significantly distinct.

When shopping for a physical security system, there is less of a question of brand than whether you choose to do it or not. In physical security, brands are pretty much the same; they are commodities when it comes most of the technology.

Biometrics

While the fingerprint scanner on a laptop might only require an image comparison of the fingerprint in order to log on a user, more sophisticated biometrics also look for other attributes, such as a heat signature, to ensure the person using the fingerprint ID is alive. Today's sophisticated fingerprint, or for that matter handprint scanners, require more than simple pattern-matching algorithms.



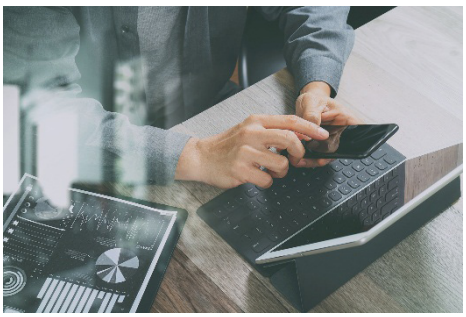
Other biometric scanners, such as those for facial recognition, also avoid some of the pitfalls of simple photo comparisons. Today's advanced facial recognition technology compares not only the shape and angles of various facial components, but measures the distance from, for example, a location on the cheek to the edge of the eye. This geometric footprint is not based on what a person looks like, but rather their physical attributes that can be measured and turned into a numeric identity. No matter how good of a mask one might use to pretend to be someone else, the software likely will defeat the imposture's attempt to impersonate someone else.

While simple biometric image-comparison scans can provide some level of security, it falls to the attribute-based access control (ABAC) technologies that separate the proverbial locked door that can be easily breached from the technology. Iris scans are unique to each individual and are as accurate as fingerprints.

Authentication Levels

While biometrics, be it facial recognition, fingerprint identification, a **retina scan**, or an iris scan, is just one component of an effective physical defense. As previously mentioned, for the most effective security, users need three levels, or factors, of authentication: Something they have, something they know and something they are. The biometrics component represents the *something they are* requirement.

To address the *something they have* factor, users might be given a card key, keyfob or other type of token they need to obtain access. The ubiquitous card key with a magnetic strip on the back is based on 1960s technology. While the strips can be programmed when the user is assigned the card, they generally are not reprogrammed while in use. Similar to the magnetic stripe on the back of a credit card, the stripe on an access card is pre-programmed and read-only. If the company needs to change the attributes on the card, a new card is assigned to the user.



One problem with mag stripe card keys is that they are spoofed easily. A stolen credit card can be read easily to determine credit card numbers, expiration dates, and other data. However, while U.S.-based consumers have legal protections if their credit card is stolen, companies that use this technology have no such protections if criminals are able to open locked doors to access rooms with private data.

The traditional access card is evolving. Rather than a magnetic stripe, many of today's **credit card** and other "smart cards" have an embedded chip that can be used for contact or contactless communications. The cards, which can be reprogrammed, can store 100 times the data stored on magnetic stripes.

A second type of smart card has both memory and a microprocessor onboard. The microprocessor permits the cards to make their own decisions about an action based on the situation.



Proximity key technology is popular for access control. The 125 KHz technology is the industry standard for most industries use radio frequency identification (RFID) technology to read and write data. RFID is a one-way technology in that data flows from the tag or card to the reader. The 13.56 MHz devices use **Near Filed communications** (NFC) technology that enables communications between devices over a distance of 10 cm or just under four inches. NFC supports existing ISO/IEC standards 14443 (Types A/B) proximity cards and 15693 vicinity cards. In addition to transmitting data from the card to the reader, it also can flow data in both directions and it can act as a peer-to-peer (P2P) communications.

This P2P technology can be used to permit cell phones to be used in place of proximity cards to communicate open doors and for other access controls. The devices can be programmed so that the user can only use them if they have the proper attributes, such as job title, security clearance, or are authorized to access the data stored on the servers in a given server room.

Passwords, PIN number or logins are examples of the *something they know* component. While the media has been full of stories recently about compromised passwords, passwords and PIN codes still play a role as something only the user should know. By themselves, passwords and PIN number, especially short or common passwords or PIN numbers, can be a weak

link. When combined with the other factors they make up the three-legged stool of security.

Physical Security Precautions

For instance, Jason Bell, manager of Security Technology Solutions at Purolator Inc. in Mississauga, Ontario, says that his company uses a risk model to determine the types of physical security precautions used for the various Purolator offices. The first step is taking into consideration employee head count, customer volume, the number of company vehicles, criticality of the facility, all of the assets in the building, cash value of the building, and its historical significance. Also considered are the physical environment (swamps, for example, are not good locations for concrete walls or electric fences), customer base and the types of products that generally move through that facility.



If a facility already exists at or near the new site, historical losses, staff turnover and preference indicators also put into the matrix. An analysis of the data will determine the number and types of cameras required, the types of access control precautions put in place, and other robbery prevention systems.

Everyone is a supporter of biometric technology, but the challenge is enrolling the staff and keeping the databases current. Using a card key for basic security is easier as there is no enrollment necessary and keys are easily replaced if lost or stolen. For more sensitive facilities, more extensive access controls are used, he notes.

Currently Purolator is not using any NFC devices because costs are high. The 125 KHz RFID devices can be used on any open platform and are sufficiently secure, he says. The NFC

devices require an HID Global system because that company still holds the patents. "The risk of someone cloning (an RFID) card to work on your system is mind-blowing," because the codes on each card are unique to the customer.

Another reason Bell is not sold on NFC devices today are that NFC readers will read every card that is within its read proximity range. If a person has more than one NFC card in their wallet that is seen by a reader, for example, the individual is automatically rejected, even if one of the cards is legitimate.



Bell notes that Purolator, like all other package delivery services, is at the greatest risk in the mornings before the trucks go out for deliveries and in the evening when packages are collected for the next day's processing. During afternoon hours, there is little of value for someone trying to access the facility physically. This differs from office environments when daytime hours offer plethora of targets from documents on desks to access to the physical network.

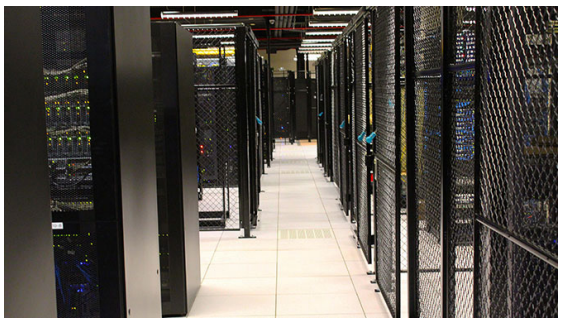
CISOs tend to worry only about what goes on within the facilities themselves. Often the physical security details are handled by other departments, such as operations or facilities. The mindsets for physical and data-centric security are not the same and use different ways to measure risk. He suggests that companies have the physical and data center security teams work more closely on risk assessment and defense.

IMPORTANCE OF DATA CENTER SECURITY

When weighing data center provider options, security is often one of the first considerations on the minds of potential customers. In order to feel comfortable outsourcing important company information, prospects will want to be ensured that their data – wherever it is hosted – will be protected and monitored in the most secure environment possible. As a data center, there are a variety of measures that can help to guarantee this level of security is established and maintained.

"I really think that if we change our own approach and thinking about what we have available to us, that is what will unlock our ability to truly excel in security. It's a perspectives exercise. What would it look like if abundance were the reality and not resource constraint?."

—Greg York



Building Infrastructure & Location

Without even entering a data center, the physical infrastructure and relative location of a provider can be very telling in regards to security. First and foremost, a strategically located data center will be positioned some distance from other offices or headquarters affiliated with it. 20 miles is a good rule of thumb, and remaining at least 100 feet from the main road is also best practice.

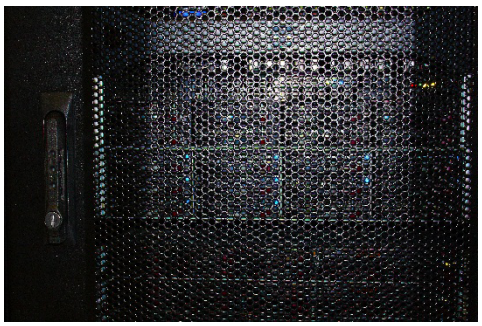
While it can be hard to distinguish at face-value, ensuring that a facility is protected from high-risk man-made and natural disaster areas is also important to keep information safe and secure. Examples of high-risk areas may include airports, chemical facilities, and power plants – anywhere that data may be compromised by an external force.



Additionally, a provider that visibly advertises itself to the community through signs or other public displays puts their customers' information at risk by divulging its location. Maintaining a relatively inconspicuous physical presence is key for data center security.

Access Control Systems & Biometrics

The way that visitors access a data center facility plays a significant role in the facility's security protocol. Biometric identification has become an increasingly popular method of entry in recent years, establishing itself as a standard procedure when attempting to gain access to sensitive areas of a facility. There are many developing access control technologies; many of them sophisticated enough to look for a variety of attributes as opposed to just one single algorithm. **Biometrics**, however, have grown in popularity due to their effectiveness and lack of invasiveness when compared to other options (such as retinal scanning).

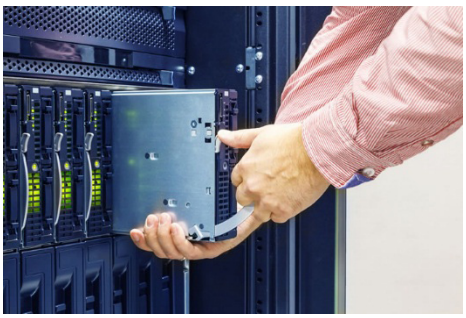


In addition to biometrics, visitors may also be given a card key, a key fob or another means of access, integrating an extra

layer of security within the facility. Additional measures such as mantraps – consisting of two separate doors with an airlock in between – function to guarantee that only one door to the data center can be opened at a time – with authentication required for both doors. All of the entrances to the data center should function similarly to this, and should include a form of hierarchy to validate visitors as they come and go from the facility. This might include authentication through the outer entrance (naturally affording a way for visitors to access the front desk), authentication through the inner entrance in order to separate the visitor area from the general staff area, authentication through the door to the room where the actual servers or other critical IT equipment is located (a mantrap), and authentication to the data center itself through a mantrap or similar setup. Once inside of the data center itself, many facilities offer customers the option of locked cabinets as well to protect their data.

Surveillance

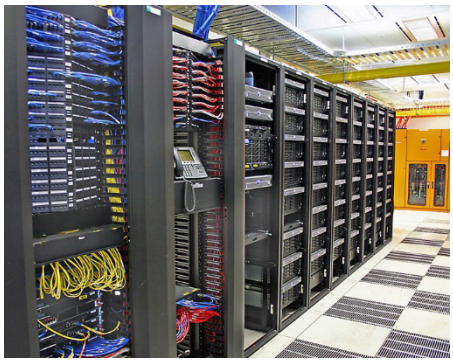
Constant surveillance of a data center is vital for monitoring daily operations. In order to ensure that all areas are functioning properly and that each element of the building is secure and protected at all times, surveillance equipment (cameras or otherwise) should be installed throughout the building at every entrance, exit, and access point.



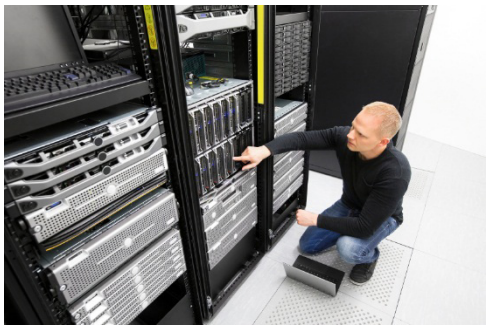
Redundancy

Redundancy in a data center amplifies security by providing an extra layer of equipment, personnel, or storage in the event of the primary source's failure.

For example – at the level of equipment – generators, batteries, heating, ventilation and air conditioning (HVAC), water, power, and telephone lines are all utilities that may be redundantly reinforced. Protecting this equipment is crucial in the event of an area-wide power outage or natural disaster, guaranteeing that if one system fails, there is a backup in place. Moreover, equipment redundancy ensures that in the event of a disruption, all systems continue to operate.



Personnel redundancy simply refers to safeguarding the data center when visitors are present with the help of employees or staff. An instance of this may occur when there are contractors or repair crews present in the building. Ideally, an employee would accompany visiting individuals at all times – especially if they are performing maintenance on critical equipment.



Finally, storage redundancy refers to an additional level of network security within a data center. Many companies offer redundant Internet access to different physical infrastructures and carriers, offering customers the element of choice in determining the best security solution for them. Some may also consider a private cloud hosting option in order to provide an extra layer

of data security for sensitive information, another convenience afforded to customers concerned with backing up their data assets.

The primary function of data center security is to ensure that all foreseeable measures are taken to protect customers' vital information and data assets. While evaluating data center providers, assessing facility security may be the first step in the decision making process.

CONCLUSION

This chapter discussed many basic concepts related to security. The design of security in the Data Center requires understanding the security threats, the attacks, and the technology to counter them. You can use such tools as ACLs, firewalls, IDSs, and security features on switches and routers to mitigate the effects of attacks against servers and network devices. Designing the Data Center requires configuring the appropriate ciphers on SSL and IPSec devices, generating and installing certificates, and enabling HMAC on routers—all of which require an understanding of the key concepts of cryptography, AAA, and VPNs.

Creating security policies, secure management, and incident response and attack mitigation are as important as understanding the technology. The last part of this chapter described the key considerations for creating a Data Center security framework.

CASE STUDY

REMOTELY REBOOTING SECURITY CAMERAS



The IDS-710HP PoE Switch will fit into any environment where extensive security, QoS, and network interoperability are required.

To be effective in areas that require surveillance, security cameras are often strategically placed at the top of a light pole or on the side of a building. However, these locations pose several challenges for surveillance system technicians, designers, and installers, including the following.

- Climate — Outdoor security cameras are designed to operate in rain, sleet, and snow. The rest of the system, such as the switch, router, Ethernet extender, or media converter, needs to fit inside an appropriate NEMA enclosure to be protected against the elements. However, even inside an enclosure, this equipment must be rated to operate in temperatures of minus 40° to 70°C as a safeguard in hot and cold environments.
- Power — Light poles and building exteriors rarely have an AC or DC power source. Therefore, the cameras are generally connected to a PoE media converter, PoE Ethernet extender, or PoE switch that can supply up to 100 W of power over the Ethernet cable.
- Maintenance — Like most electronics, surveillance cameras sometimes need to be rebooted. Because they are spread out in hard-to-access locations, it is best if the PoE device supports the capability to reboot the attached cameras remotely.
- Aesthetics — Most municipalities would like the cameras and system enclosures to be discreet and blend in with the surrounding architecture. Because surveillance system equipment can be quite compact, custom enclosures are often designed to house everything.

Louis-Marie Bouvattier, project manager at Engie Solutions' Ineo Infracom, shared his experience about a surveillance system the company deployed in France.

"Every surveillance system is different, but the core components of our build included the cameras, a Perle Managed Industrial Ethernet Switch, and a 24vDC PSU housed in a weatherproof enclosure," he said. "The Perle Switches were chosen because they supported the required management features and allowed us to add secure authentication methods through IEEE 802.1X or MAC addresses." While this setup worked very well, there was one area in which Engie wanted to improve.

"Surveillance cameras can sometimes freeze, glitch, or be otherwise temperamental — requiring a reboot," Bouvattier said. "With our existing setup, we had to send staff out to reset and reboot cameras regularly. This needed to change. On paper, the cameras we were using had power requirements of 30 W. However, we found that they sometimes operated erratically unless at least 60 W of power was available. A standard PoE switch would not be enough. We needed a PoE switch that supported IEEE 802.3bt." The Perle IDS-710HP-XT was the solution. Able to provide up to a total of 450 W of power across its eight ports to connected devices, it was also small enough to fit into enclosures designed to be discreet and blend in with the surrounding architecture. And, the PD Power Reset feature provides an easy way for the surveillance cameras to be remotely rebooted at any time.



With operating temperatures of minus 40°F to 158°F, extensive management features, secure authentication methods, and network connectivity through two SFP Slots supporting 1G/2.5G Fiber or 10/100/1000 Copper, the IDS-710HP PoE Switch will fit into any environment where extensive security, QoS, and network interoperability are required. "The IDS-710HP-XT matched all our requirements, dimensional and technical," Bouvattier said. "This was the missing piece to create a flexible solution that could be used in all urban security camera settings."

KEY TERMS

Virtualization
Reverse proxy
Maglock
Logical Security
Vulnerability
Network security
Cyberattack
Surveillance
Authentication
Anti-passback
Warehouse

DISCUSSION QUESTIONS

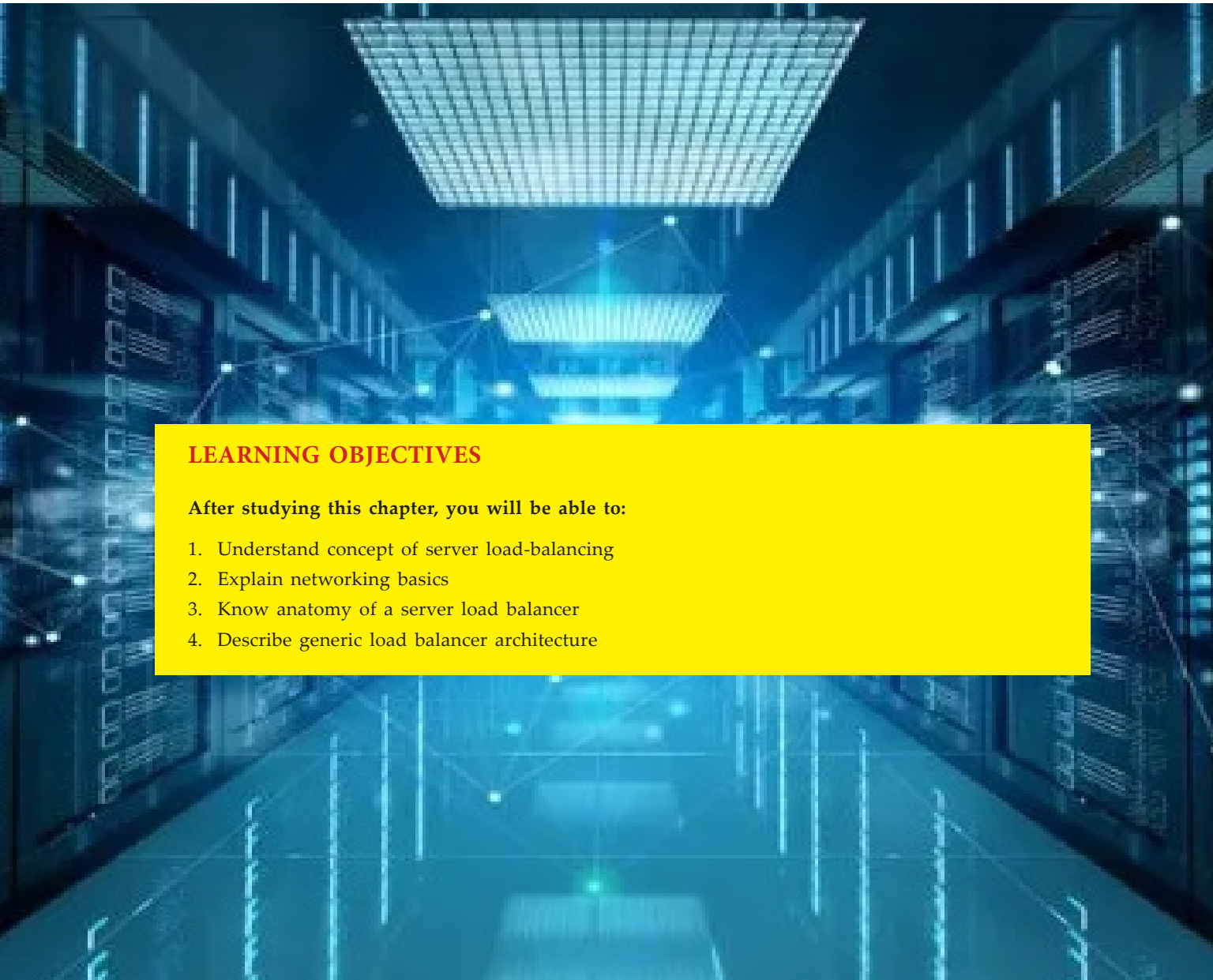
1. What is data center security? Discuss.
2. What are the objectives in the design of any high performance data center? Explain.
3. Discuss how to protect physical data center.
4. How do you deal with potential data loss? Describe.
5. What are the security aspects provided with cloud?
6. Why security is important in data center?
7. What are the security devices used on the data Centres?
8. What are the most important data center security standards?
9. How do you physically secure a data center?

REFERENCES

1. A. Anandkumar, C. Bisdikian, and D. Agrawal. Tracking in a Spaghetti Bowl: Monitoring Transactions Using Footprints. In ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), 2008
2. A. Brown, G. Kar, and A. Keller. An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Environment. In IFIP/ IEEE International Symposium on Integrated Network Management (IM), 2001
3. A. Gavrilovska, K. Schwan, and V. Oleson. A Practical Approach for 'Zero' Downtime in an Operational Information System. In IEEE International Conference on Distributed Computing Systems (ICDCS), 2002
4. B. C. Tak, C. Tang, C. Zhang, S. Govindan, B. Urgaonkar, and R. N. Chang. vpath: precise discovery of request processing paths from black-box observations of thread and network activities. In USENIX Annual Technical Conference (ATC), 2009
5. C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan. Statistical Techniques for Online Anomaly Detection in Data Centers. In IFIP/ IEEE International Symposium on Integrated Network Management (IM), 2011
6. C. Yuan, N. Lao, J.-R. Wen, J. Li, Z. Zhang, Y.-M. Wang, and W.-Y. Ma. Automated Known Problem Diagnosis with Event Traces. In European Conference on Computer Systems (EuroSys), 2006
7. E. Galstad. Nagios Enterprises LLC., 2008 [63] S. Meng, S. Kashyap, C. Venkatramani, and L. Liu. REMO: Resource-Aware Application State Monitoring for Large-Scale Distributed Systems. In IEEE International Conference on Distributed Computing Systems (ICDCS), 2009
8. E. Kiciman and A. Fox. Detecting ApplicationLevel Failures in Component-based Internet Services. IEEE Trans. on Neural Networks: Special Issue on Adaptive Learning Systems in Communication Networks, 2005
9. G. Jiang, H. Chen, and K. Yoshihira. Discovering Likely Invariants of Distributed Transaction Systems for Autonomic System Management. Cluster Computing (Springer), 2006
10. H. Kang, H. Chen, and G. Jiang. PeerWatch: a Fault Detection and Diagnosis Tool for Virtualized Consolidation Systems. In ACM International Conference on Automatic Computing (ICAC), 2010
11. I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik. Real-Time Problem Determination in Distributed Systems Using Active Probing. In IEEE/IFIP Network Operations and Management Symposium (NOMS), 2004

CHAPTER 6

SERVER LOAD-BALANCING



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

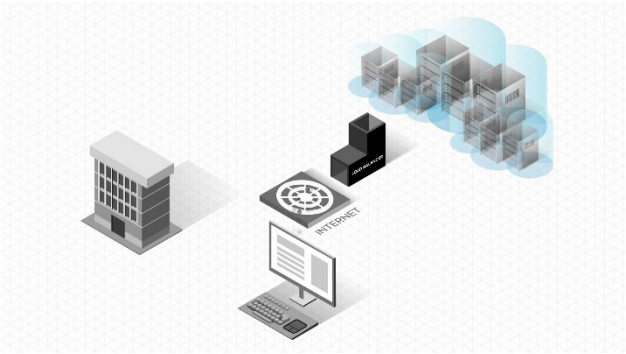
1. Understand concept of server load-balancing
2. Explain networking basics
3. Know anatomy of a server load balancer
4. Describe generic load balancer architecture

INTRODUCTION

The world of Server Load Balancing (and network-based load balancing in general) is filled with confusing jargon and inconsistent terminology. Because of the relative youth and the fierce competition of the SLB industry, vendors have come up with their own sets of terminology, which makes it difficult to compare one product and technology to another. Load balancing is the process by which inbound Internet protocol (IP) traffic can be distributed across multiple servers. This enhances the performance of the servers, leads to their optimal use, and ensures that no single server is overwhelmed. The practice is particularly important for busy networks, where it is difficult to predict the number of requests that will be issued to a server.

Typically, two or more web serves are employed in a load balancing scheme. In case one of the servers begins to get overloaded, the requests are forwarded to another server. This process brings down the service time by allowing multiple servers to handle the requests. Service time is reduced by using a load balancer to identify which server has the appropriate availability to receive the traffic.

The process, very generally, is straightforward. A webpage request is sent to the load balancer, which forwards the request to one of the servers. That server responds back to the balancer, which in turn sends the request on to the end user.



Load balancing allows service to continue even in the face of server down time due to server failure or server maintenance. If a company is using several servers and one of them fails, its website or other services will still be available to its users, as the traffic will be diverted to the other servers in the server farm.

In Global Server Load Balancing (GSLB) the load is distributed to geographically scattered server farms, depending on their load, health, or proximity.

There are several methods by which loads can be balanced. If the servers are similar in hardware specifications, the Perceptive (which predicts the server based on historical and current data) and the Fastest Response Time methods can be the best to use. On the other hand, if the hardware specifications are different, the Weighted Round Robin method, which assigns requests to servers in turn according their weights, may be a better solution because it can assign more requests to the server that can handle a greater volume.

CONCEPT OF SERVER LOAD-BALANCING

While Server Load Balancing (SLB) could mean many things, it is defined as a process and technology that distributes site traffic among several servers using a network-based device. This device intercepts traffic destined for a site and redirects that traffic to various servers. The load-balancing process is completely transparent to the end user. There are often dozens or even hundreds of servers operating behind a single URL.

"When a hacker gains access to any corporate data, the value of that data depends on which server, or sometimes a single person's computer, that the hacker gains access to."

—John McAfee

As its name indicates, server load balancing is load balancing of traffic among servers. Figure 1 presents the basic idea behind server load balancing. In Figure 1, the server load balancer is capable of determining that srv1 and srv2 should not receive any new connections because the servers are offline and beyond the maximum allowed capacity respectively. The connections then are load-balanced using a simple round-robin algorithm so that each server receives an even load (or close to even load) of the incoming requests.

An implication of load balancing is the virtualization of the service being advertised. The load balancer must take over the service so that it can determine what to do with the traffic load. In doing so, the load balancer assumes the IP address

used by the service and hides the IP addresses of the servers supporting the actual service. This makes the load balancer a virtual server using a virtual address hiding the real addresses of the real servers. The load balancer does not replace the server; it simply pretends to be the server so that the clients think they are talking to the server, when, in reality, they are interacting with the load balancer. Remember that the many servers used behind the load balancer are all hidden from the users through the use of a single IP address that is sent back to the clients in DNS responses. This **IP address** is virtual, in that no specific server is associated with it, but rather a load balancer that receives all requests destined to it. A load balancer performs the following functions:

- Intercepts network-based traffic (such as web traffic) destined for a site.
- Splits the traffic into individual requests and decides which servers receive individual requests.
- Maintains a watch on the available servers, ensuring that they are responding to traffic. If they are not, they are taken out of rotation.
- Provides redundancy by employing more than one unit in a fail-over scenario.
- Offers content-aware distribution, by doing things such as reading URLs, intercepting cookies, and XML parsing.

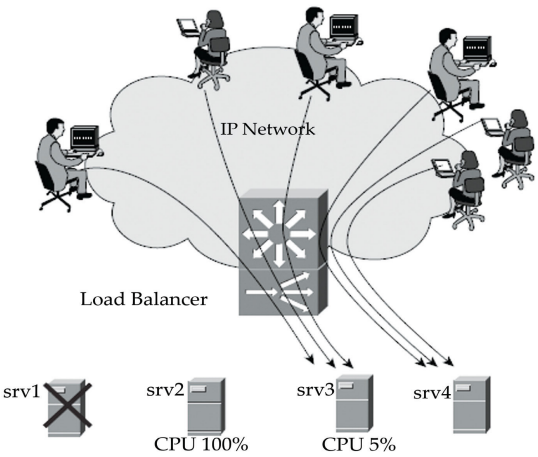


Figure 1: Server Load-Balancing

DNS-Based Load Balancing

Before SLB was a technology or a viable product, site administrators would (and sometimes still do) employ a load-balancing process known as DNS round robin. DNS round robin uses a function of DNS that allows more than one IP address to associate with a hostname. Every DNS entry has what is known as an A record, which maps a hostname (such as www.vegan.net) to an IP address (such as 208.185.43.202). Usually only one IP address is given for a hostname. With DNS round robin, it is possible to give multiple IP addresses to a hostname, distributing traffic more or less evenly to the listed IP addresses. For instance, let's say you had three web servers with IP addresses of 208.185.43.202, 208.185.43.203, and 208.185.43.204 that we wanted to share the load for the site www.vegan.net. The configuration in the DNS server for the three IP addresses would look like this:

```
www.vegan.net.      IN      A       208.185.43.202
                    IN      A       208.185.43.203
                    IN      A       208.185.43.204
```

You can check the effect using a DNS utility known as nslookup, which would show the following for www.vegan.net:

```
[zorak]# nslookup www.vegan.net

Server: ns1.vegan.net

Address: 198.143.25.15

Name: www.vegan.net

Addresses: 208.185.43.202, 208.185.43.203, 208.185.43.204
```

The end result is that the traffic destined for www.vegan.net is distributed between the three IP addresses listed, as shown in Figure 2.

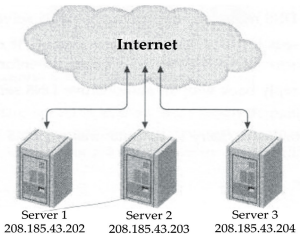


Figure 2: Traffic distribution by DNS-based load balancing.

This seems like a pretty simple way to distribute traffic among servers, so why bother spending the money and time implementing SLB at all? The reason is that **DNS round robin** has several limitations, including unpredictable load distribution, caching issues, and a lack of fault-tolerance measures. An understanding of how DNS works will help to explain the problems of DNS-based load balancing.

DNS 101

DNS associates IP addresses with hostnames so that we don't have to memorize numbers; instead, we memorize domain names. A computer needs to know the IP address, however. To perform that translation, every computer connected to the Internet, whether it be a server or a dialup user's home machine, has one or more DNS servers configured. When a user types the URL of a hostname into his browser, for instance, the operating system sends a query to the configured DNS server requesting the IP address of that hostname. The DNS server doesn't usually have that information (unless it is cached, which is something we'll discuss later), so the domain name server looks up the domain name with one of the root servers. The root servers do not have the IP address information either, but they know who does, and report that to the user's DNS server. (Servers with the appropriate DNS information are known as the authoritative DNS servers. There are usually at least two listed for a domain, and you can find out what they are by using the who is utility supplied by most Unix distributions, or through several domain registration web sites.) The query goes out to the authoritative name server, the IP address is reported back, and in a matter of seconds the web site appears on the user's screen. The entire process works like this:

1. The user types the URL into the browser.
2. The OS makes a DNS request to the configured DNS server.
3. The DNS server sees if it has that IP address cached. If not, it makes a query to the root servers to see what DNS servers have the information.
4. The root servers reply back with an authoritative DNS server for the requested hostname.

5. The DNS server makes a query to the authoritative DNS server and receives a response.

Caching Issues

Many of the limitations of DNS round robin are caused by DNS caching. To prevent DNS servers from being hammered with requests, and to keep bandwidth utilization low, DNS servers employ quite a bit of DNS caching. Since DNS information typically changes very little, this is fine for normal functions. When a DNS server goes out and gets a DNS entry, it caches that entry until the entry expires, which can take anywhere from a few days to a week (that parameter is configurable). You can configure a domain to never cache, although some DNS servers throughout the world may ignore this and cache anyway.

Traffic Distribution

Traffic distribution is one of the problems with DNS round robin that caching causes. DNS round robin is supposed to distribute traffic evenly among the IP addresses it has listed for a given hostname. If there are three IP addresses listed, then one-third of the traffic should go to each server. If four IP addresses are listed, then one-fourth of the traffic should go to each server. Unfortunately, this is not how round robin works in live environments. The actual traffic distribution can vary significantly. This is because individual users do not make requests to the authoritative name servers; they make requests to the name servers configured in their operating systems. Those DNS servers then make the requests to the authoritative DNS servers and cache the received information. Figure 3 shows a typical failure scenario with DNS-based load balancing.

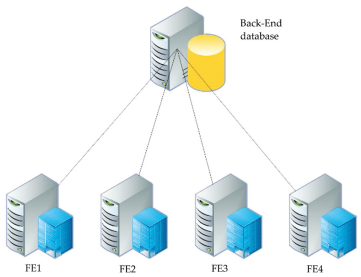


Figure 3: DNS-based load balancing

The lack of DNS update speed is also an issue when demand increases suddenly, and more servers are required quickly. Any new server entries in DNS take a while to propagate, which makes scaling a site's capacity quickly difficult.

Load Balancing Functions

The most fundamental requirement behind load balancing is scalability. Traditional scaling methods, as presented in Figure 4, include vertical scaling and horizontal scaling. Vertical scaling implies adding more CPU capacity to the service device by increasing the number of processors or adding faster processors, more memory, newer and faster components, and so on. Every step shown in Figure 4 provides additional performance at an incremental price, which could be significant if the base system is upgraded completely. Horizontal scaling implies that more devices of similar or better capacity are added. Again, at every stage, you add more processing capacity; however, the existing equipment is not upgraded and the incremental cost is the result of the new systems that are added. This upgrade path is typically more cost-effective than the path for vertical scaling. Overall, the scaling methods increase the transaction rates offered by the system. Using vertical scaling, the service identified by a particular IP address, protocol, and port number is under the control of the single device. In horizontal scaling, the service belongs to many service devices, which requires a mechanism to distribute the load.

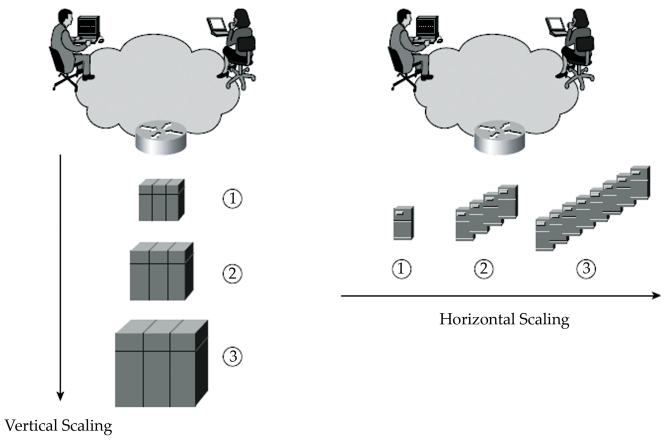
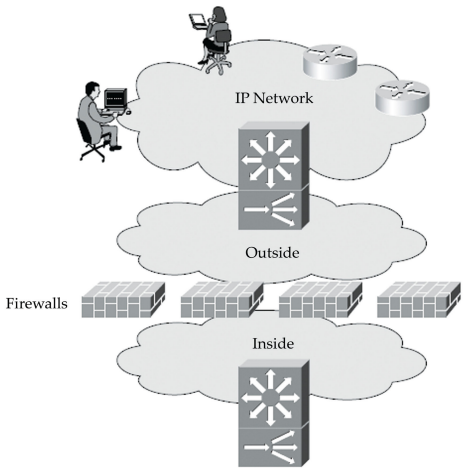


Figure 4: Scaling Methods

Before load balancers were available, the mechanism that was used to distribute the traffic load between servers was DNS round-robin. The DNS server, also known as the name server that was responsible for the domain used by servers supporting a specific service, would reply with a specific IP address. This IP address was one of a few in the available address records that matched the service name (host name) that the name server used in round-robin for each response. The round-robin process distributed the load to the different servers in the address records.

Firewall Load Balancing

Firewall load balancing typically scales firewall services at the edge of the enterprise network to the Internet. In a firewall load-balancing setup, the firewalls are placed between two layers of load balancers. However, unlike with VPN/IPSec load balancing, firewall load balancing deals with whichever protocols the firewalls deal with; depending on the **network location**, they can be exposed to a wide range of protocols in the TCP/IP protocol suite. This might require the load balancers to understand more than just the most commonly used protocols or to develop a mechanism to ensure the proper selection of the firewall for both inbound and outbound traffic.



Firewalls can operate in Layer 3 mode or Layer 2 mode. In Layer 3 mode, firewalls route packets between subnets. In Layer 2 mode, firewalls bridge packets, which implies that they are transparent to the network operation. To function properly,

load balancers used for firewall load balancing are required to understand both firewall operation modes. Additionally, firewalls can exchange connection information between themselves, which requires that the load balancer avoid impacting their exchange. The benefits of firewall load balancing are marginal and often offset by the complexity introduced by the design.

This does not mean that firewall load balancing is not useful, but it does mean that the requirements and benefits must be clearly understood before you decide to implement and support a firewall load-balancing environment. Follow these simple guidelines to make sure that you reap the benefits of firewall load balancing:

- Firewall load balancing is a viable option if you cannot achieve the desired performance with faster firewalls. It is much simpler to deploy a faster and more capable firewall to fix a firewall-scaling problem than to deploy a complex firewall load balancing solution.
- Make sure that the load balancer is capable of achieving the performance required by the aggregate number of firewalls, and allow some room for growth.
- Make sure that the load balancer is capable of dealing with the mode of operation of the firewalls and that it does not diminish the high-availability behavior of the firewalls.
- If the installation is so complex that only you understand it, there is most likely a simpler installation possible.

NETWORKING BASICS

Server Load Balancing works its magic in the networking realm. While this book assumes that the reader has experience in networking, it may be beneficial to cover some common networking terms and concepts and their relation to SLB. IP Networks with Cisco Routers by Scott M. Ballew provides a good general review of basic networking concepts and strategies.

"Cloud computing is a great euphemism for centralization of computer services under one server."

—Evgeny Morozov

OSI Layer Model

When referring to load balancers, OSI layers are often mentioned. OSI was developed as a framework for developing protocols and applications that could interact seamlessly. It closely resembles the Internet IP world in which load balancers exist today. The OSI model is broken into seven layers and is appropriately referred to as the 7-Layer Model. Each layer represents a separate abstraction layer and interacts only with its adjoining layers:

Layer 1

This is the lowest layer, often referred to as the "Physical" layer. The basic units of data, 1s and 0s, are transmitted on this layer electronically, such as with amplitude modulation on an Ethernet line or a Radio Frequency (RF) signal on a coaxial connection.

Layer 2

This layer refers to the method of organizing and encapsulating binary information for transport over a Layer 1 medium. Since SLB devices are almost always exclusively Ethernet-based, Layer 2 refers to Ethernet frames. An Ethernet frame consists of a header, a checksum (for error-correction), and a payload. Ethernet frames range in size, usually with a limit (known as Maximum Transmittable Units, or MTUs) of 1.5 KB for Ethernet, Fast Ethernet, and **Gigabit Ethernet**. Some devices support Jumbo Frames for Gigabit Ethernet, which is over 9,000 bytes.

Layer 3

Layer 3 devices are routers, which represent the level of information moved from one location to another in an intelligent manner (hence the clever name, router). IPv4 is the current standard for which Layer 3 IP packets are structured. An IP packet has a source IP address and a destination IP address in the header.

Layer 4

This layer deals with an IP address and a port. TCP and UDP are two protocols that run on this layer. They have a source

and destination IP address in the header, as well as a source and destination port. The payload is an encapsulated IP packet.

Layers 5- 7

Layers 5-7 involve URL load balancing and parsing. The URL may be complete (such as http://www.vegan.net/home) or may be a cookie embedded into a user session. An example of URL load balancing is directing traffic to http:// www.vegan.net/cgi-bin through one group of servers, while sending http:// www.vegan.net/images to another group. Also, URL load balancing can set persistence based on the "cookie" negotiated between the client and the server.

Server Load Balancers

A load balancer is a device that distributes load among several machines. As discussed earlier, it has the effect of making several machines appear as one. There are several components of SLB devices.

VIPs

Virtual IP (VIP) is the load-balancing instance where the world points its browsers to get to a site. A VIP has an IP address, which must be publicly available to be useable. Usually a TCP or UDP port number is associated with the VIP, such as TCP port 80 for web traffic. A VIP will have at least one real server assigned to it, to which it will dispense traffic. Usually there are multiple real servers, and the VIP will spread traffic among them using metrics and methods.

Servers

A server is a device running a service that shares the load among other services. A server typically refers to an HTTP server, although other or even multiple services would also be relevant. A server has an IP address and usually a TCP/UDP port associated with it and does not have to be publicly addressable (depending on the network topology).

Groups

While the term "group" is often used by vendors to indicate several different concepts, we will refer to it loosely as a group of servers being load balanced. The term "farm" or "server farm" would also be applicable to this concept.

User-Access Levels

A user-access level refers to the amount of control a particular user has when logged into a load balancer. Not only do different vendors refer to their access levels differently, but most employ very different access-level methods. The most popular is the Cisco style of user and enable (superuser) accounts. Another popular method is the Unix method of user-level access.

Read-only

A read-only access level is one in which no changes can be made. A read-only user can view settings, configurations, and so on, but can never make any changes. An account like this might be used to check the performance stats of a device. Read-only access is also usually the first level a user logs into before changing to a higher access-level mode.

Superuser

A superuser is the access level that grants the user full autonomy over the system. The superuser can add accounts, delete files, and configure any parameter on the system.

Other levels

Many products offer additional user levels that qualify somewhere between the access level of a superuser and a read-only user. Such an account might allow a user to change SLB parameters, but not system parameters.

Another level might allow configuration of Ethernet port settings, but nothing else. Vendors typically have unique methods for user-access levels.

Redundancy

Redundancy as a concept is simple: if one device should fail, another will take its place and function, with little or no impact on operations as a whole. There are several ways to achieve this functionality. Typically, two devices are implemented. A protocol is used by one device to check on its partner's health. In some scenarios, both devices are active and accept traffic, while in others, only one device is used while the other waits in case of failure.

Redundancy Roles

In redundancy, there is often an active-standby relationship. One unit, known as the active unit, takes on some or all of the functions, while another, the standby, waits to take on these functions. This is also often called the master/slave relationship. In certain scenarios, both units can be masters of some functions and slaves of others, in order to distribute the load. In other cases, both are masters of all functions, sharing between the two.

Active-Standby Scenario

The active-standby redundancy scenario is the easiest to understand and implement. One device takes the traffic while the other waits in case of failure (see Figure 5).

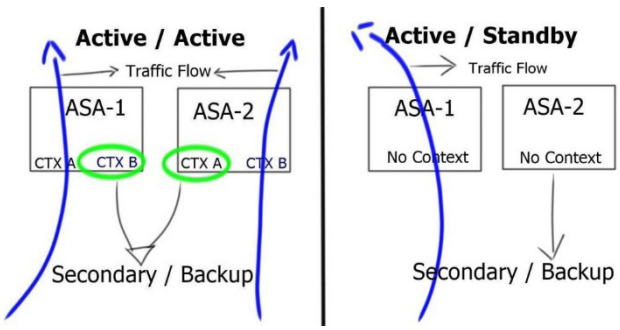


Figure 5: An active-standby redundancy scenario.

If the second unit were to fail, the other device would have some way of determining that failure and would take over the traffic.

Active-Active Scenario

There are several variations of the active-active scenario. In all cases, however, both units accept traffic. In the event of one of the devices failing, the other takes over the failed unit's functions. In one variation, VIPs are distributed between the two load balancers to share the incoming traffic. VIP 1 goes to Load Balancer A and VIP 2 to Load Balancer B (see Figure 6).

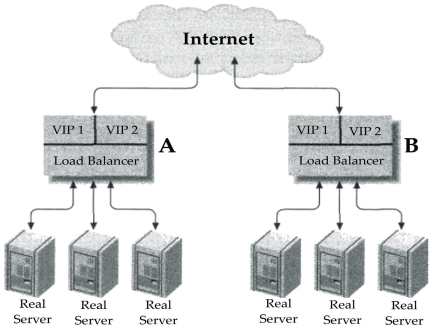


Figure 6: An active-active redundancy scenario.

In another variation, both VIPs answer on both load balancers, with a protocol circumventing the restriction that two load balancers may not hold the same IP address (see Figure 7).

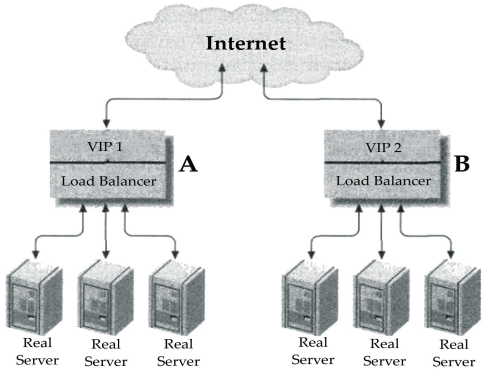


Figure 7: An active-active redundancy scenario variation.

As in all active-active scenarios, if one load balancer should fail, the VIP(s) will continue to answer on the remaining one. The other unit takes over all functions (see Figure 8).

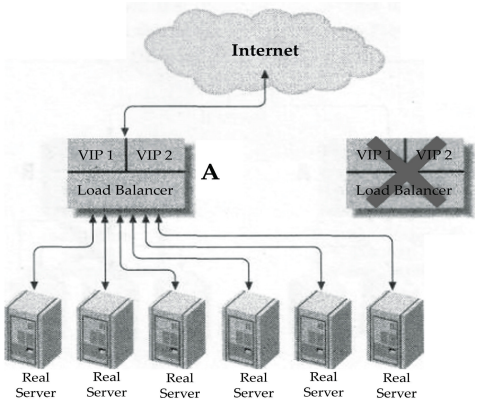


Figure 8: An active-active failure-recovery scenario.

VRRP

Perhaps the most common redundancy protocol is the Virtual Router Redundancy Protocol (VRRP). It is an open standard, and devices claiming VRRP support conform to the specifications laid out in RFC 2338.

Each unit in a pair sends out packets to see if the other will respond. If the sending unit does not get a response from its partner, then the unit assumes that its partner is disabled and initiates taking over its functions, if any.

While it's not necessary to know the inner workings of the VRRP protocol, some details may come in handy. VRRP uses UDP port 1985 and sends packets to the multicast address 225.0.0.2. These details are useful when dealing with some types of IP-filtering or firewalling devices, VRRP requires that the two units are able to communicate with each other. Should the two units become isolated from one another, each will assume the other unit is dead and take on "master" status. This circumstance can cause serious network problems because of IP-address conflicts and other network issues that occur when two units think they are both the active units in an active-standby situation.

xxRP

There are several proprietary versions of VRRP, each usually ending in "RP." Two examples are Extreme Network's Extreme Standby Router Protocol (ESRP) and Cisco's **Hot Standby Routing**

Protocol (HSRP). While these protocols vary slightly from the standard, they all behave in essentially the same manner.

Fail-Over Cable

Another method for detecting unit failure between a pair of devices is provided by the fail-over cable. This method uses a proprietary "heartbeat" checking protocol running over a serial line between a pair of load balancers. If this fail-over cable is disconnected, it causes both units to believe they are the only units available, and each takes on "master" status. This, as with the VRRP scenario, can cause serious network problems. Spanning-Tree Protocol (STP) is a protocol for Layer 2 redundancy that avoids bridging loops. STP sets a priority for a given port, and when multiple paths exist for traffic, only the highest-priority port is left active, with the rest being administratively shut down.

Stateful Fail-Over

One of the issues that a fail-over scenario presents (the "little" in little or no impact on network operations, as stated earlier) is if a device fails over, all of the active TCP connections are reset, and TCP sequence number information is lost, which results in a network error displayed on your browser. Also, if you are employing some form of persistence, that information will be reset as well (a bad scenario for a web-store type application). Some vendors have employed a feature known as "stateful fail-over," which keeps session and persistence information on both the active and standby unit. If the active unit fails, then the standby unit will have all of the information, and service will be completely uninterrupted. If done correctly, the end user will notice nothing.

Persistence

Persistence is the act of keeping a specific user's traffic going to the same server that was initially hit when the site was contacted. While the SLB device may have several machines to choose from, it will always keep a particular user's traffic going to the same server. This is especially important in web-store type applications, where a user fills a shopping cart, and

that information may only be stored on one particular machine. There are several ways to implement persistence, each with their advantages and drawbacks.

Service Checking

One of the tasks of an SLB device is to recognize when a server or service is down and take that server out of rotation. Also known as health checking, this can be performed a number of ways. It can be something as simple as a ping check, a port check, (to see if port 80 is answering), or even a content check, in which the web server is queried for a specific response. An SLB device will continuously run these service checks, usually at user-definable intervals.

Provider Infrastructure

The networking infrastructure is composed of the networking components that give connectivity to the Internet, Extranet, or Intranet for your web servers. It connects them to the users of your services. This is usually done one of two ways: in a location controlled by the site or in a location maintained by a colocation/ hosting provider that specializes in hosting other companies' server infrastructures.

Data Center

Whether your site is housed internally or at a colocation provider, your equipment is usually housed in type of space called a data center. Data center is a fairly general term, but it usually refers to an area with high security, environmental controls (usually air conditioning), non-water-based fire suppression, and UPS power backup systems with generators on standby, among other things. Money is probably the determining factor in the level and quality of the data center environment, from Fort Knox conditions to (literally) someone's basement.

Leased Line

In a leased-line scenario, a site is housed internally with one or more leased-line connections from one or more providers. It

can be as simple as a DSL line or as complicated as multiple OC3s running full BGP sessions to multiple providers. The advantage of this is that you have full control and access over your equipment. In Figure 9, we see a common leased-line scenario, where one location is connected to the Internet via two DS-3 (45 Mbps) lines from two separate providers. The site would probably run BGP on both lines, which is a protocol that allows redundancy in case a line from one provider goes down.

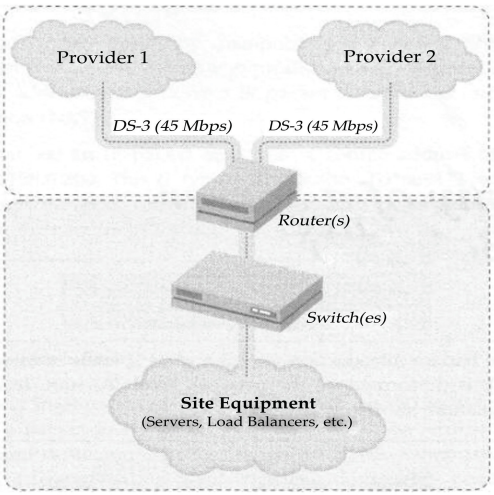


Figure 9: A leased-line scenario.

Colocation

Colocation is when you take your servers and equipment to a provider's location and house all your equipment there. Usually in racks or secure cages, your equipment sits on the colocation provider's property and is subject to its security, power, bandwidth, and environmental controls. The colocation provider typically provides the bandwidth through his or her own connectivity/backbone through a "network drop," usually an Ethernet connection (or multiple connections for redundancy). The advantage to colocation is that a colocation provider's bandwidth is usually more scalable than what you would have at your own facility. When you want more bandwidth from a colocation provider, you just take it, or upgrade your Ethernet lines, which don't take long to procure (a couple of days, depending on your provider). If you have leased lines to your

own facility, it can take anywhere from 30 days to 6 months to get telco companies to add more bandwidth, such as T-l (1.5 Mbps) or DS-3 (45 Mbps). Higher capacity lines usually take even longer. Colocation is the typical route taken nowadays, mostly because of cost and scalability concerns. It's just easier and cheaper in most situations to let another company worry about the data center and connectivity. Its network connectivity is usually very complex, involving peering points, leased lines to other providers, sometimes even its own backbone. Usually, all a hosted site need concern itself with is the network drop from the provider.

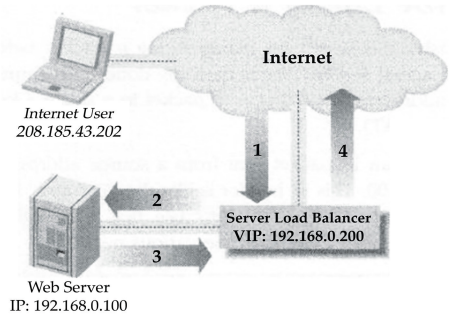
ANATOMY OF A SERVER LOAD BALANCER

SLB works by manipulating a packet before and (usually) after it reaches an actual server. This is typically done by manipulating the source or destination IP addresses of a Layer 3 IP packet in a process known as **Network Address Translation (NAT)**. An IP packet sent from a source address of 208.185.43.202 destined for 192.168.0.200. This IP header is like the "To" and "From" portions of a letter sent through the post office. Routers use that information to forward the packets along on their journeys through the various networks.

"I don't need a hard disk in my computer if I can get to the server faster ... carrying around these non-connected computers is byzantine by comparison."

—Steve Jobs

One issue of great importance in dealing with SLB, and TCP/IP in general, is that when you send a packet to an IP, the packet needs to be returned with the same source address and destination address. In other words, when you send a packet to a destination, it has to be sent back from that destination, not from another IP altogether. If the packet does not come from the IP address it was sent to, the packet is dropped. This is not an issue with UDP-based packets, since UDP is a connectionless protocol. However, most SLB implementations involve web serving, which is TCP-based. A user inputs a URL, which translates into the VIP address of 192.168.0.200. The packet traverses the Internet with a source IP address of 208. 185.43.202 and a destination address of 192.168.0.200.



The load balancer takes the packet destined for 192.168.0.200 and, instead of responding to the request, rewrites the packet. For the packet to get to the real server, the destination address is rewritten with the address of the real server. The source in step 2 is 208.185.43.202, and the destination is 192.168.0.100. The real server responds to the packet, and it is sent back to the user.

In step 3, the source becomes 192.168.0.100, and the destination becomes 208.185. 43.202, which presents a problem. The user will ignore a response from the IP address of 192.168.0.100, since the user never initiated a connection to that address; the user sent a packet to 192.168.0.200. The SLB unit solves this problem by being the default route of the real server and rewriting the packet before it is sent. The source address is rewritten to be the VIP, or 192.168.0.200.

The source in step 4 is 192.168.0.200 and the destination is 208.185.43.202. With that final rewrite, the packet completes its journey (see Table 1). To the user, it all seems like there is just one server, when in reality there could be several, even hundreds of real servers.

Table 1: SLB traffic manipulation

Step	Source IP	Destination IP
1	208.185.43.202	192.168.0.200
2	208.185.43.202	192.168.0.100
3	192.168.0.100	208.185.43.202
4	192.168.0.200	208.185.43.202

Direct Server Return

Direct Server Return (DSR) is a method of bypassing the load balancer on the outbound connection. This can increase the performance of the load balancer by significantly reducing the amount of traffic running through the device and its packet rewrite processes. DSR does this by skipping step 3. It tricks a real server into sending out a packet with the source address already rewritten to the address of the VIP (in this case, 192. 168.0.200). DSR accomplishes this by manipulating packets on the Layer 2 level to perform SLB. This is done through a process known as MAC Address Translation (MAT). To understand this process and how DSR works, let's take a look at some of the characteristics of Layer 2 packets and their relation to SLB.

MAC addresses are Layer 2 Ethernet hardware addresses assigned to every Ethernet network interface when they are manufactured. With the exception of redundancy scenarios, MAC addresses are generally unique and do not change at all with a given device. On an Ethernet network, MAC addresses guide IP packets to the correct physical device. They are just another layer of the abstraction of network workings.

DSR uses a combination of MAT and special real-server configuration to perform SLB without going through the load balancer on the way out. A real server is configured with an IP address, as it would normally be, but it is also given the IP address of the VIP. Normally you cannot have two machines on a network with the same IP address because two MAC addresses can't bind the same IP address. To get around this, instead of binding the VIP address to the network interface, it is bound to the loopback interface.

A loopback interface is a pseudointerface used for the internal communications of a server and is usually of no consequence to the configuration and utilization of a server. The loopback interface's universal IP address is 127.0.0.1. However, in the same way that you can give a regular interface multiple IP addresses (also known as IP aliases), loopback interfaces can be given IP aliases too. By having the VIP address configured on the loopback interface, we get around the problem of not having more than one machine configured with the same IP on a network. Since the VIP address is on the loopback interface,

there is no conflict with other servers as it is not actually on a physical Ethernet network.

In a regular SLB situation, the web server or other service is configured to bind itself to the VIP address on the loopback interface, rather than to a real IP address. The next step is to actually get traffic to this nonreal VIP interface. This is where MAT comes in. As said before, every Ethernet-networked machine has a MAC address to identify itself on the Ethernet network. The load balancer takes the traffic on the VIP, and instead of changing the destination IP address to that of the real server (step 2 in Table 1), DSR uses MAT to translate the destination MAC address. The real server would normally drop the traffic since it doesn't have the VIP's IP address, but because the VIP address is configured on the loopback interface, we trick the server into accepting the traffic. The beauty of this process is that when the server responds and sends the traffic back out, the destination address is already that of the VIP, thus skipping step 3 of Table 1, and sending the traffic unabated directly to the client's IP.

Table 2: The DSR process

Step	Source IP	Destination IP	MAC Address
1	208.185.43.202	192.168.0.200	Destination:00:00:00:00:00:aa
	208.185.43.202	192.168.0.200	Destination:00:00:00:00:00:bb
2	192.168.0.200	208.185.43.202	Source: 00:00:00:00:00:bb
3			

Included in this table are the MAC addresses of both the load balancer (00:00:00: 00:00:aa) and the real server (00:00:00:00:00:bb).

As with the regular SLB example, 192.168.0.200 represents the site to which the user wants to go, and is typed into the browser. A packet traverses the Internet with a source IP address of 208.185.43.202 and a destination address of the VIP on the load balancer. When the packet gets to the LAN that the load balancer is connected to, it is sent to 192.168.0.200 with a MAC address of 00:00:00:00:aa.

In step 2, only the MAC address is rewritten to become the MAC address that the real server has, which is 00:00:00:00:00:bb. The server is tricked into accepting the packet and is processed

by the VIP address configured on the loopback interface. In step 3, the traffic is sent out to the Internet and to the user with the source address of the VIP, with no need to send it through the load balancer. Figure 10 shows the same process in a simplified diagram.

Web traffic has a ratio of about 1:8, which is one packet out for every eight packets in. If DSR is implemented, the workload of the load balancer can be reduced by a factor of 8. With streaming or download traffic, this ratio is even higher. There can easily be 200 or more packets outbound for every packet in, thus DSR can significantly reduce the amount of traffic with which the load balancer must contend.

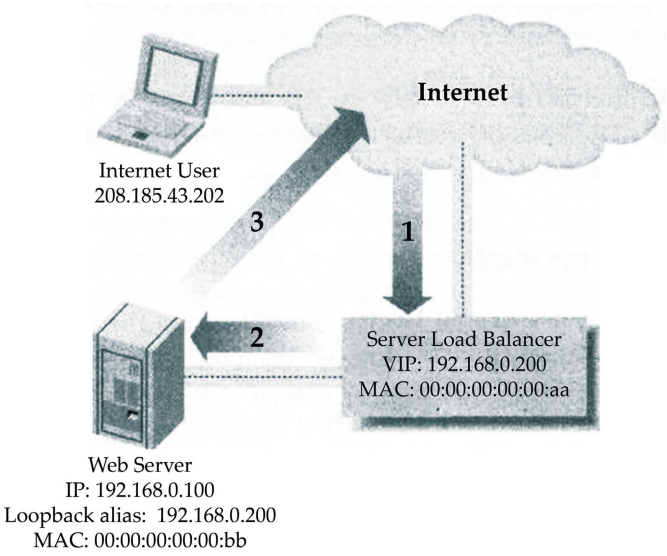


Figure 10: The DSR traffic path.

The disadvantage to this process is that it is not always a possibility. The process requires some fairly interesting configurations on the part of the real servers and the server software running on them. These special configurations may not be possible with all operating systems and server software. This process also adds complexity to a configuration, and added complexity can make a network architecture more difficult to implement. Also, any Layer 5-7 URL parsing or hashing would not work because that process requires a synchronous data path in and out of the load balancer.

SLB Methods

There are several other ways to perform network-based SLB. The way it is normally implemented is sometimes called "half-NAT," since either the source address or the destination address of a packet is rewritten, but not both. A method known as "full-NAT" also exists. Full-NAT rewrites the source and destination addresses at the same time. A given scenario might look like the one in Table 3.

Table 3: Full-NAT SLB

Step	Source	Destination
1	208.185.43.202	192.168.0.200
2	10.0.0.1	10.0.0.100
3	10.0.0.100	10.0.0.1
4	192.168.0.200	208.185.43.202

All source addresses, regardless of where the requests come from, are set to one IP address. The downside to this is that full-NAT renders web logs useless, since all traffic from the web server's point of view comes from one IP address.

Under the Hood

SLB devices usually take one of two basic incarnations: the switch-based load balancer or the server-based load balancer. Each has its general advantages and drawbacks, but these greatly depend on how the vendor has implemented the technology.

Server-Based Load Balancers

Server-based load balancers are usually PC-based units running a standard operating system. Cisco's Local Director and F5's BIG-IP are both examples of server based load balancers. SLB functions are performed by software code running on top of the network stack of the server's OS. Generally, the OS is an OEMed version of a commercial OS such as BSDI or a modified freeware OS such as Linux or FreeBSD. In a load balancer such as Cisco's Local Director, the entire OS is written by the manufacturer.

Server-based load balancers are typically easy to develop for because the coding resources for a widely used OS are easy to come by. This can help shorten code and new-feature turnaround, but it can also be a hindrance. With shorter code cycles, bugs can become more prevalent. This easy development cycle means that sever-based load balancers are typically flexible in what they can do. New features can be rolled out swiftly, and the machines themselves can take on new and creative ways of performance monitoring, as well as other tasks.

Switch-Based Load Balancers

Switch-based load balancers, also known as hardware-based load balancers, are devices that rely on Application Specific Integrated Circuit (ASIC) chips to perform the packet-rewriting functions. ASIC chips are much more specialized processors than their Pentium or PowerPC cousins. Pentium and PowerPC chips have a general instruction set to them, which enables a wide variety of software to be run, such as Quake III or Microsoft Word. An ASIC chip is a processor that removes several layers of abstraction from a task. Because of this specialization, ASIC chips often perform software tasks much faster and more efficiently than a general processor. The drawback to this is that the chips are very inflexible. If a new task is needed, then a new ASIC design may have to be built. However, the IP protocol has remained unchanged, so it's possible to burn those functions into an ASIC.

Switch-based load balancers are typically more difficult to develop code for. They often run on proprietary architectures, or at least architectures with minimal development resources. Therefore, code comes out slower but is more stable. The switch-based products are also usually faster. Their ASIC chips are more efficient than software alone. Typically, they also have internal-bandwidth backbones capable of handling a Gbps worth of traffic. PCs are geared more toward general I/O traffic and are not optimized for IP or packet traffic.

Connections Per Second

As far as pure performance goes, this is probably the most important metric, especially with HTTP. Connections per second relates to the number of incoming connections an SLB device

accepts in a given second. This is sometimes referred to as transactions per second or sessions per second, depending on the vendor. It is usually the limiting factor on any device, the first of any of the metrics to hit a performance limit. The reason this metric is so critical is that opening and closing HTTP connections is very burdensome on a network stack or network processor. Lets take a simplified look at the steps necessary to transfer one file via HTTP:

- The client box initiates an HTTP connection by sending a TCP SYN packet destined for port 80 to the web server.
- The web server sends an ACK packet back to the client along with an additional SYN packet.
- The client sends back an ACK packet in response to the server's SYN request.

The beginning of a connection is known as the "three-way handshake." After the handshake is negotiated, data can pass back and forth. In the case of HTTP, this is usually a web page. Now this process has quite a few steps for sending only 30 KB worth of data, and it strains a network device's resources. Setting up and tearing down connections is resource-intensive. This is why the rate at which a device can accomplish this is so critical.

Total Concurrent Connections

Total concurrent connections is the metric for determining how many open TCP user sessions an SLB device can support. Usually, this number is limited by the available memory in an SLB device's kernel or network processor.

The number ranges from infinity to only a few thousand, depending on the product. Most of the time, however, the limit is theoretical, and you would most likely hit another performance barrier before encountering the total available session number. For UDP traffic, concurrent connections are not a factor, as UDP is a completely connectionless protocol.

UDP traffic is typically associated with either streaming media or DNS traffic, although there are several other protocols that run on UDP. Most load balancers are capable of handling UDP protocols for SLB.

Throughput

Throughput is another important metric. Typically measured in bits per second, throughput is the rate at which an SLB device is able to pass traffic through its internal infrastructure. All devices have internal limiting factors based on architectural design, so it's important to know the throughput when looking for an SLB vendor. For instance, a few SLB vendors only support Fast Ethernet, thus limiting them to 100 Mbps (Megabits per second). In addition, some server-based products may not have processors and/or code fast enough to handle transfer rates over 80 Mbps. While throughput is measured in bits per second, it is actually a combination of two different variables: packet size and packets per second. Ethernet packets vary in length, with a typical Maximum Transmittable Unit (MTU) of about 1.5 KB. If a particular piece of data is larger than 1.5 KB, then it is chopped up into 1.5 KB chunks for transport. The number of packets per second is really the most important factor a load balancer or any network device uses. The combination of this and packet size determines the bits per second. For example, an HTTP GET on a 100-byte text file will fit into one packet very easily. An HTTP GET on a 32 KB image file will result in the file being chopped into about 21 Ethernet packets, but each would have a full 1.5 KB payload. The bigger the payload, the more efficient use of resources. This is one of the main reasons why connections per second is such an important metric. Not only do connections per second cause quite a bit of overhead on just the initiation of a connection, but sites that experience high rates of connections per second typically have small payloads. Throughput can be calculated as follows:

- Throughput = packet transmission rate x payload size

The 100 Mbps Barrier

Many SLB models are equipped with only Fast Ethernet interfaces, thus limiting the total throughput to 100 Mbps. While most users aren't necessarily concerned with pushing hundreds of Megs worth of traffic, many are concerned that while they push 50 Mbps today, they should be able to push 105 Mbps in the future. To get around this, there are a couple of techniques available. One technique involves Fast EtherChannel, which binds two or more Fast Ethernet links into one link, combining the available

bandwidth. This isn't the simplest solution by far, and there are limits to how Fast EtherChannel distributes traffic, such as when one portion of the link is flooded while another link is unused. Another solution is the Since DSR does not involve the outbound traffic passing the SLB device, which is typically the majority of a site's traffic, the throughput requirements of an SLB device are far less. At that point, the limiting factor would become the overall connectivity of the site.

The simplest solution to this problem is using Gigabit Ethernet (GigE) on the load balancers. The costs of GigE are dropping to more affordable levels, and it's a great way to aggregate large amounts of traffic to Fast Ethernet-connected servers. Since the limit is 1 Gbps (Gigabit per second), there is plenty of room to grow a 90 Mbps site into a 190 Mbps site and beyond. Getting beyond 1 Gbps is a challenge that future SLB products will face.

Traffic Profiles

Each site's traffic characteristics are different, but there are some patterns and similarities that many sites do share. There are three typical traffic patterns HTTP, FTP/Streaming, and web store traffic seem to be fairly typical as far as traffic patterns go. Table 4 lists these patterns and their accompanying metrics. Of course, the traffic pattern for your site may be much different. It is critical to identify the type or types of traffic your sites generate to better design your site, secure your site, and tune its performance.

Table 4: The metrics matrix

Traffic pattern	Most important metric	Second most important metric	Least important metric
HTTP	Connections per second	Throughput	Total sustained
FTP/Streaming	Throughput	Total sustained	connections
Web store	Total sustained connections	connections	Connections per second
		Connections per	Throughput
		second	

HTTP

HTTP traffic is generally bandwidth-intensive, though it generates a large amount of connections per second. With HTTP 1.0, a TCP connection is opened for every object, whether it be an HTML file, an image file (such as a GIF or JPEG), or text file. A web page with 10 objects on it would require 10 separate TCP connections to complete. The HTTP 1.1 standard makes things a little more efficient by making one connection to retrieve several objects during a given session. Those 10 objects on the example page would be downloaded in one continuous TCP connection, greatly reducing the work the load balancer and web server would need to do. HTTP is still fairly inefficient as far as protocols go, however. Web pages and their objects are typically kept small to keep download times small, usually with a 56K modem user in mind (a user will likely leave your site if the downloads take too long). So web pages generally don't contain much more than 70 or 80 KB worth of data in them. Now, that number greatly varies depending on the site, but it is still a relatively small amount of data.

FTP/Streaming

FTP and streaming traffic are very similar in their effects on networks. Both involve one initial connection (or in the case of streaming, which often employs UDP, no connection) and a large amount of data transferred. The rate of FTP/streaming initial connections will always remain relatively small compared to the amount of data transferred. One FTP connection could easily involve a download of a Megabyte or more worth of data. This can saturate networks, and the 100 Mbps limit is usually the one to watch.

Web Stores

Web stores are where the money is made on a site. This is the money that usually pays the bills for the network equipment, load balancers, and salaries (and also this book!), so this traffic must be handled with special care. Speed is of the utmost importance for this type of traffic; users are less likely to spend money on sites that are too slow for them. This type of traffic does not generally involve a large amount of bandwidth, nor does it

involve a large amount of connections per second (unless there is a media-related event, such as a TV commercial). Sustained connections are important, though, considering that a site wants to support as many customers as possible.

GENERIC LOAD BALANCER ARCHITECTURE

The load-balancing overview cannot be complete without the description of the architecture of a load balancer. This is an important topic because load balancers provide a new set of application services that require some special software and hardware features to make the processing possible at reasonable rates.

"Traditional PCs face competition from specialty products like Palm Pilots and from the servers that provide the nodes in computer networks. Microsoft's Windows CE hasn't done too well in the specialty-device market, and its Windows NT faces strong competition for server customers."

—Virginia Postrel

The generic architecture that support fundamental processing functions in the load balancer. Some of these components can be combined on specific hardware architectures, yet the functions remain constant. Figure 11 presents a generic architecture used on a load balancer.

The main components of the architecture are as follows:

- Switch fabric
- Main processor or CPU
- LB processors
- Control bus
- Interface ASICs
- Memory components

The switch fabric provides the connectivity between the interfaces, the LB processors, and the main CPU. The main processor performs the control functions, such as configuration and management of the components and processing of control traffic. It uses the control bus to transmit and receive control information to and from the LB processors. The LB processors perform the different intelligent operations on the packets, such

as pattern matching, TCP spoofing, and so on. When you have multiple processors, each is specialized for a certain task, yet they all complement each other. The ASICs automate the packet rewrite functions, which offload the other processors from the burden. Memory is used throughout for different purposes. The main CPU uses memory for connection and session tables, the LB processors for specific connection information, and the ASICs at the interface level for connection information associated with the particular interface.

An example of the operation of this architecture is as follows:

- The client sends a new connection request to VIPA that is supported by lb1 (load balancer 1) in Figure 11.
- lb1 receives the packet on a specific port p1.
- A lookup process using the 5-tuple (source/destination IP, source/destination port, and protocol number) takes place to determine whether the packet is part of an existing connection or a new connection. In the latter case, the packet is a connection request (SYN), so the lookup process comes back empty.
- The packet is forwarded to the main CPU for processing. The main CPU processing steps are as follows:
  - Selection of the content policies that should be applied to the packet
  - Application by the lb processors of the appropriate policies to the packet
  - Selection of the real server by the main CPU
  - Creation of a new entry in the connection table (MEM) and other potential memory locations (interface MEM) for the specific connection
- After a server is selected, the header re-write operations take place (interface ASIC), and the packet is forwarded to the selected real server.
- The interface memory is updated with the header re-write information to be used in subsequent packets.

When a packet that belongs to an existing connection arrives, policies are applied to it. The packet then is forwarded to the appropriate interface based on the connection table, and the interface ASIC performs the header rewrite tasks before

forwarding the packet. Depending on the hardware architecture, there are variations to where the processing takes place. In certain cases, ASICs at the port level help the processing by keeping track of connection information so that they can rewrite header information before the packet is forwarded, thereby offloading the main CPU from processing.

A common approach to offloading the main CPU is to add more processors that share the connection-processing load. The load sharing is done either by assigning connections to processors or by assigning different tasks to each processor. The tasks per processor then are executed sequentially on each packet as the packet moves through the processors. Because the processors are organized in pipeline fashion, they also execute their precise tasks concurrently, yet on the different packets moving through the pipeline.

The forwarding operation can take advantage of a switch fabric if it is present, or it can just happen on a bus in older architectures. If a fabric is present, the logic to perform the rewrite most likely is located on the ports. The alternative is to have central logic that rewrites the Layer 2 information to Layer 4 information.

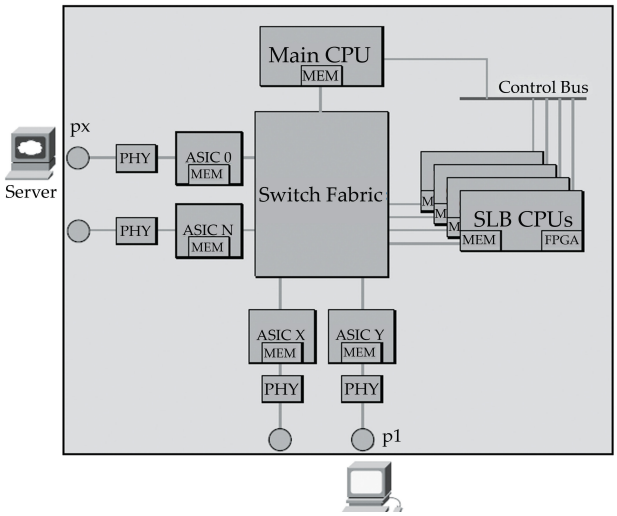


Figure 11: Generic Architecture of a Load Balancer.

Critical Components of a Load Balancer

- You could conclude that three critical areas are related to the performance of a load balancer, each with its own set of critical components: CPU processing capacity, memory, and switching capacity. These areas depend, in turn, on the critical components that support them.
- CPU processing capacity is used for control functions, which include but are not limited to connection creation and teardown, deep packet inspection, Layer 2 functions such as **Spanning Tree Protocol (STP)**, and Internal Gateway Protocols (IGPs).
- Memory holds packet and connection information for transit traffic. Transit traffic applied to load balancers when connection state information must be maintained increases the overall demand for memory utilization.
- Switch fabric allows transit traffic and other control messages to be communicated among multiple ports concurrently. In addition to each port capacity, the switch fabric capacity determines the overall raw bandwidth available.
- Each critical area is then bound to one or multiple components that are coupled tightly to the previously described connection types.

Table 5: *Critical Components for Load Balancing*

Performance Impact Areas	Critical Components
Processing capacity	Main or central CPU, port ASICs, dedicated processors
Memory	DRAM, port memory, CAM
Switching capacity	Shared bus, switch fabric, port capacity

Table 5 reflects the main areas in question and the critical components per area. The components listed are deemed critical because they determine the performance capacity of the load balancer. The performance capacity is quantified by performance metrics, and the performance metrics are related to the critical components.

The processing of both short-lived and long-lived connections is tightly coupled with the critical components. Before explaining this relationship, it is important to state the following:

- The critical components are finite resources.
- A real traffic mix is composed of a combination of short-lived and long-lived connections that are changing constantly.

The one assumption that can be made is that the traffic patterns for any one network include a combination of short-lived and long-lived connections that can exceed the capacity of a load balancer. The processing capacity limits are related to the critical components, which are the components of a load balancer that are worth monitoring.

IP Address Configuration: Flat-Based SLB Versus NAT-Based SLB

The flat-based SLB architecture shown in Figure 12, VIPs and nodes are on the same subnets. This can be done with either a bridging-path or route-path SLB method.

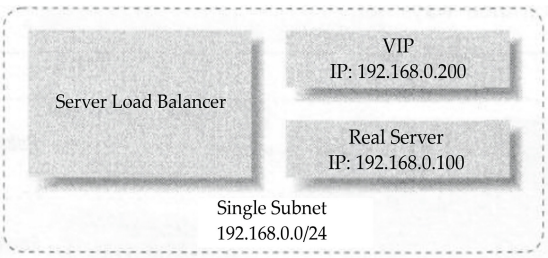


Figure 12. Flat-based SLB architecture.

With the NAT-based SLB architecture shown in Figure 13, the load balancer sits on two separate subnets and usually two different VLANs. The load balancer is the default gateway of the real servers, and therefore employs the route-path SLB method. Bridging-path SLB will not work with NAT-based SLB.

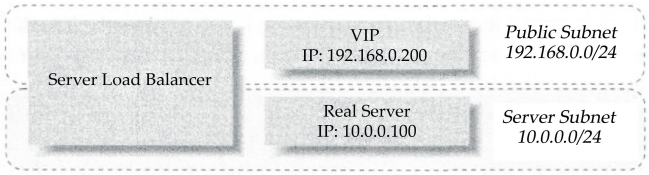


Figure 13: NAT-based SLB architecture.

Return Traffic Management: Bridging-Path versus Routing-Path versus DSR

Most of the configurations will use route-path SLB rather than bridgingpath. There are a number of reasons for this, first and foremost being that routepath is typically simpler to implement. In conjunction with the one-armed or twoarmed physical connectivity, any inherent topology and redundancy issues are much easier to resolve. Bridging-path SLB works only with a flat-based SLB implementation, while route-path SLB works with either flat-based or NAT-based SLB. With bridging-path SLB, you run into some deployment limitations. Since bridgingpath SLB works on the Layer 2 level, there can be only one Layer 2 path on which traffic flows, thus limiting your load-balancer installation to one redundant pair (one does not forward Layer 2 traffic as a standby unit). If there is more than one pair, there is more than one Layer 2 path, resulting in either a bridging loop (very bad) or Layer 2 devices on the network shutting off one or more of the load-balancer ports.

In Figure 14, you can see how bridging works with SLB. The load balancer acts as a Layer 2 bridge between two separate LANs. The packets must traverse the load balancer in and on their ways out.

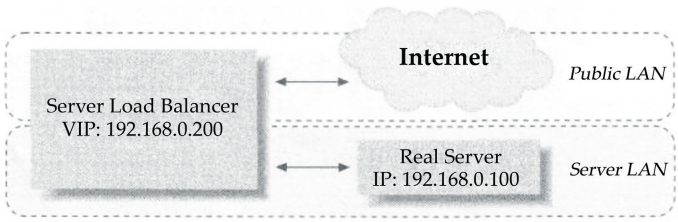


Figure 14: Bridging-path SLB architecture.

With route-path SLB (shown in Figure 15), the load balancer is the default route of the real servers. It works like a router by forwarding packets.

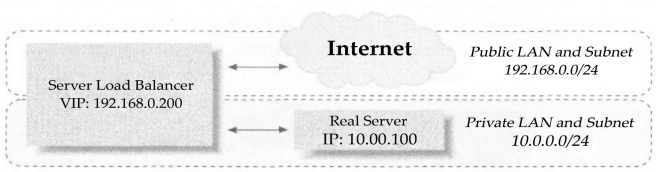


Figure 15: NAT-based, route-path SLB architecture.

One-Armed Versus Two-Armed

There are a number of possible configurations with flat-based and NAT-based SLB architectures. One-armed is used only with flat-based SLB and two-armed, only with NAT-based SLB. There are a number of reasons for this. With flat-based, using a one-armed method means using the route-path method. With NAT-based SLB, security is better served by using two separate VLANs for the outside and internal networks.

With a one-armed configuration (shown in Figure 16), there is only one connection from the load balancer to the network infrastructure. This is perfect for flat based SLB, since it involves real servers and VIPs on the same subnet and LAN. With NAT-based SLB, it is possible to have both the outside and internal networks on the same link, but that creates a security hazard and, thus, should be avoided.

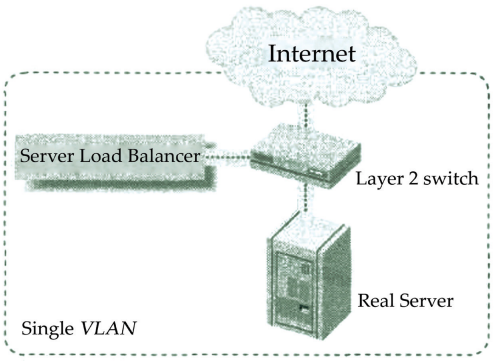


Figure 16: One-armed SLB configuration.

With a two-armed configuration (shown in Figure 17), two separate links connect to two separate VLANs on two different subnets. This is the perfect configuration for NAT-based SLB, as it provides two links on separate networks. This also works great when using NAT-based SLB as part of a security scheme.

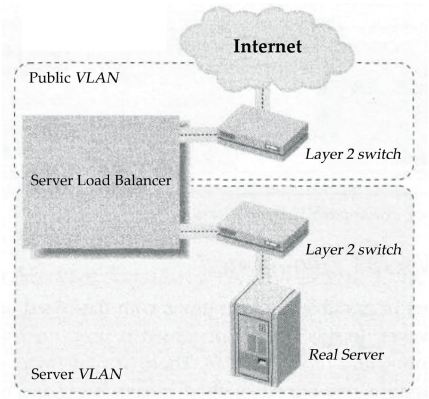


Figure 17: Two-armed SLB configuration.

Two-armed is also used for bridge-path topologies, since the load balancer bridges two separate LANs. It isn't possible to achieve a one-armed configuation with bridge-path, since the load balancer bridges between two separate LANs.

CONCLUSION

This chapter described various fundamental concepts of load balancing that aid in understanding how load balancers work, what they do, and how to design networks that use them. Load balancers are used in a variety of configurations to solve distinct problems that typically are related to scalability. They are deployed in front of farms of devices to increase the overall capacity of the farms while offering high availability and extended applicationaware features. Load balancers operate at multiple layers and perform critical functions on behalf of the server farm. The section, "Key Concepts of Load Balancing," introduced the details of how load balancers perform processing tasks at various stages and varying depth. The depth ranges from Layer 2 (MAC) through Layer 5 (applications such as HTTP), yet load balancing could operate at Layer 3 (IP) and Layer 4 (TCP/UDP) as well.

High availability is a major design consideration when deploying load balancers. The different options are driven by the application requirements. This means that careful consideration of design details leads to a better implementation. Finally, this chapter explained the processing details and architectural issues in a generic architecture for a load balancer. Understanding the components and their functions enables you to determine which of the critical components is likely to be affected first by the traffic load and also enables you to determine track and understand the implications of the traffic types on the load balancer, and vice versa.

KEY TERMS

IP address	DNS round robin
Network location	Gigabit Ethernet
Network infrastructure	Spanning Tree Protocol

DISCUSSION QUESTIONS

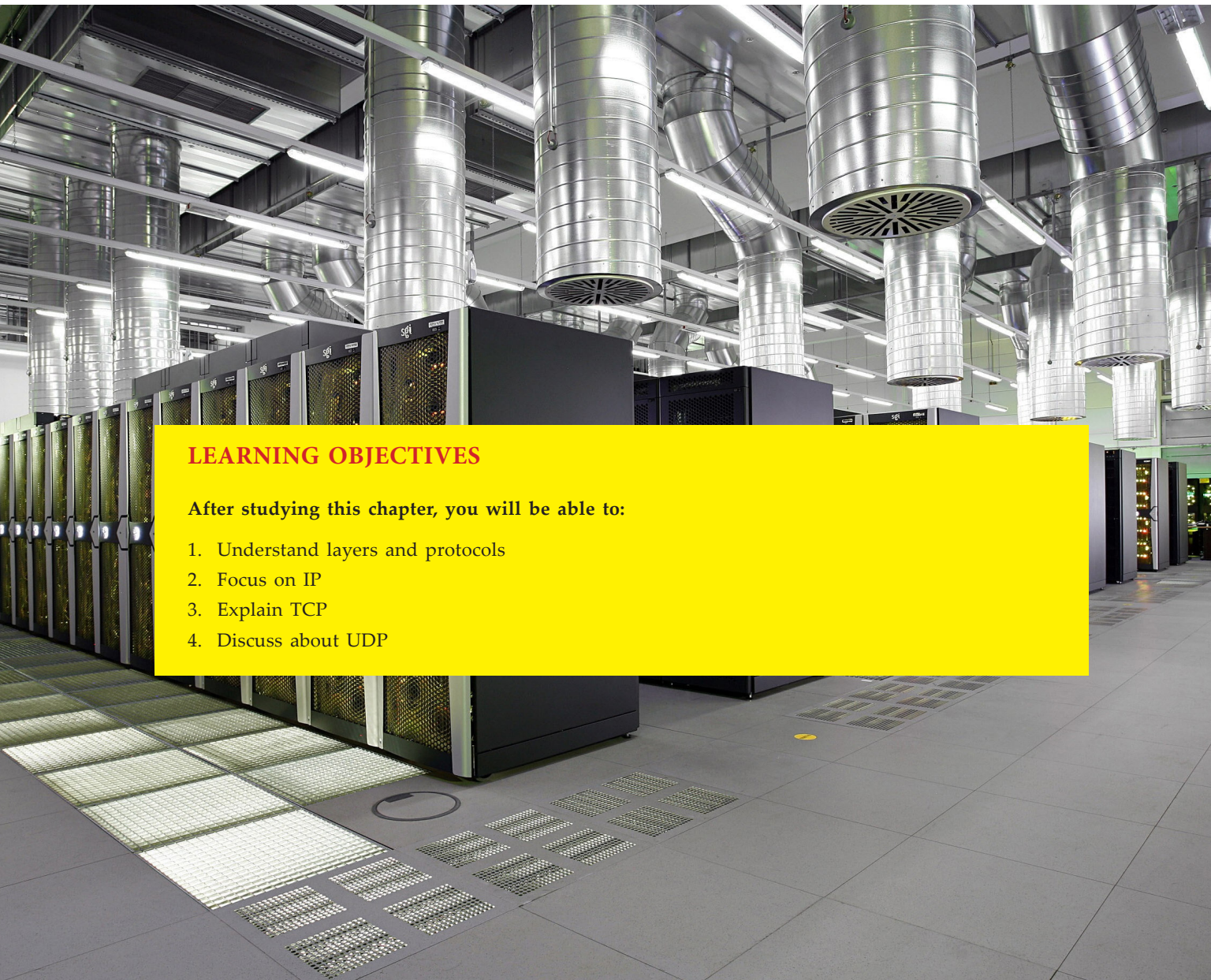
1. Describe concept of server load-balancing.
2. Briefly Explain the DNS-based load balancing.
3. Discuss about load balancing functions.
4. Define the OSI layer model.
5. What do you understand by VRRP?
6. Write the direct server return.
7. Evaluate the SLB methods.
8. What are difference between server and switch based load balancers?

REFERENCES

1. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. Symposium on Networked Systems Design and Implementation, San Francisco CA, Mar 2004.
2. C. Huitema and S. Weerahandi. Internet Measurements: the Rising Tide and the DNS Snag., ITC Specialist Seminar on Internet Traffic Measurement and Modeling, Monterey CA, Sep 2000.
3. Coulouris GF, Dollimore J, Kindberg T. Distributed systems: concepts and design. Pearson Education; 2005.
4. D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A Scalable and Dynamic Emulation of the Butterfly. Symposium on Principles of Distributed Computing, Monterey CA, Aug 2002.
5. E. Cohen and H. Kaplan. Proactive Caching of DNS Records: Addressing a Performance Bottleneck. Symposium on Applications and the Internet, San Diego-Mission Valley CA, Jan 2001.
6. F. Kaashoek and D. Karger. Koorde: A Simple Degree-Optimal Distributed Hash Table. International Workshop on Peer-To-Peer Systems Workshop, Berkeley CA, Feb 2003.
7. Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G. Palacios. Open, closed, and mixed networks of queues with different classes of customers.
8. Gupta, B. Liskov, and R. Rodrigues. Efficient Routing for Peer-to-Peer Overlays. Symposium on Networked Systems Design and Implementation, San Francisco CA, Mar 2004.
9. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead. International Workshop on Peer-To-Peer Systems, Berkeley CA, Feb 2003.
10. Internet Systems Consortium. BIND Vulnerabilities. www.isc.org/sw/bind/bind-security.php, Feb 2004.
11. J. Douceur, A. Adya, W. Bolosky, D. Simon, and M. Theimer. Reclaiming Space from Duplicate Files in a Serverless Distributed File System. International Conference on Distributed Computing Systems, Vienna Austria, Jul 2002.
12. J. Jung, A. Berger, and H. Balakrishnan. Modeling TTL-based Internet Caches. International Conference on Computer Communications, San Francisco CA, Mar 2003.

CHAPTER 7

SERVER FARM: IP, TCP, AND UDP



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understand layers and protocols
2. Focus on IP
3. Explain TCP
4. Discuss about UDP

INTRODUCTION

A server farm is a group of networked servers that are housed in one location. A server farm streamlines internal processes by distributing the workload between the individual components of the farm and expedites computing processes by harnessing the power of multiple servers.

This chapter introduces the fundamental protocols used in intranets and, in particular, the server farms supporting web-based services. A number of other networking devices, such as firewalls, load balancers, and Secure Socket Layer (SSL) offloading devices, are also exposed to the operation of these fundamental protocols. These protocols include Transport Control Protocol (TCP) and User Datagram Protocol (UDP) at Layer 4 and Hypertext Transfer Protocol (HTTP) at Layer 5, which combined make a substantial percentage of traffic destined to the server farms in the Data Center.

LAYERS AND PROTOCOLS

Open system interconnection (OSI) reference model consists of seven layers or seven steps which concludes the overall communication system.

"Plenty of crooks are happy to run a corporate phishing program for no up-front cost. The only expenses are ransomware fees if someone fails."

—G. Mark Hardy

Server load balancers, for example, operate at different layers, which are often named differently according to their operation. These labels include Layer 4, Layer 7, or Layers 5 through 7, referring to the layers of the packets in which the server load balancing is performed.

The classical model used to describe the layered approach to protocols is the Open System Interconnection (OSI) reference model, which is depicted in Figure 1.

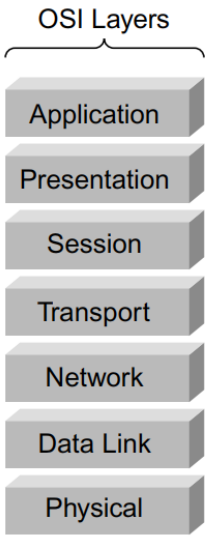


Figure 1. Layers in the OSI Model.

The layers in Figure 1 represent the typical layers required in a **protocol stack** to efficiently partition functional tasks in each layer while permitting layer-to-layer communication through clearly identified interfaces. When referring to Layer 5 or Layer 7, the general assumption is that the layers are those defined in the OSI model. Unfortunately, the OSI model does not completely map to existing protocol stacks such as TCP/IP. TCP/IP in fact does not use seven layers; thus, the concept of Layers 5 through 7 does not apply. Figure 2 presents the mapping from the OSI model to the TCP/IP protocol stack.

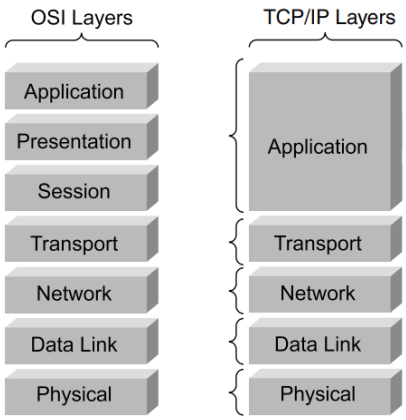


Figure 2. OSI to TCP/IP Layer Mappings.

As shown in Figure 2, the TCP/IP stack does not have Layer 6 or Layer 7. The application layer of the TCP/IP stack, Layer 5, corresponds to the session, presentation, and application layers of the OSI model. Applications that run on top of a TCP/IP stack must perform the tasks and functions that the OSI Layers 5 through 7 support, all in the same layer. When referring to Layers 5 through 7 and applications that run on TCP/IP, the implication is that the session, presentation, and application functions are distinct, although they are supported by the application layer. For example, a SQL request that uses TCP as the transport protocol could potentially have information from a user making the request. This information (user ID, such as a name or account number) is different from requests originated by a different user. From the perspective of the TCP/IP protocol stack, these different requests are simply considered part of the TCP payload, but the information they contain might be important to the application. The user ID is then considered a Layer 7 piece of information.

Figure 3 describes how the protocols in the TCP/IP stack map to the various layers of the TCP/IP stack.

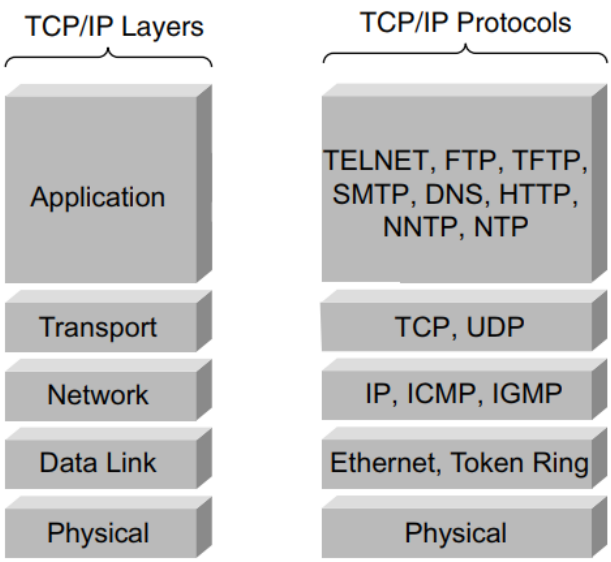


Figure 3. TCP/IP Protocols.

The TCP/IP protocol suite includes many protocols organized in layers. Each layer is responsible for a different function of the communication exchange.

The protocols at the transport and application layers that are commonly found in server farm environments and, in particular, those that make up the majority of any given traffic mix. These protocols include TCP and UDP at the transport layer and HTTP, FTP, DNS, RTP, FTP, and many others at the application layer.

Notice that the application layer protocols use the transport protocols TCP and UDP, which implies that the application layer protocols rely on the flow control and error correction and detection characteristics of the transport layer protocols if available. For example, HTTP uses TCP and relies on the connection-oriented nature of TCP. It implies that HTTP does not need to support flow control and error correction because TCP provides these capabilities.

Before delving into the details of transport protocols, it is important to understand some aspects of the IP protocol, how it encapsulates TCP and UDP, and the header details. Figure 4 introduces the encapsulation steps as data is sent from the application down the protocol stack.

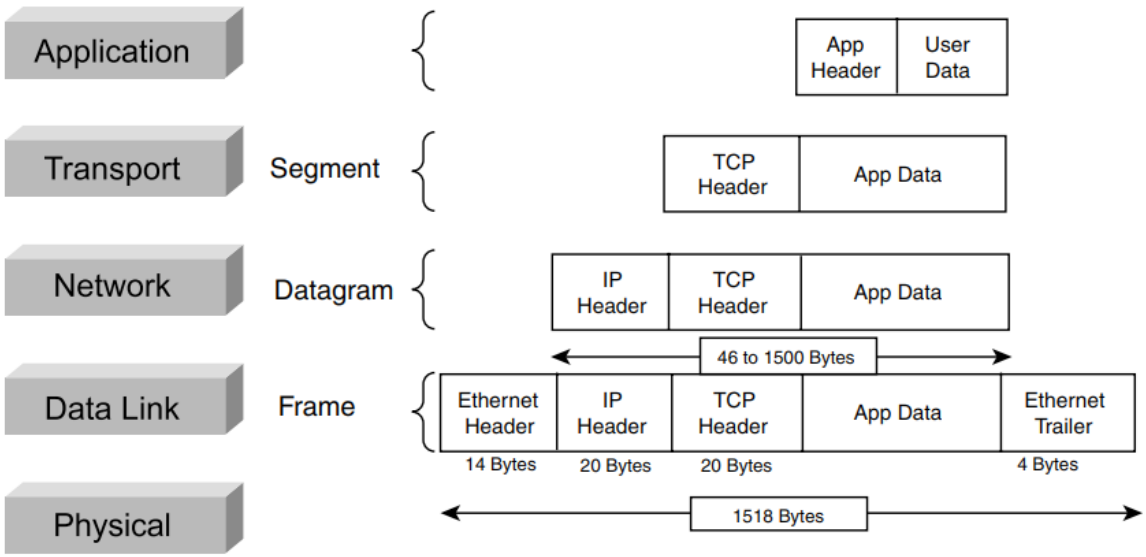


Figure 4. Encapsulation.

As application data comes down the protocol stack, it is encapsulated into the transport protocol payload (TCP or UDP), which in turn is encapsulated onto the network protocol (IP), which is placed into an **Ethernet frame**. As the application data is made available to the TCP/IP stack, the stream of bytes uses a specific name that denotes the protocol stack layer handling the stream. A TCP or UDP chunk of data is referred to as a segment, an IP chunk of data is a datagram, and an Ethernet chunk of data is a frame. Packet describes a chunk of data going from the IP layer to the network interface (data link layer); however, it is not exactly the same as a datagram. As defined in Stevens, "A packet can be either an IP datagram or a fragment of an IP datagram."

IP

The original specification of IP is available in RFC 0791. Like many other protocols in the TCP/IP suite, IP has been evolving. The latest IP specification, IP version 6 (IPv6), also known as IP Next Generation (IPng), is available in RFC 2460, which is part of a long list of specifications related to IPV6. This list, controlled by the IETF IPv6 working group.

"Securing a computer system has traditionally been a battle of wits: the penetrator tries to find the holes, and the designer tries to close them."

—Gosser

As previously shown in Figure 3, IP operates at the network layer, and it is a connectionless protocol. Being connectionless implies that IP is not aware of the concept of a connection between two entities, and therefore it does not maintain state information between the parties communicating. IP is used by the transport protocols TCP and UDP. The header and payload of the transport protocols are encapsulated, as shown in Figure 5.

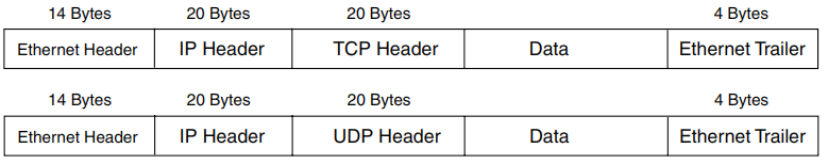


Figure 5. Protocol Encapsulation.

IP Header

Figure 6 illustrates the IP header.

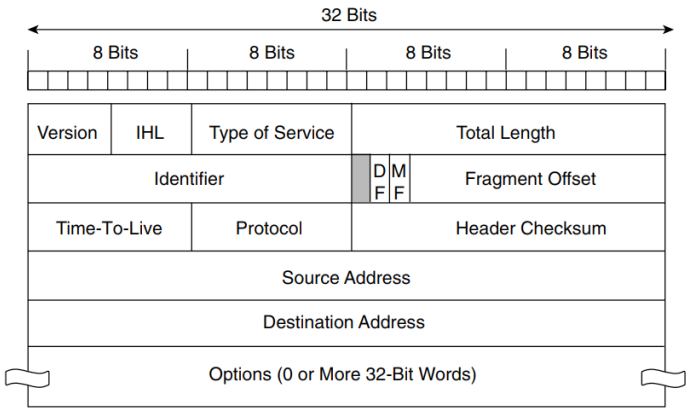


Figure 6. IP Header.

The fields in the IP header control the communication exchange between the two parties. Each field carries specific information explained in the sections that follow.

Version Field

Version is a 4-bit field that specifies the version of IP in use by the server and receiver. Table 1 describes the current IP version numbers.

Table 1. IP Version Field

Binary	Decimal	Version	RFC
0000	0	Reserved	—
0001, 0010, 0011	1-3	Unassigned	—
0100	4	IPv4 (standard IP)	791
0101	5	Stream IP Datagram Mode (experimental)	1190
0110	6	IPv6, also known as IPng	1883
0111	7	TP/IX: The Next Internet	1475
1000	8	The "P" Internet Protocol (PIP)	1621
1001	9	TCP and UDP over Bigger Addresses (TUBA)	1347
1010–1110	10–14	Unassigned	—
1111	15	Reserved	—

The Internet Assigned Numbers Authority (IANA) in RFC 2780 provides the guidelines for the allocation of "Values in the Internet Protocol and Related Headers." The IANA addresses fields for the following protocol headers:

- IPv4
- IPv6
- Internet Control Message Protocol (ICMP)
- UDP
- TCP

Some of the IANA guidelines affect how the IPv4 fields are used. These effects are discussed in sections that are relevant to each field. For instance, RFC 2780 specifies how the high-order bit of the version field is used for TCP/IP (RFC 1144) header compression and how the three high-order bits are used for IP (RFC 2507) **header compression**.

NOTE: You typically use TCP/IP header compression on low-speed serial lines on which bandwidth is scarce. When header compression is in use, the IP version field is ignored. The header information on a compressed packet is different from that of a non-compressed packet.

Header Length Field

Header length is a 4-bit field specifying the number of 4-byte words in the IP header, which yields a maximum value of 60 bytes:

60 Bytes = 15 (maximum field value) * 4-byte words

The typical value of this field is 5 (20 bytes) in the absence of options.

Type of Service Field

Type of service is an 8-bit field known as the TOS field, which originally consisted of a 3-bit precedence field, 4 TOS bits, and an unused bit (bit 0). The 8-bit field, referred to as the differentiated services (DS) field, which superseded the TOS definition, is divided in a 6-bit differentiated services code point (DSCP) and a 2-bit field currently unused. The use of the DS field, defined

in RFC 2474, "An Architecture for Differentiated Services." RFC 2481, a Proposal to add Explicit Congestion Notification (ECN) to IP, presents an experimental use of the currently unused bits.

RFC 3168 obsoletes RFC 2481 and updates RFC 2474 by defining an ECN field in the IP header.

Under this previous definition, depicted in part A of Figure 7, the TOS bits were divided into a 3-bit precedence field, a 4-bit TOS field, and the unused 0 bit.

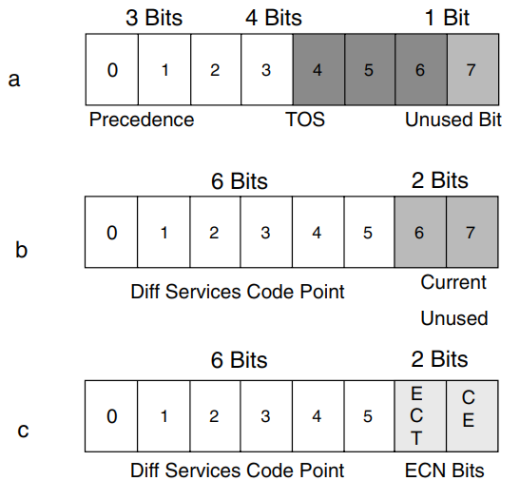


Figure 7. TOS Definition.

Table 2 shows how the precedence bits (bits 0, 1, and 2, starting from the high-order bit) are mapped (as specified in RFC 1349).

Table 2. Precedence Bits

Precedence Bits	Priority
000	Routine
001	Priority
010	Immediate
011	Flash
100	Flash override
101	Critic/ECP
110	Internetwork control
111	Network control

3G E-LEARNING

Table 3 shows how the TOS field bits are mapped (as specified in RFC 791 and RFC 1349).

Table 3. Type of Service Bits

TOS Bits	Semantics
0000	Normal service (default value)
Bit 6: 0001	0: Normal monetary cost 1: Low monetary cost
Bit 5: 0010	0: Normal reliability 1: High reliability
Bit 4: 0100	0: Normal throughput 1: High throughput
Bit 3: 1000	0: Normal delay 1: Low delay

Although the DS approach is adopted and its support is pervasive on networking equipment, the previous specification of the TOS field is still applicable. This point is important especially when it comes to applying quality of service (QoS) to IP network traffic.

Part B of Figure 7 depicts the interim step before the ECN bit was explicitly added to the TCP header. (ECN is no longer experimental.) The TOS field now consists of 6 bits allocated for the DSCP and 2 unused bits. Part C of Figure 7 presents the new definition for the previously unused bits. The new definition, in RFC 3168, specifies the 2-bit field as the ECN codepoint with four possible values (the possible combination of their binary values). The ECN capable transport (ECT) codepoints 10 and 01, or ETC(0) and ETC(1), indicate that the endpoints of the transport protocol are ECN capable. The other codepoints, 00 and 11, indicate that the codepoints are not ECN capable and that there is congestion in the endpoints, respectively. Congestion experienced is also known as CE. Table 4 displays the bit in reference to other ECN values.

Table 4. ECN Field Values

ECT	CE	Semantics
0	0	No ECT endpoints
0	1	ECT(1)
1	0	ECT(0)
1	1	Congestion experienced

Total Length Field

Total length is a 16-bit field that carries the total length of the IP datagram, including the header in chunks of 8 bytes (octets). A 16-bit field yields a maximum value of 65,535 bytes in an IP datagram.

TIP: Subtracting the header length from the total length yields the IP datagram payload size.

Identifier Field

The identifier is a 16-bit field that identifies the fragments of a fragmented IP datagram. The identifier is incremented by the source when it transmits a new IP datagram. The identifier field with the flags and fragmented offset fields are used to reassemble fragmented datagrams.

IP datagrams are fragmented when they exceed the maximum transmission unit (MTU) of a particular segment they traverse. This process implies that either the source host or an intermediate router placing the datagram in the segment performs the fragmentation. The host or router fragments the datagram in pieces that do not exceed the MTU and places the same number in the identifier field. The destination host uses the ID to reassemble the datagram.

NOTE: The MTU size depends on the encapsulation used by the data link layer protocol, which is typically 1500 (1492 bytes with the Logical Link Control [LLC] and Subnetwork Access Protocol [SNAP] headers) for Ethernet v2 and 802.3. Path MTU refers to the smallest MTU found in the path the packets traverse between source and destination hosts. The path is assumed to have more than one data link segment. The **fragmentation** is performed based on the MTU, which implies it is done potentially at the source hosts or intermediate routers.

Flags Field

The flags field is a 3-bit field that contains information about IP datagram fragmentation. The first bit is unused. The second bit is the Don't Fragment (DF) bit, which when set to 0, indicates the packet is not to be fragmented. A packet that cannot be fragmented is discarded by the router, which sends an ICMP

"Fragmentation needed and Don't Fragment Flag Set" message back to the source host. The third bit is the More Fragments (MF) bit that signals whether there are more fragments expected. The router sets this bit to 1 in all fragments and 0 in the last fragment. The destination host expects to receive fragments until the MF bit equals 0.

Fragment Offset Field

The fragment offset field is a 13-bit field that carries the offset of the fragment from the header of the original datagram in chunks of 8 bytes. Because fragments could arrive out of order to the destination hosts, the offset value allows the host to reassemble them in the correct order.

Time-To-Live Field

The Time-To-Live field (TTL) is an 8-bit field that specifies how long the packet should live, which is counted in terms of the number of routers traversed by the datagram (hop count). The initial TTL value is selected by the source host, and it is recommended to use 64. Often, other values are used to limit the time, in hops, that the packet should live. This TTL keeps packets from lingering on the network unnecessarily.

The TTL is decremented by 1 by every router it traverses. After the TTL has been decremented to 0, the packet is discarded, and an ICMP time-to-live equals 0 during transit message is sent to the source host.

NOTE: The traceroute program uses the TTL to determine the "route" or path from the source host to the destination host. The traceroute program sends a datagram with a TTL of 1 to the destination host. The intermediate router decrements the TTL to 0, discards the packet, and sends the ICMP "time exceeded" message back, which permits the source host to identify the first router in the path. The source host sends the datagram, this time with a TTL value of 2, which reaches the second intermediate router. This process is repeated until the final destination is reached and the path is recorded to the source host. A router that receives a datagram in which the TTL field is 0 must discard it and send back an ICMP "time

exceeded" message to the source host. This detail is important to note: When the datagram finally reaches the destination host with TTL 1, the host returns an ICMP "port unreachable" message instead of a "time exceeded" message, which makes it possible for the source host to determine that it has reached the destination host. The "port unreachable" message results from the destination port being in an unlikely range supported by the destination host (greater than 30,000).

Protocol Field

The protocol field is an 8-bit field that identifies the protocol in use. Most protocols in the TCP/IP suite rely on IP, which requires some kind of identification for the specific protocol using the IP layer.

Table 5 lists the most common protocols.

Table 5. Protocol Number Table

Decimal	Keyword	Protocol	References
0	HOPOPT	IPv6 Hop-by-Hop Option	RFC 1883
1	ICMP	Internet Control Message Protocol	RFC 792
2	IGMP	Internet Group Management Protocol	RFC 1112
3	GGP	Gateway-to-Gateway Protocol	RFC 823
4	IP	IP in IP (encapsulation)	RFC 2003
5	ST	Stream	RFC 1190, RFC 1819
6	TCP	Transmission Control Protocol	RFC 793
8	EGP	Exterior Gateway Protocol	RFC 888, DLM 1
9	IGP	Any private interior gateway (used by Cisco for its IGRP)	IANA
16	CHAOS	Chaos	NC 3
17	UDP	User Datagram	RFC 768, JBP
29	ISO-TP4	ISO Transport Protocol Class 4	RFC 905, RC 77
30	NETBLT	Bulk Data Transfer	RFC 969, DDC 1
35	IDPR	Inter-Domain Policy Routing	MXS1
41	IPv6	IP Version 6	Deering
43	IPv6-Route	Routing Header for IPv6	Deering
44	IPv6-Frag	Fragment Header for IPv6	Deering
45	IDRP	Inter-Domain Routing Protocol	Sue Hares
46	RSVP	Resource Reservation Protocol	Bob Braden
47	GRE	General Routing Encapsulation	Tony Li
48	MHRP	Mobile Host Routing Protocol	David Johnson

50	ESP	Encapsulating Security Payload for IPv6	RFC 1827
51	AH	Authentication Header for IPv6	RFC 1826
55	MOBILE	IP Mobility	Perkins
56	TLSP	Transport Layer Security Protocol using Kryptonet key management	Oberg
58	IPv6-ICMP	ICMP for IPv6	RFC 1883
59	IPv6-NoNxt	No Next Header for IPv6	RFC 1883
60	IPv6-Opts	Destination Options for IPv6	RFC 1883
80	ISO-IP	International Organization for Standardization IP	MTR
83	VINES	VINES	BXH
85	NSFNET-IGP	National Science Foundation Network IGP	HWB
88	EIGRP	Enhanced Interior Gateway Routing Protocol	Cisco, GXS
89	OSPFIGP	Open Shortest Path First Interior Gateway Protocol	RFC 1583, JTM 4
94	IPIP	IP-within-IP Encapsulation	JI6
95	MICP	Mobile Internetworking Control Protocol	JI6
97	ETHERIP	Ethernet-within-IP Encapsulation	RFC 3378
98	ENCAP	Encapsulation Header	RFC 1241, RXB3
102	PNNI	Private Network-Network Interface over IP	Callon
103	PIM	Protocol Independent Multicast	Farinacci
108	IPComp	IP Payload Compression	RFC 2393
111	IPX-in-IP	Internetwork Packet Exchange in IP	Lee
112	VRRP	Virtual Router Redundancy Protocol	Hinden
113	PGM	Pragmatic General Multicast Reliable Transport Protocol	Speakman
114		Any 0-hop protocol	IANA
115	L2TP	Layer 2 Tunneling Protocol	Aboba
121	SMP	Simple Message Protocol	Ekblad
132	SCTP	Stream Control Transmission Protocol	Stewart
133	FC	Fibre Channel	Rajagopal
135-254		Unassigned	IANA
255		Reserved	IANA

Header Checksum Field

The header checksum field is a 16-bit field that contains the checksum of the IP header only. The checksum is calculated by using the 16-bit 1s complement sum of the header. On arrival, the

receiver of the datagram performs its own checksum calculation on the header and determines whether the datagram needs to be discarded.

The next layer protocols are in many cases responsible for their own checksum calculations, which in TCP and UDP include fields from the IP header as well.

Source Address and Destination Address Fields

Both the source address and destination address are 32-bit fields that carry the source and destination IP addresses.

Options Field

The options field is a variable-length field in which the actual chunks come in 32-bit size portions that carry optional information used by the **datagram**. This field is optional and might not be present on every IP packet. The following are the currently defined options:

- **Source Routing**—Source routing is the ability to route packets based on information contained in the source route fields. The information in the source route fields is placed by the source host. Intermediate routers must obey the source routes. There are two types of source routing: loose source route (LSR) and strict source route (SSR). LSR permits a router to pick the next hop router used to forward the datagram based on the route indicated in the source route field. The IP header carries only a number of IP addresses to use to route but not all of them. SSR forces intermediate routers to follow the routes specified in the source route field.
- **Record Route**—The record route (RR) option makes each router record its own IP address. The ping program uses this option to issue an RR in the datagram, which includes an ICMP echo request message. Each router the datagram traverses adds its IP address to the options field. When the destination host receives the datagram, it sends back an ICMP echo reply with all the recorded IP addresses. All routers in the return path add their own addresses to the datagram, which implies that the

source host has the IP addresses of the routers in the forward and reverse paths.

- **Timestamp**—The timestamp is similar to the RR option, except that it adds the time when the packet touched each router. In addition to the 3 bytes used for the IP option, length and pointer, there are two more fields in use: the overflow (4 bits) and the flags (4 bits) fields. The overflow field is incremented by the routers when there is no room left in the header. The flags field is used to signal whether the routers should record the timestamp only (0 value), timestamp with IP address (1 value), or a timestamp when the router's IP address matches the next IP address in the pre-initialized list of IP addresses. When IP addresses and timestamps are used, there could be up to four pairs (IP address and timestamp) in the options part of the IP header. Each timestamp requires 4 bytes.
- **Security and Handling Restrictions**—RFC 1108 describes in detail the use of the options for security and handling restrictions.

TIP: The length of the IP header is limited by the size of the IP header length field of 4 bits. This limit implies that the maximum length of the IP header is 60 bytes, minus the fixed 20 bytes from the header, leaving 40 bytes for options. There are 3 bytes used: 1 byte for the IP option, 1 byte for the length of the RR option, and 1 byte for the pointer field to signal the start of the current IP addresses list (typically the fourth byte). It leaves room for nine IP addresses, which across a large network might not be enough to record all hops. After the ninth IP address is recorded, the pointer field points to byte 40, indicating that the list is full.

TCP

TCP, originally defined in RFC 793, provides a reliable host-to-host communication service (also referred to as client-to-server). From the discussion in the previous section, you understand that IP does not offer a reliable connection. The extent of IP's reliability is the ability to determine whether the IP header has been changed through the checksum field. TCP was designed to address the following areas, some of which are defined in RFC 793:

- Basic data transfer (byte-stream service)
- Reliability
- Flow control
- Multiplexing
- Connections

"A computer lets you make more mistakes faster than any invention in human history - with the possible exceptions of handguns and tequila."

—Mitch Ratliff

Basic Data Transfer

Basic data transfer refers to the capability for a continuous data transfer, which is bidirectional sender-to-receiver and receiver-to-sender. The implication is a mechanism that allows the upper layers of the stack to tell the TCP process to send data immediately. This process is possible because of the push flag, which is discussed in detail in the section, "TCP Control Flags." An important aspect of the basic data transfer is TCP's capability for continuous data transfer, which is better described as a byte-stream service. TCP is not based on the concept of application data boundaries such as records or block.

TCP simply receives data from the application layer of the stack, stuffs the data into a segment, and sends it down to the stack. The applications are expected to control the contextual boundaries of the data because TCP just sends streams from one end to the other. In some cases, the TCP protocol stack might wait for more data from the application to send a larger segment, thus increasing efficiency. Because the stream is just a number of bytes, the state information about the connection requires knowledge of the transmitted and received bytes. It uses two key pieces of information:

- Sequence number (SEQ)
- Acknowledgement number (ACK)

Both sequence and acknowledgement numbers use bytes or rather, the byte numbers of the data being sent or received, to keep track of the state, which is easier to control as it maps directly to the byte stream.

Reliability

Reliability refers to the capability of TCP to recover from any kind of error condition that could prevent the successful communication between sender and receiver. TCP expects the datagrams to be acknowledged (ACK) within a timeout interval, after which it attempts retransmissions for a period of time before ceasing any attempts. Sequence numbers related to each segment are used to order the segments because they could have arrived out of order. Finally, a checksum over the entire segment ensures that there is no damage to the payload or header. If there is damage, the segment is discarded.

Flow Control

Flow control refers to the capability of the receiver to control the amount of data the sender is sending. The receiver sends the window size (advertised window) along with every ACK that tells the sender how many bytes can be sent after a specific segment SEQ. This process allows the receiver to throttle back the amount of data being received.

Multiplexing

Multiplexing refer to the capability of TCP to maintain many simultaneous processes in a single host. It uses port numbers, which identify distinct application processes. The application process and the network address form a socket. Different connections can use a single socket, yet a connection between two hosts is uniquely identified by the pair of sockets (source and destination addresses and source and destination port numbers).

Connection

Connection refers to the concept of a communication stream between two devices for which state is maintained. The state is maintained to ensure the communication process is successful for as long as it is active. Additionally, the aforementioned control mechanisms of SEQ and window size help maintain reliability and flow control. TCP ensures reliability by using flow-control mechanisms to control the conversation, and error correction and

detection to handle communication problems when they appear. TCP is also known for its connection-oriented service in which a connection between server and receiver is established before they start the **data exchange**. This initial connection uses what is referred to as a three-way handshake.

The reliability offered by TCP results from handling the host-to-host communication in the following manner:

- Selecting the preferred size of portion of data to send.
- Maintaining a timer when a segment is sent. An acknowledgement of the segment is expected, and if it is not received, the portion of data is retransmitted.
- Tagging each segment being sent with a SEQ so that the receiver is able to order the data.
- Using a mechanism to throttle the traffic flow between sender and receiver by means of a window that increases as packets are successfully received or decreases as packets are dropped.

TCP Header

This section on TCP presents a summary of the TCP header fields and introduces concepts that are critical in the understanding of TCP as it relates to devices that interact with the TCP connection, purposely or transparently, when processing server farm traffic. TCP as IP has changed over the years. RFC 3168 has updated RFC 793. Some of the new changes in RFC 3168 affect the use of two TOS bits for ECN in the IP header and some include changes to the TCP header. We introduce these changes as we explain the different fields in the TCP header.

Figure 8 introduces the TCP header, as defined in RFC 793.

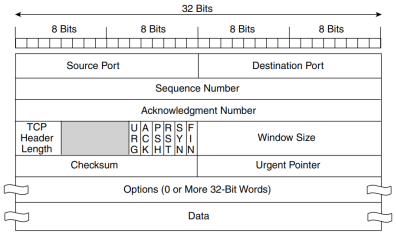


Figure 8. TCP Header.

Source Port and Destination Port Fields

The source and destination port number fields are each 16 bits. (Values range from 1 to 65,535.) Port numbers uniquely identify an application process at the source and destination hosts. An application process running on a host uses a unique port number that could be multiplexed.

Figure 9 presents an example of multiplexing in which Client1 and Client2 have established connections with Server1. Client1 and Client2 are connected to Server1 using the same destination port (port 80), yet their source ports are unique. Each combination of source/ destination port number identifies a unique connection.

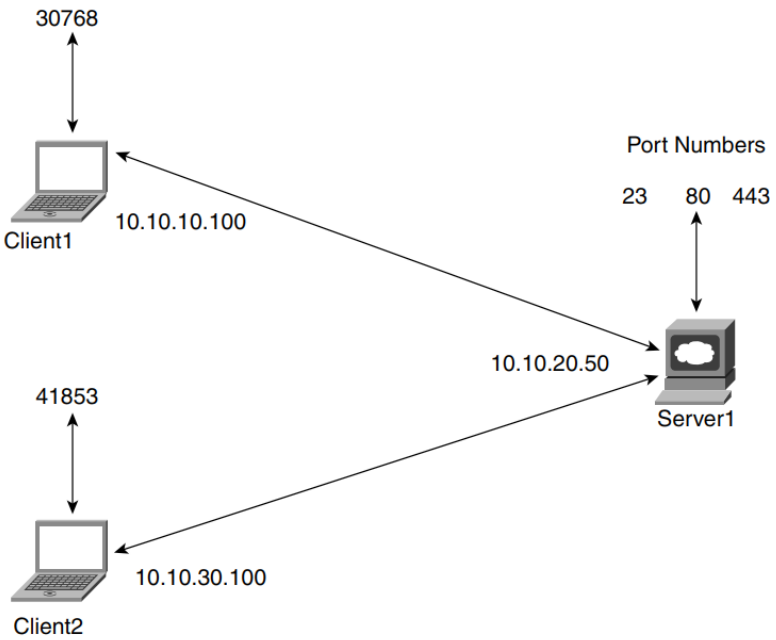


Figure 9. TCP Connections.

The application process listens on a specific port for all incoming traffic, in this case from Client1 and Client2. The host, Server1, must provide a mechanism to handle multiple connections on the same port concurrently. The combination of source and destination port numbers and IP addresses identifies a single and unique connection between sender and receiver.

TIP: The port numbers were traditionally documented in the "assigned numbers" RFCs, which have become obsolete regularly since RFC 739. (RFC 739 obsoleted both RFC 604, "Assigned Link Numbers," and RFC 503, "Socket Number List.") RFC 3232 obsoleted, which was the 1994 snapshot of the assigned list and indicates, officially, that the assigned number list is an online database accessible through the web at http://www.iana.org. The list of port numbers appears at http://www.iana.org/assignments/port-numbers.

Table 6 lists well-known port numbers and their associated protocols.

Table 6. Well-Known Port Numbers and Associated Protocols

Keyword	Decimal	Description
ftp-data	20/tcp	FTP (default data)
ftp-data	20/udp	FTP (default data)
ftp	21/tcp	FTP (control)
ftp	21/udp	FTP (control)
ssh	22/tcp	Secure Shell (SSH) remote login protocol
ssh	22/udp	SSH remote login protocol
telnet	23/tcp	Telnet
telnet	23/udp	Telnet
smtp	25/tcp	Simple Mail Transfer Protocol (SMTP)
smtp	25/udp	SMTP
domain	53/tcp	Domain Name System (DNS)
domain	53/udp	DNS
tacacs-ds	65/tcp	TACACS (database service)
tacacs-ds	65/udp	TACACS (database service)
tftp	69/tcp	Trivial File Transfer Protocol (TFTP)
tftp	69/udp	TFTP
finger	79/tcp	Finger
finger	79/udp	Finger
http	80/tcp	HTTP
http	80/udp	HTTP
www	80/tcp	HTTP
www	80/udp	HTTP

www-http	80/tcp	HTTP
www-http	80/udp	HTTP
kerberos	88/tcp	Kerberos
kerberos	88/udp	Kerberos
pop2	109/tcp	Post Office Protocol (POP) Version 2
pop2	109/udp	POP Version 2
pop3	110/tcp	POP Version 3
pop3	110/udp	POP Version 3
nntp	119/tcp	Network News Transfer Protocol (NNTP)
nntp	119/udp	NNTP
ntp	123/tcp	Network Time Protocol (NTP)
ntp	123/udp	NTP
snmp	161/tcp	Simple Network Management Protocol (SNMP)
snmp	161/udp	SNMP
snmptrap	162/tcp	SNMPTRAP
snmptrap	162/udp	SNMPTRAP
bgp	179/tcp	Border Gateway Protocol (BGP)
bgp	179/udp	BGP
rsvp_tunnel	363/tcp	Resource Reservation Protocol (RSVP) Tunnel
rsvp_tunnel	363/udp	RSVP Tunnel
https	443/tcp	HTTP over Transport Layer Security/Secure Socket Layer (TLS/SSL)
https	443/udp	HTTP over TLS/SSL

The list of port numbers is organized as follows:

- Well-known ports: 0–1023
- Registered Ports: 1024–49,151
- Dynamic or private ports: 49,152–65,535

Registered ports are ports that have been explicitly reserved for specific applications by request yet they "can be used by ordinary user processes or programs executed by ordinary users," as defined by RFC 1340. A good example of registered ports is the Hot Standby Router Protocol (HSRP):

hsrp 1985/tcp HSRP

hsrp 1985/udp HSRP

NOTE: The list shows both TCP and UDP used as transport, which indicates the protocol port number has been reserved instead of it supported using UDP and TCP.

The dynamic or private port range is used by hosts to obtain a randomly generated port number when there is no port associated with the application.

Sequence Number Field

The sequence number field is a 32-bit field that identifies the first byte of data in the specific segment. TCP labels each byte in the communication stream with a sequence number and the sequence number of the first byte in the data that is sent. When a new connection is being established (the SYN flag is set), the sequence number field contains the initial sequence number (ISN) selected by the host for the connection. The ISN must be selected using the host clock to ensure the randomness of the actual number. Because TCP poses no restriction on a connection used repeatedly, it is important that TCP is capable of identifying potential duplicate segments for different instances of the same connection. TCP avoids this problem by preventing duplicate sequence numbers on the network for different instances of the same connection. The ISN selection is based on the clock incrementing every so often so that the ISN values are able to cycle less frequently than the maximum lifetime of a segment. This process prevents the same sequence number from being used simultaneously. RFC 793 specified the **maximum segment lifetime (MSL)** to be 2 minutes yet flexible to more practical values applicable in operational networks.

TIP: The selection of the ISN depends on the implementation of the TCP/IP stack on each host. Current implementations might not be compliant with the RFC 973 specification. The implications of a nonrandom selection of the ISN could imply a repeatable pattern that could be predicted, which opens the door for potential connection hijacks or prevents the successful completion of a connection handshake.

Certain TCP protocol stack implementations support SYN cookies. SYN cookies provide a choice in the values of the ISN, which are used to prevent SYN flood attacks. SYN flood attacks are designed to keep the SYN queue on the server full. The SYN cookie mechanism protects the SYN queue by selecting an ISN (the cookie value) based on a Message Digest 5 (MD5) of the source and destination IP addresses and port numbers. When the queue is full, it still sends a SYN/ACK but keeps no

3G E-LEARNING

connection state information. If the final ACK for the three-way handshake is received, the server recalculates the original information that had come with the initial SYN. The original information is encoded into the sequence number of the reply to the SYN packet. The acknowledgement number in the last packet of the three-way handshake will be the sequence number that was sent plus 1. When the acknowledgement arrives, 1 is subtracted and the number decoded. Using this mechanism, the server does not have to remember any connection information.

SYN cookies are supported in Linux and FreeBSD, but they are not enabled by default. The following command enables them:

> /proc/sys/net/ipv4/tcp_syncookies

There are some restrictions in using SYN cookies: the server must reject TCP options such as large windows and selective acknowledgement. The large window restriction is because the ISN is a 32-bit field.

Acknowledgement Number Field

The acknowledgement number field is a 32-bit field that identifies the sequence number the sender of the acknowledgment is expecting to receive next. The sender of the acknowledgement uses the sequence number of the last segment plus 1, which corresponds to the next byte the sender expects plus 1 byte. This field is used only when the ACK flag is set, which is after the connection is established. After the connection is established the field is always set and the ACK flag is always on (set to 1).

NOTE: A TCP connection between sender and receiver is full duplex, which implies that it supports concurrent information exchanges from sender to receiver and from receiver to sender. Each side of the connection is controlled independently by its own sequence and acknowledgement numbers, which ensures that flow control is accomplished on either direction independently.

According to the delayed ACK algorithm, TCP can delay an ACK for as long as two full-sized (one ACK for every second full-size segment) segments but no more than 500 milliseconds. TCP specifies the last byte that was successfully received. If it receives a new segment that does not correspond to the next chunk of contiguous bytes, TCP has to send an acknowledgment

to indicate the expected byte, even if the segment just received was successful. TCP has no mechanism to provide negative acknowledgment or to acknowledge that a chunk of data was successfully received.

Selective Acknowledgment (SACK) is a mechanism that could be used by the receiver to acknowledge all the segments that might have been successfully received so that the sender only retransmits the segments that have been lost.

TCP Header Length Field

The TCP header length field is a 4-bit header that carries the total length of the TCP header in 32-bit words (4-byte chunks). The maximum header length is 60 bytes (16 * 4), which leaves room for 40 bytes of options because the set fields are 20 bytes.

The next 4 bits are reserved.

NOTE: In RFC 793, the 6 bits following the header length field were reserved. RFC 3168 updates RFC 793 and takes 2 reserved bits for new control flags, leaving 4 reserved bits. The next section, "TCP Control Flags," explains the new flags.

TCP Control Flags

Figure 10 illustrates the eight control flags, per RFC 793 and RFC 3168.

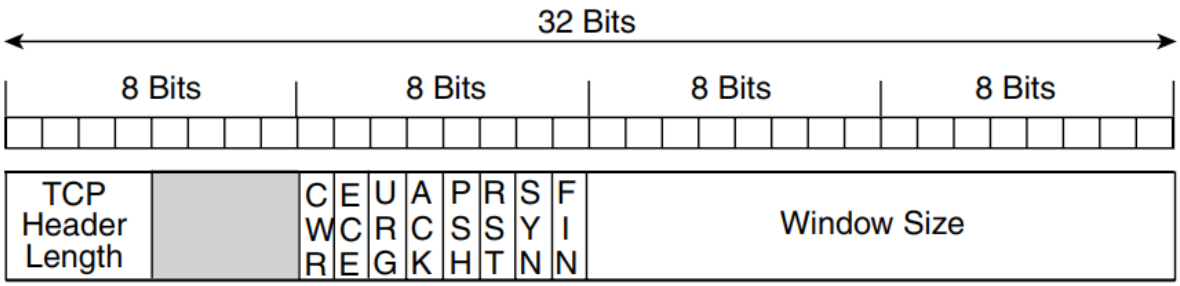


Figure 10. TCP Control Flags.

- **CWR and ECE**—ECN uses the ECT and CE flags in the IP header, introduced in the "Type of Service" section, for signaling between routers and endpoints. ECN uses CWR

3G E-LEARNING

and ECE for TCP endpoint-to-TCP endpoint signaling.

- **URG**—The sender uses this Urgent Pointer flag, when set, to indicate to the receiver that the receiver should accept urgent data. The receiver uses it to indicate to the sender when the user has received the last byte of urgent data. When set, this flag indicates that the urgent pointer field value should be used.
- **ACK**—The Acknowledgment flag, when set, indicates that the segment is used to acknowledge the successful reception up to the byte indicated by the acknowledgement value.
- **PSH**—The Push flag indicates to the receiver that it should pass the received data to the application as soon as possible.
- **RST**—The Reset flag notifies the receiver to reset the connection.
- **SYN**—The Synchronize sequence number flag indicates to the receiver that the sequence numbers should be synchronized. The SYN flag, when set, indicates a connection request from the sender to the receiver.
- **FIN**—The Finished control flag indicates that the sender has finished sending data. The receiver then proceeds to close that half of the connection.

Window Size Field

The window size field is a 16-bit field used for flow control between the sender and receiver. The sender and receiver each advertise their own window size. The size indicates the number of bytes, starting with the byte indicated by the acknowledgment field, that the sender of the segment is willing to receive. The maximum size of the window is 65,535 bytes (without the use of the Window Scale option).

Checksum Field

The checksum field is a 16-bit field that carries the value which must be calculated for every segment and which must cover both the TCP header and the TCP data. The receiver calculates and compares the checksum against the value received in the segment, and if it is different, the segment is discarded and not acknowledged.

Urgent Pointer Field

The urgent pointer field is a 16-bit field that indicates to the receiver that the value is an offset that must be added to the sequence number field to produce the sequence number of the last byte or urgent data. This field is used only when the URG flag is set. The receiver is notified of the urgent data (by the URG flag and urgent pointer value) in the data stream, and the receiver decides what to do when the notification is received.

Options Field

The options field is a variable-length field ranging from 0 to 40 bytes that carries TCP options. A few of the most common TCP options are the MSS, the window scale (windows larger than 65,535 bytes), the TCP selective acknowledgement (SACK), and the timestamp option (round-trip time measurement for data segments). Some of these options are discussed in the sections that follow. RFC 793 indicates that a TCP stack must support all options, and it indicates the defined options (at the time of the writing).

There are two alternatives for the option format:

- A single byte for the option kind
- A byte for the option kind, a byte for the length, and the option-data bytes

Kind refers to the type of option and length to its value in bytes. Table 7 lists the kind options defined in RFC 793.

Table 7. 3-Byte Option Format

Kind	Length	Meaning
0	—	End of option list
1	—	No operation
2	4	MSS

UDP

UDP, originally defined in RFC 768, provides a datagram host-to-host (also referred to as client-to-server) communication service with minimum protocol overhead. The protocol, as defined in

RFC 768, is "transaction oriented, and delivery and duplicate protection are not guaranteed." Unlike TCP, UDP does not have the concept of a connection between two peers. UDP is in fact referred to as a connectionless protocol and does not have the structure required for a connection-state tracking mechanism.

"You can't hold firewalls and intrusion detection systems accountable. You can only hold people accountable."

—Daryl White

In UDP, datagrams are sent without expectations of acknowledgments, timeouts, or **retransmissions**. Some applications built on UDP, such as TFTP, actually provide some means of reliability by retransmitting if packet acknowledgements are not received.

UDP Header

Figure 11 introduces the UDP header format.

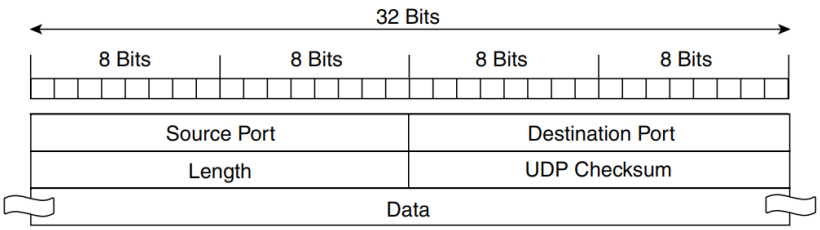


Figure 11. UDP Header Format.

As is visible in the UDP header format, the number of fields in a UDP header is less than that for TCP and amounts to 8 bytes instead of 20.

Source Port Field

The source port field is a 16-bit field that indicates the process port number of the sender that the receiver can use to communicate back if required. The source port is not always used, and it has the value 0 when not in use.

Destination Port Field

The destination port field is also a 16-bit field used to identify the process in the receiving host.

Length Field

The length field is a 16-bit field that indicates the length of the entire datagram, including the header and the data. The minimum header length is 8 bytes.

Checksum Field

The checksum field is a 16-bit field that indicates whether there is any corruption in the datagram. The checksum calculation, as described in RFC 768, is "the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets." This process basically means using the content of the pseudo header (which also includes a portion of the IP header) to come up with a padded value (the value may be padded if it is not multiple of 2 bytes) to which the 1s complement operation is applied. Except for the pseudo header, the checksum calculation is the same as that with TCP, but unlike TCP, the inclusion of the checksum is optional. The pseudo header, as shown in Figure 12, contains the source and destination IP addresses, the protocol, and the UDP length fields. The header provides this information in case the datagrams are not properly routed so the receiving host can check that the received datagram arrived at the correct destination. The RFC specifies that in the IP stack, there should be a mechanism that allows UDP to determine the source and destination IP address and the protocol field from the IP header.

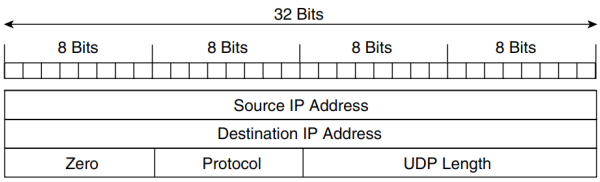


Figure 12. UDP Pseudo Header Format.

Using the same procedure as in TCP, the receiver uses the checksum field to calculate and compare against the checksum value it received in the segment, and if they are different, the segment is discarded.

UDP Header Compression

UDP header compression follows the same principles of compression applied to TCP and has the same overall benefits. The UDP header can be compressed down to 2 bytes. Both the source and destination source ports remain the same during the life of the connection, and the length must match the length of previous headers, which leaves the checksum (2 bytes) as the field that changes regularly.

Figure 13 presents two compressed non-TCP header formats.

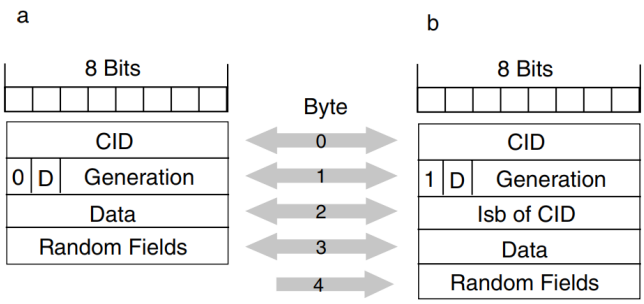


Figure 13. Compressed Non-TCP Header Formats.

The context ID (CID) can be either 8 or 16 bits, and it is determined by the first bit of the second byte. If the bit equals 1, the next byte holds the least significant bits of the CID. The D bit on the second byte determines whether there is data (if bit = 1). A simple explanation for generation is a version related to a particular context identified by a CID.

CONCLUSION

IP is simply the most prevalent network layer protocol on the Internet and intranets to date. The TCP/IP protocols suite supports two transport protocols: TCP and UDP. TCP, a connection-oriented protocol, provides reliability and flow control. It guarantees error

detection and correction and packet delivery for applications needing such features. TCP continues to evolve by improving its operation and functions to better utilize network resources. UDP, a connectionless protocol, provides no reliability or flow control, yet it offers low overhead, which makes it the choice for applications in which low overhead is important and reliability is not critical.

Understanding each of these protocols and their behavioral implications is fundamental to the Data Center, where they come together as the source or destination of traffic. The Data Center is an environment where application awareness makes the knowledge about the protocol's operation relevant and key to architectural and design considerations. TCP offers a number of services better suited for applications that require certain service guarantess and can tolerate the header overhead and flow control issues and need error correction and detection. UDP, on the other hand, offers less overhead and fewer services, yet it is useful for application where packet loss is less sensitive but time delays are more problematic. The use of devices such as firewalls and load balancers requires you understand how the protocols in the TCP/IP suite work to ensure the operational behavior is the expected one. A good understanding of the working details of IP, TCP and UDP help you design and operate a better network.

3G E-LEARNING

KEY TERMS

Protocol stack
Ethernet frame
Header compression
Fragmentation
Datagram
Data exchange
Maximum Segment Lifetime (MSL)

DISCUSSION QUESTIONS

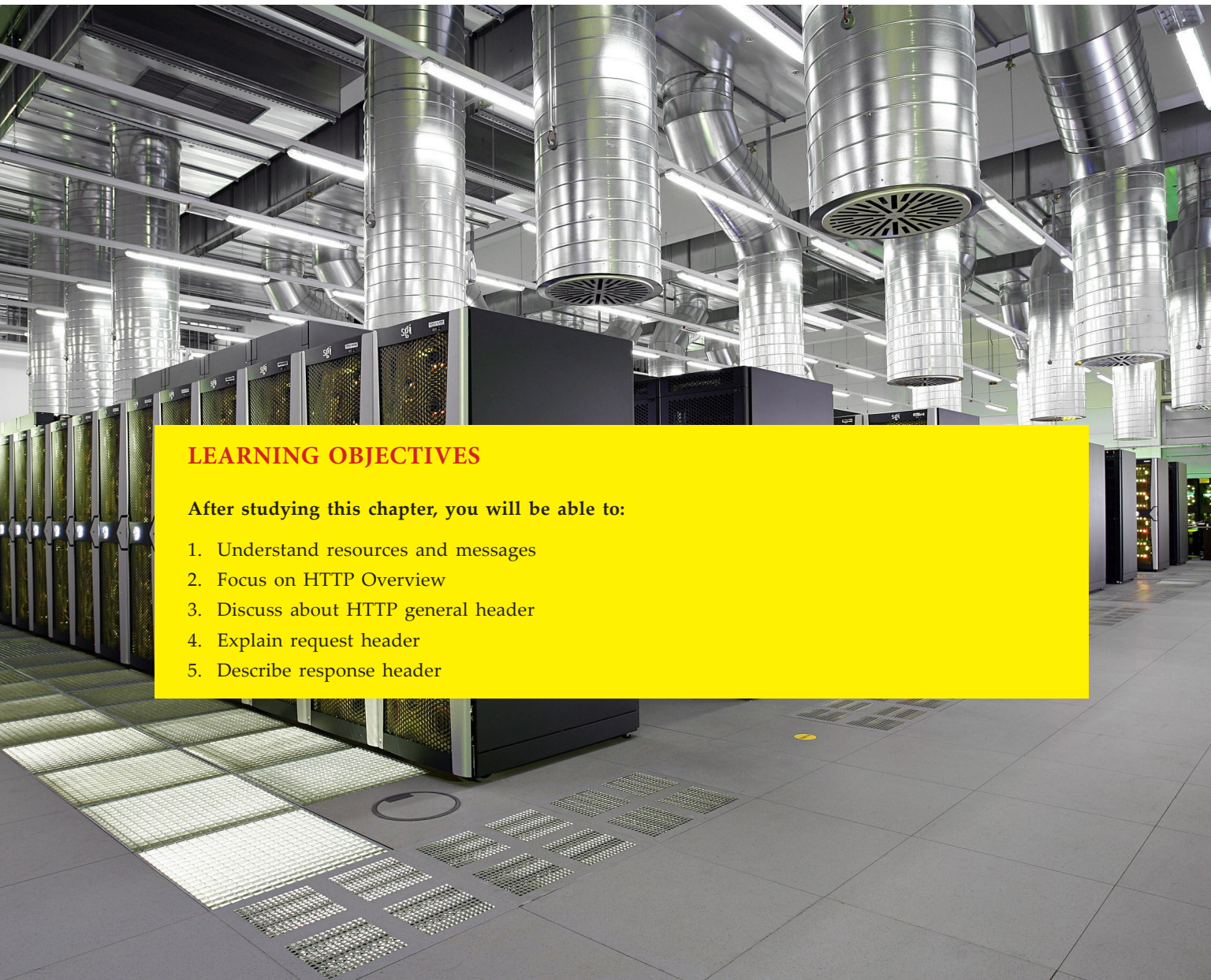
1. What is OSI model?
2. What do you mean by IP?
3. Is IP a computer or internet?
4. What is an IP network?
5. How do you find your IP?
6. Describe the type of service field.
7. What is TCP and its features?
8. Write the short notes on following:
   - Source Routing
   - Record Route
   - Timestamp
9. What do you understand by basic data transfer?
10. What is UDP? Give the UDP header format.

REFERENCES

1. Bennett, Richard (September 2009). "Designed for Change: End-to-End Arguments, Internet Innovation, and the Net Neutrality Debate" (PDF). Information Technology and Innovation Foundation. p. 11. Retrieved 11 September 2017.
2. Bill Cerveny (2011-07-25). "IPv6 Fragmentation". Arbor Networks. Retrieved 2016-09-10.
3. Clark, M.P. (2003). Data Networks IP and the Internet, 1st ed. West Sussex, England: John Wiley & Sons Ltd.
4. Comer, Douglas E. (2006). Internetworking with TCP/IP: Principles, Protocols, and Architecture. 1 (5th ed.). Prentice Hall. ISBN 978-0-13-187671-2.
5. David Murray; Terry Koziniec; Sebastian Zander; Michael Dixon; Polychronis Koutsakis (2017). "An Analysis of Changing Enterprise Network Traffic Characteristics" (PDF). The 23rd Asia-Pacific Conference on Communications (APCC 2017). Retrieved 3 October 2017.
6. F. Gont (July 2011). Security Assessment of the Internet Protocol version 4. doi:10.17487/RFC6274. RFC 6274.
7. Fernando Gont (July 2008), Security Assessment of the Internet Protocol (PDF), CPNI, archived from the original (PDF) on 2010-02-11
8. Forouzan, B.A. (2000). TCP/IP: Protocol Suite, 1st ed. New Delhi, India: Tata McGraw-Hill Publishing Company Limited.
9. Gont, Fernando (November 2008). "On the implementation of TCP urgent data". 73rd IETF meeting. Retrieved 2009-01-04.
10. John T. Hagen; Barry E. Mullins (2013). TCP veto: A novel network attack and its application to SCADA protocols. Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES. pp. 1–6. doi:10.1109/ISGT.2013.6497785. ISBN 978-1-4673-4896-6.
11. Kurose, J. F.; Ross, K. W. (2010). Computer Networking: A Top-Down Approach (5th ed.). Boston, MA: Pearson Education. ISBN 978-0-13-136548-3.
12. Mitrani, Isa (January 2013). "Managing performance and power consumption in a server farm". Annals of Operations Research. 202 (1): 121–122. doi:10.1007/s10479-011-0932-1. Retrieved February 8, 2013.
13. Mulligan, Geoff. "It was almost IPv7". O'Reilly. O'Reilly Media. Retrieved 4 July 2015.

CHAPTER 8

HTTP AND RELATED CONCEPTS



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Understand resources and messages
2. Focus on HTTP Overview
3. Discuss about HTTP general header
4. Explain request header
5. Describe response header

INTRODUCTION

HTTP is the most widely used protocol for web applications, both on the Internet and on intranets. HTTP, which relies on TCP as the transport protocol, is used to carry many types of objects or content such as text, images, and video, yet it also allows file transfers and video streaming.

This chapter describes the functional and operational aspects of HTTP and presents examples of its use based on traffic captured when real transactions are conducted. This chapter also presents concepts related to HTTP that are fundamental to the full understanding of how HTTP works. These include the treatment of resources as the object of a request or response, and the messages used by the peers in the HTTP transaction (request/response interaction).

RESOURCES AND MESSAGES

The HTTP specification provides a set of commands, referred to as methods, that indicate the purpose of the communication exchange (request or response) in addition to some basic functionality used to retrieve, update, and search information. This "information" is typically referred to as a resource or resources, and they must be clearly defined so that they are handled in a nonambiguous manner. This definition includes an identification, location, and name that are universally recognized and accepted so the resources can be used in a variety of applications by a number of protocols, including HTTP. For example, if a user is trying to perform a search on the web for a particular web page, the identification, location, and name must be unique to ensure a successful result (whether the object of the search exists or not).

"A successful website does three things:

It attracts the right kinds of visitors.

Guides them to the main services or product you offer.

Collect Contact details for future ongoing relation."

— Mohamed Saad

Another equally important aspect is the format of the pieces of information used in the communication exchange between peers when dealing with resources. These pieces of information are call messages, and like resources, they too must be clearly defined. The HTTP specification relies on how resources are identified (URI), located (URL), and named (URN) to execute the proper operation. Messages, the information unit between HTTP peers, are based on the Multipurpose Internet Mail Extensions (MIME) definitions. The following sections discuss these concepts.

URIs

In our daily life, we refer to many objects and every object is identified by its name. But a name is not a unique identifier. There can be many people with the same name.

The next element which assists in making the name unique is the location or the address. The address has a hierarchical structure that helps us to navigate to the specific location and reach the specific person with the name. **For example,** Flat No, Building Name, Suburb, City, Country.

Similar to the real world, the web world is also loaded with lots of information and documents which are distributed worldwide. In order to reach the specific document on the web, we need a unique identifier.

A sequence of characters that identifies a logical or physical resource uniquely in Web technology is called a Uniform Resource Identifier.

Types of URIs

The main two types of URI are:

- Uniform Resource Locator (URL)
- Uniform Resource Name (URN)

The other types are:

- Uniform Resource Characteristics (URC)
- Data URI

Uniform Resource Locator (URL)

- It gives the location of the object in a disciplined and structured format. This enables a unique identification of the object. But any change in the location of the object, due to say a server change, cannot be carried out automatically.
- URLs are a subset of URIs. All URLs are URIs, but all URIs are not URLs.
- **For Example,** mailto:myname@webpage.com & ftp://webpage.com/download.jpg

Uniform Resource Name (URN)

- It gives the name of the object which may not be unique. There is no common universal standard for naming the object. Hence this method to identify objects uniquely has failed.
- **Example: urn:isbn:00934563** identifies a book by its unique ISBN number

Uniform Resource Characteristics/Citations (URC)

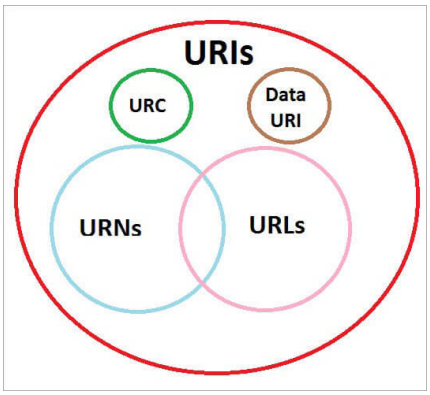
- It gives basic metadata about the resource that can be understood by humans and also parsed by a machine.
- URCs were a third identifier type. The purpose was to give a standardized representation of document properties, such as access restrictions, encoding, owner, etc.
- **Example:** view-source: http://exampleURC.com/ is a URC pointing to the HTML source code of a page.
- The basic functional expectation from a URC is structure, encapsulation, scalability, caching, resolution, easy readability, and interchangeability between protocols like TCP, SMTP, FTP, etc.
- URCs were never practiced and are not so popular, but the core concepts influenced future technologies like RDF.

Data URI

- Data can be directly placed into a Uniform Resource Identifier instead of

giving its location (URL) and Name (URN). Data URI allows embedding all kinds of objects within a web page. It is very useful to load frequently used images or lots of small images (less than 32×32 pixels).

- Performance enhancement is the main purpose of using data Identifiers. All resources used in the website are fetched by the browser using an HTTP request and almost all browsers limit the concurrent HTTP request usage to two. This creates a bottleneck of data impacting the overall performance of the site.
- Data URI removes the need for the browser to fetch additional resources and helps in performance improvement.
- It is important to note that the base64 encoding enlarges the images to ~ 30%. So, data URI with base64 encoding should be avoided if image size is of importance.
- Second, the decoding process involved makes the initial page load slower.
- **Syntax:** data: [media type] [; base64], [data]
    - Media type -> It is optional. But it is always a good idea to include it. The default is "text/plain".
    - base64 -> It is optional. It indicates that the data is base64 encoded data.
    - Data -> The data that needs to be embedded in the page.
- **Example:** data:,Hello%2021World.



Features of URI

Listed below are the main features or basic requirements for Uniform Resource Identifier:

- **Uniqueness:** Uniform Resource Identifier should give every resource available on the Internet or worldwide web a unique distinguishing identity.
- **Universality:** It should be able to identify or address every available resource on the Internet.
- **Extensibility:** New resources which are not yet a part of the worldwide web should be able to be identified by a unique new Uniform Resource Identifier.
- **Fixability:** This Identifier should be editable and changeable. It should be shareable and printable.

Syntax of Uniform Resource Identifier

Internet Engineering Task Force IETF and Worldwide Web Consortium (W3C), an international community working to develop web standards, has published a document RFC 1630. This document provides guidance and information to the Internet community for a unifying syntax to encode the names and addresses of objects on the Internet as used by WWW.

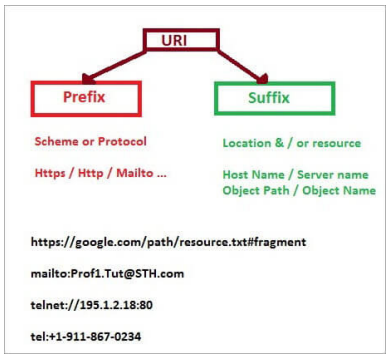
Syntax of URI -> Prefix + Suffix

- **Prefix** details the protocol
- **Suffix** details of the location and/or resource identification

https://www.google.com/login.html

Here,

- **https:** Protocol
- **www.google.com:** location
- **login.html:** resource identifier (a file)



URLs

Every time you click a link on a website or type a web address into your browser, it's a **URL**. URL stands for Uniform Resource Locator. Think of it like a street address, with each portion of the URL as different parts of the address, and each giving you different information.

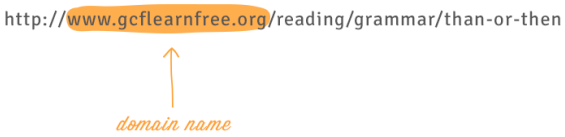
Scheme

Every URL begins with the scheme. This tells your browser what type of address it is so the browser connects to it correctly. There are many types of schemes, but for typical web browsing you will mostly see http and https. Your browser usually won't show the scheme in the address bar, and usually you don't need to type the scheme when typing a web address; instead, you can just begin with the domain name. The scheme is still always part of the URL; it just isn't being displayed.



Domain name

The domain name is the most prominent part of a web address. Typically, different pages on the same site will continue to use the same domain name. For example, all pages on this site share the GCFLearnFree.org domain name.



Each segment of the domain name separated by a period is called a domain. The domain on the right is called a top-level domain, with the domain to the left of it called the second-level domain, then third-level domain, and so on.

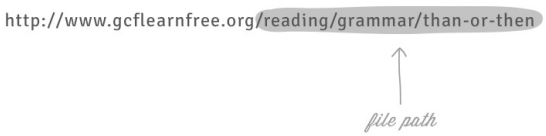


You can often learn something about the site from the domains. One of the domains usually identifies the organization, while the top-level domain may give you more general information on what kind of site it is. For example, in the domain name nc.gov, the .gov domain means it is a government website in the United States, the nc domain identifies it as the website of North Carolina.

In most URLs, the www domain can be omitted. Google.com and www.google.com lead to the same page. However, other subdomains cannot be omitted. For example, all pages under news.google.com require the news subdomain in the URL.

File path

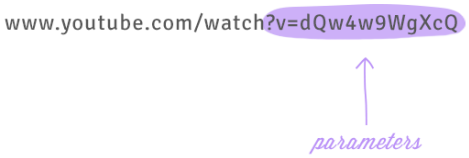
The file path—often just called the path—tells your browser to load a specific page. If you don't specify a path and only enter a domain name, your browser is still loading a specific page; it's just loading a default page, which usually will help you navigate to other pages.



URLs that end with the domain name without a file path usually will load a homepage or an index page that›s designed to help you navigate to specific pages on the site. Often, if you can't remember the file path for a specific page, you can go to the homepage and search for it.

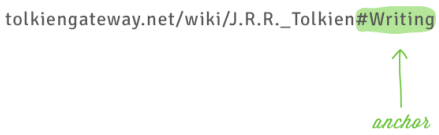
Parameters

Some URLs include a string of characters after the path—beginning with a question mark—called the parameter string. You have probably noticed this part of a URL appear in your address bar after performing a search on Google or YouTube. The parameter string can be clear or confusing to a human user, but it is critical information for the server.



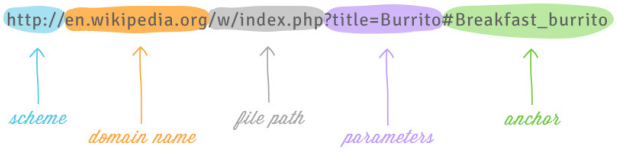
Anchor

Also appearing after the path, the anchor tells your browser to scroll to or load a specific part of the page. Usually the anchor begins with a hashtag and is used to direct your browser to a specific part of a very long page, much like a bookmark. Different anchors don't load different pages; they simply tell the browser to display different parts of the page.



A whole URL

When combined, these elements make up a URL, although not all URLs will have all five parts.



Using what we learned section, we can tell from the domain name of this URL that it's on Wikipedia, from the parameters that the page is probably about burritos, and from the anchor that we'll be looking at the section on burritos for breakfast.

Keep in mind that these are the most common components of a URL. There are many other parts, but these are the five you will see most often and that can usually give you the most information.

How to use URLs

Any URL can be typed right inside the browser's address bar to get to the resource behind it. But this is only the tip of the iceberg!

The HTML language — which will be discussed later on — makes extensive use of URLs:

- to create links to other documents with the <a> element;
- to link a document with its related resources through various elements such as <link> or <script>;
- to display media such as images (with the <img> element), videos (with the <video> element), sounds and music (with the <audio> element), etc.;
- to display other HTML documents with the <iframe> element.

Note: When specifying URLs to load resources as part of a page (such as when using the <script>, <audio>, <img>, <video>, and the like), you should generally only use HTTP and HTTPS URLs, with few exceptions. Using FTP, for example, is not secure and is no longer supported by modern browsers.

Other technologies, such as CSS or JavaScript, use URLs extensively, and these are really the heart of the Web.

Absolute URLs vs relative URLs

What we saw above is called an *absolute URL*, but there is also something called a *relative URL*. The URL standard defines both — though it uses the terms *absolute URL string* and *relative URL string*, to distinguish them from URL objects (which are in-memory representations of URLs).

Let's examine what the distinction between *absolute* and *relative* means in the context of URLs.

The required parts of a URL depend to a great extent on the context in which the URL is used. In your browser's address bar, a URL doesn't have any context, so you must provide a full (or *absolute*) URL, like the ones we saw above. You don't need to include the protocol (the browser uses HTTP by default) or the port (which is only required when the targeted Web server is using some unusual port), but all the other parts of the URL are necessary.

When a URL is used within a document, such as in an HTML page, things are a bit different. Because the browser already has the document's own URL, it can use this information to fill in the missing parts of any URL available inside that document. We can differentiate between an *absolute URL* and a *relative URL* by looking only at the *path* part of the URL. If the path part of the URL starts with the «/» character, the browser will fetch that resource from the top root of the server, without reference to the context given by the current document.

Let's look at some examples to make this clearer.

Examples of absolute URLs

Full URL (the same as the one we used before)	https://developer.mozilla.org/en-US/docs/Learn
Implicit protocol	//developer.mozilla.org/en-US/docs/Learn In this case, the browser will call that URL with the same protocol as the one used to load the document hosting that URL.
Implicit domain name	/en-US/docs/Learn This is the most common use case for an absolute URL within an HTML document. The browser will use the same protocol and the same domain name as the one used to load the document hosting that URL. **Note:** *it isn't possible to omit the domain name without omitting the protocol as well.*

Examples of relative URLs

To better understand the following examples, let's assume that the URLs are called from within the document located at the following URL: https://developer.mozilla.org/en-US/docs/Learn

Sub-resources	Skills/Infrastructure/Understanding_URLs Because that URL does not start with /, the browser will attempt to find the document in a subdirectory of the one containing the current resource. So in this example, we really want to reach this URL: https://developer.mozilla.org/en-US/docs/Learn/Skills/Infrastructure/Understanding_URLs.
Going back in the directory tree	../CSS/display In this case, we use the ../ writing convention — inherited from the UNIX file system world — to tell the browser we want to go up from one directory. Here we want to reach this URL: https://developer.mozilla.org/en-US/docs/Learn/../CSS/display, which can be simplified to: https://developer.mozilla.org/en-US/docs/CSS/display.

Semantic URLs

Despite their very technical flavor, URLs represent a human-readable entry point for a Web site. They can be memorized, and anyone can enter them into a browser's address bar. People are at the core of the Web, and so it is considered best practice to build what is called *semantic URLs*. Semantic URLs use words with inherent meaning that can be understood by anyone, regardless of their technical know-how.

Linguistic semantics are of course irrelevant to computers. You've probably often seen URLs that look like mashups of random characters. But there are many advantages to creating human-readable URLs:

- It is easier for you to manipulate them.
- It clarifies things for users in terms of where they are, what they're doing, what they're reading or interacting with on the Web.
- Some search engines can use those semantics to improve the classification of the associated pages.

URNs

A uniform resource name (URN) is an Internet resource with a static name that remains valid even if its data is moved to another location. Unlike a uniform resource locator (URL), which cannot work if the content is moved, a URN is always able to track the resource of certain data on the Web, hence resolving a frequent issue of moving of data.

With a Uniform Resource Name, various items (resources) can be given a unique name. This is especially of interest on the internet, as there, different people and applications encounter each other – and in doing so make exchanges related to specific objects. With a clear identifier, you can be sure that all participants are talking about the same content.

For this purpose, however, you must ensure that object identification functions reliably and does not change – that is to say, it is independent of time and place. The URN is perfectly suited for guaranteeing this, as it is designed to be location-independent and have no time limit. This means that even if you moves the data and stores it in another location, the name of the object won't change.

Fact: Although it's true that various resource types can theoretically be addressed via URN, in practice one primarily uses the format for media. In libraries especially, URNs have been established in order to be able to clearly identify books and other content. In doing so, additional information can be utilized, such as ISBNs.

The Structure of a URN

The URN is rendered in URI syntax. Here the URN is provided with a specific scheme – the name for the first part of the syntax. This is followed by information that is separated by a colon.

urn:<nid>:<nss></nss></nid>

Each URN thus consists of at least three parts.

- Each URN begins with the URN's scheme specification.
- Then the namespace identifier (NID) follows, which must

be registered with IANA and (separated with additional colons) can be subdivided.

- Finally, one finds the namespace-specific string (NSS), which then precisely identifies the particular object.

The namespace being used is managed by the Internet Assigned Numbers Authority (IANA). The official list contains more than 60 different namespaces. The most well-known include:

- **issn**: *International Standard Serial Number*. A unique number for serials, such as periodicals.
- **isbn**: *International Standard. Books Number*. A unique identifier for books.
- **nbn**: *National Bibliography Number*. A country-specific identification format for some national libraries.
- **uuid**: *Universally Unique Identifier*. A unique information identifier within a computer system.
- **isan**: *International Standard Audiovisual Number*. A unique number for audio-visual works (e.g. movies) that is independent of the release format.

The NBN serves as a good example. An URN with this namespace could for example look like this:

urn:nbn:de:101:1-201907280240175770291 3

One can recognize in the identifier that it is a matter of a URN being included in the NBN namespace. This however is specified further: With "de," the country code (which stands for Germany - "Deutschland") is established. A subnamespace identifier (SNID) then follows that denotes the German National Library. Only then does the identifier of the actual work follow, with a long, unique number.

URN, URL and URI: The Differences Explained

The Uniform Resource Names shares some similarities with the Uniform Resource Locator (URL). Both the URN and URL are subcategories of the Uniform Resource Identifier (URI). This identifier was conceived so that every form of resource – regardless of whether it is a website or email – can be identified and interacted with in a network such as the internet. This

also applies to URL and URN; and in spite of their respective particularities, both subcategories follow the URI scheme.

However, URN and URL differentiate themselves in that the former permanently identifies a resource and the latter primarily specifies the path to the resource. While the URN is designed in such a way that the identifier is independent of where the data is stored, URLs point to a specific place where the resource is located. Both identifier formats are thus similar to each other in their design, but serve different purposes.

Mime

Multipurpose Internet Mail Extension (MIME) is a standard that was proposed by Bell Communications in 1991 in order to expand the limited capabilities of email.

MIME is a kind of add-on *or a supplementary protocol* that allows non-ASCII data to be sent through SMTP. It allows the users to exchange different kinds of data files on the Internet: audio, video, images, application programs as well.

Why do we need MIME?

Limitations of Simple Mail Transfer Protocol (SMTP):

- SMTP has a very simple structure
- Its simplicity however comes with a price as it only sends messages in NVT 7-bit ASCII format.
- It cannot be used for languages that do not support 7-bit ASCII format such as French, German, Russian, Chinese and Japanese, etc. so it cannot be transmitted using SMTP. So, in order *to make SMTP more broad, we use MIME*.
- It cannot be used to send binary files or video or audio data.

Purpose and Functionality of MIME

Growing demand for Email Messages as people also want to express themselves in terms of Multimedia. So, MIME another

email application is introduced as it is not restricted to textual data.

MIME *transforms non-ASCII data* at the sender side to NVT 7-bit data and delivers it to the client SMTP. The message on the receiver side is transferred back to the original data. As well as we can send video and audio data using MIME as it transfers them also in 7-bit ASCII data.

Features of MIME

- It is able to send multiple attachments with a single message.
- Unlimited message length.
- Binary attachments (executables, images, audio, or video files) may be divided if needed.
- MIME provided support for varying content types and multi-part messages.

Working of MIME

Suppose a user wants to send an email through a user agent and it is in a non-ASCII format so there is a MIME protocol that converts it into 7-bit NVT ASCII format. The message is transferred through the e-mail system to the other side in the 7-bit format now MIME protocol again converts it back into non-ASCII code and now the user agent of the receiver side reads it and then information is finally read by the receiver. MIME header is basically inserted at the beginning of any e-mail transfer.

MIME with SMTP and POP

SMTP transfers the mail being a message transfer agent from the sender's side to the mailbox of the receiver side and stores it and MIME header is added to the original header and provides additional information. while POP being the message access agent organizes the mails from the mail server to the receiver's computer. POP allows the user agent to connect with the message transfer agent.

MIME Header

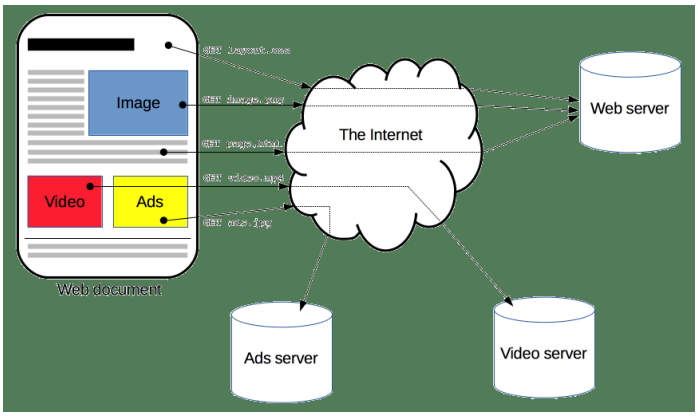
It is added to the original e-mail header section to define transformation. There are *five headers* that we add to the original header:

- **MIME-Version –** Defines the version of the MIME protocol. It must have the parameter *Value 1.0*, which indicates that message is formatted using MIME.
- **Content-Type –** Type of data used in the body of the message. They are of different types like text data (plain, HTML), audio content, or video content.
- **Content-Type Encoding –** It defines the method used for encoding the message. Like 7-bit encoding, 8-bit encoding, etc.
- **Content Id –** It is used for uniquely identifying the message.
- **Content description –** It defines whether the body is actually an image, video, or audio.

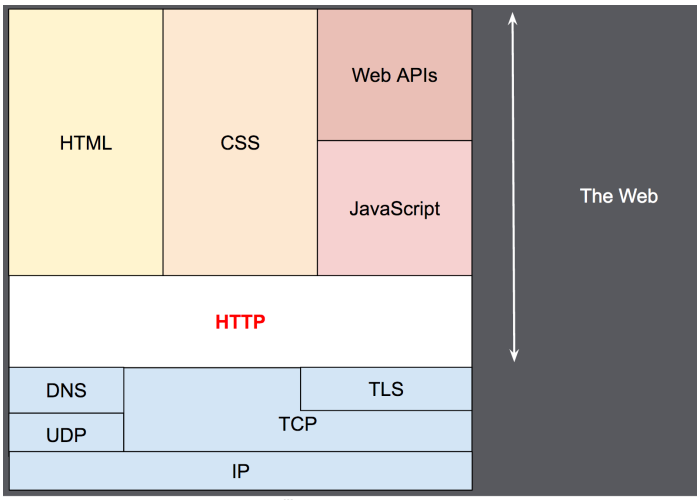
HTTP OVERVIEW

HTTP is a protocol for fetching resources such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.

A complete document is reconstructed from the different sub-documents fetched, for instance, text, layout description, images, videos, scripts, and more.



Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client, usually a Web browser, are called *requests* and the messages sent by the server as an answer are called *responses*.

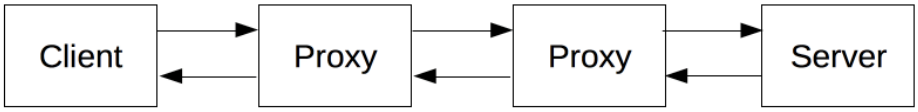


Designed in the early 1990s, HTTP is an extensible protocol which has evolved over time. It is an application layer protocol that is sent over TCP, or over a TLS-encrypted TCP connection, though any reliable transport protocol could theoretically be used. Due to its extensibility, it is used to not only fetch hypertext documents, but also images and videos or to post content to servers, like with HTML form results. HTTP can also be used to fetch parts of documents to update Web pages on demand.

Components of HTTP-based systems

HTTP is a client-server protocol: requests are sent by one entity, the user-agent (or a proxy on behalf of it). Most of the time the user-agent is a Web browser, but it can be anything, for example, a robot that crawls the Web to populate and maintain a search engine index.

Each individual request is sent to a server, which handles it and provides an answer called the *response*. Between the client and the server there are numerous entities, collectively called proxies, which perform different operations and act as gateways or caches, for example.



In reality, there are more computers between a browser and the server handling the request: there are routers, modems, and more. Thanks to the layered design of the Web, these are hidden in the network and transport layers. HTTP is on top, at the application layer. Although important for diagnosing network problems, the underlying layers are mostly irrelevant to the description of HTTP.

Client: the user-agent

The *user-agent* is any tool that acts on behalf of the user. This role is primarily performed by the Web browser, but it may also be performed by programs used by engineers and Web developers to debug their applications.

The browser is **always** the entity initiating the request. It is never the server (though some mechanisms have been added over the years to simulate server-initiated messages).

To display a Web page, the browser sends an original request to fetch the HTML document that represents the page. It then parses this file, making additional requests corresponding to execution scripts, layout information (CSS) to display, and sub-resources contained within the page (usually images and videos). The Web browser then combines these resources to present the complete document, the Web page. Scripts executed by the browser can fetch more resources in later phases and the browser updates the Web page accordingly.

A Web page is a hypertext document. This means some parts of the displayed content are links, which can be activated (usually by a click of the mouse) to fetch a new Web page, allowing the user to direct their user-agent and navigate through the Web. The browser translates these directions into HTTP requests, and further interprets the HTTP responses to present the user with a clear response.

The Web server

On the opposite side of the communication channel is the server, which *serves* the document as requested by the client. A server appears as only a single machine virtually; but it may actually be a collection of servers sharing the load (load balancing), or a complex piece of software interrogating other computers (like cache, a DB server, or e-commerce servers), totally or partially generating the document on demand.

A server is not necessarily a single machine, but several server software instances can be hosted on the same machine. With HTTP/1.1 and the Host header, they may even share the same IP address.

Proxies

Between the Web browser and the server, numerous computers and machines relay the HTTP messages. Due to the layered structure of the Web stack, most of these operate at the transport, network or physical levels, becoming transparent at the HTTP layer and potentially having a significant impact on performance. Those operating at the application layers are generally called **proxies**. These can be transparent, forwarding on the requests they receive without altering them in any way, or non-transparent, in which case they will change the request in some way before passing it along to the server. Proxies may perform numerous functions:

- caching (the cache can be public or private, like the browser cache)
- filtering (like an antivirus scan or parental controls)
- load balancing (to allow multiple servers to serve different requests)
- authentication (to control access to different resources)
- logging (allowing the storage of historical information)

Basic aspects of HTTP

HTTP is simple

HTTP is generally designed to be simple and human readable,

even with the added complexity introduced in HTTP/2 by encapsulating HTTP messages into frames. HTTP messages can be read and understood by humans, providing easier testing for developers, and reduced complexity for newcomers.

HTTP is extensible

Introduced in HTTP/1.0, HTTP headers make this protocol easy to extend and experiment with. New functionality can even be introduced by a simple agreement between a client and a server about a new header›s semantics.

HTTP is stateless, but not sessionless

HTTP is stateless: there is no link between two requests being successively carried out on the same connection. This immediately has the prospect of being problematic for users attempting to interact with certain pages coherently, for example, using e-commerce shopping baskets. But while the core of HTTP itself is stateless, HTTP cookies allow the use of stateful sessions. Using header extensibility, HTTP Cookies are added to the workflow, allowing session creation on each HTTP request to share the same context, or the same state.

HTTP and connections

A connection is controlled at the transport layer, and therefore fundamentally out of scope for HTTP. HTTP doesn't require the underlying transport protocol to be connection-based; it only requires it to be *reliable*, or not lose messages (at minimum, presenting an error in such cases). Among the two most common transport protocols on the Internet, TCP is reliable and UDP isn't. HTTP therefore relies on the TCP standard, which is connection-based.

Before a client and server can exchange an HTTP request/response pair, they must establish a TCP connection, a process which requires several round-trips. The default behavior of HTTP/1.0 is to open a separate TCP connection for each HTTP request/response pair. This is less efficient than sharing a single TCP connection when multiple requests are sent in close succession.

In order to mitigate this flaw, HTTP/1.1 introduced *pipelining* (which proved difficult to implement) and *persistent connections*: the underlying TCP connection can be partially controlled using the Connection header. HTTP/2 went a step further by multiplexing messages over a single connection, helping keep the connection warm and more efficient.

Experiments are in progress to design a better transport protocol more suited to HTTP. For example, Google is experimenting with QUIC which builds on UDP to provide a more reliable and efficient transport protocol.

What can be controlled by HTTP

This extensible nature of HTTP has, over time, allowed for more control and functionality of the Web. Cache and authentication methods were functions handled early in HTTP history. The ability to relax the *origin constraint*, by contrast, was only added in the 2010s.

Here is a list of common features controllable with HTTP:

- *Caching* How documents are cached can be controlled by HTTP. The server can instruct proxies and clients about what to cache and for how long. The client can instruct intermediate cache proxies to ignore the stored document.
- *Relaxing the origin constraint* To prevent snooping and other privacy invasions, Web browsers enforce strict separation between Web sites. Only pages from the same origin can access all the information of a Web page. Though such a constraint is a burden to the server, HTTP headers can relax this strict separation on the server side, allowing a document to become a patchwork of information sourced from different domains; there could even be security-related reasons to do so.
- *Authentication* Some pages may be protected so that only specific users can access them. Basic authentication may be provided by HTTP, either using the WWW-Authenticate and similar headers, or by setting a specific session using HTTP cookies.
- *Proxy and tunneling* Servers or clients are often located

on intranets and hide their true IP address from other computers. HTTP requests then go through proxies to cross this network barrier. Not all proxies are HTTP proxies. The SOCKS protocol, for example, operates at a lower level. Other protocols, like ftp, can be handled by these proxies.

- *Sessions* Using HTTP cookies allows you to link requests with the state of the server. This creates sessions, despite basic HTTP being a state-less protocol. This is useful not only for e-commerce shopping baskets, but also for any site allowing user configuration of the output.

HTTP flow

When a client wants to communicate with a server, either the final server or an intermediate proxy, it performs the following steps:

1. Open a TCP connection: The TCP connection is used to send a request, or several, and receive an answer. The client may open a new connection, reuse an existing connection, or open several TCP connections to the servers.
2. Send an HTTP message: HTTP messages (before HTTP/2) are human-readable. With HTTP/2, these simple messages are encapsulated in frames, making them impossible to read directly, but the principle remains the same. For example:
3. GET / HTTP/1.1
4. Host: developer.mozilla.org
5. Accept-Language: fr

Copy to Clipboard

6. Read the response sent by the server, such as:
7. HTTP/1.1 200 OK
8. Date: Sat, 09 Oct 2010 14:28:02 GMT
9. Server: Apache
10. Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
11. ETag: “51142bc1-7449-479b075b2891b”
12. Accept-Ranges: bytes

13. Content-Length: 29769
14. Content-Type: text/html
15.
16.<!DOCTYPE html>… (here come the 29769 bytes of the requested web page)

Copy to Clipboard

17. Close or reuse the connection for further requests.

If HTTP pipelining is activated, several requests can be sent without waiting for the first response to be fully received. HTTP pipelining has proven difficult to implement in existing networks, where old pieces of software coexist with modern versions. HTTP pipelining has been superseded in HTTP/2 with more robust multiplexing requests within a frame.

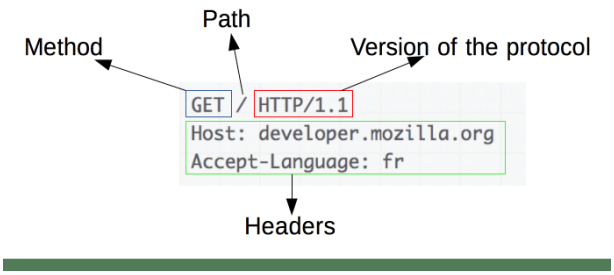
HTTP Messages

HTTP messages, as defined in HTTP/1.1 and earlier, are human-readable. In HTTP/2, these messages are embedded into a binary structure, a *frame*, allowing optimizations like compression of headers and multiplexing. Even if only part of the original HTTP message is sent in this version of HTTP, the semantics of each message is unchanged and the client reconstitutes (virtually) the original HTTP/1.1 request. It is therefore useful to comprehend HTTP/2 messages in the HTTP/1.1 format.

There are two types of HTTP messages, requests and responses, each with its own format.

Requests

An example HTTP request:

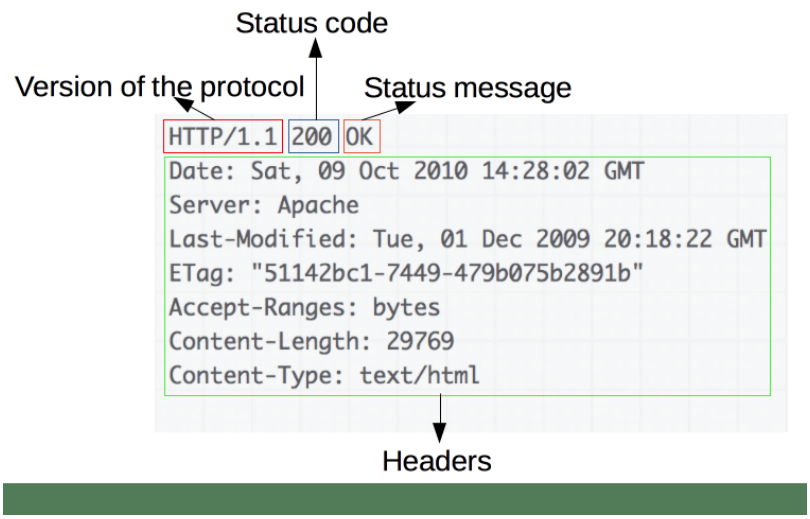


Requests consist of the following elements:

- An HTTP method, usually a verb like GET, POST, or a noun like OPTIONS or HEAD that defines the operation the client wants to perform. Typically, a client wants to fetch a resource (using GET) or post the value of an HTML form (using POST), though more operations may be needed in other cases.
- The path of the resource to fetch; the URL of the resource stripped from elements that are obvious from the context, for example without the protocol (http://), the domain (here, developer.mozilla.org), or the TCP port (here, 80).
- The version of the HTTP protocol.
- Optional headers that convey additional information for the servers.
- A body, for some methods like POST, similar to those in responses, which contain the resource sent.

Responses

An example response:



Responses consist of the following elements:

- The version of the HTTP protocol they follow.
- A status code, indicating if the request was successful

or not, and why.

- A status message, a non-authoritative short description of the status code.
- HTTP headers, like those for requests.
- Optionally, a body containing the fetched resource.

APIs based on HTTP

The most commonly used API based on HTTP is the XMLHttpRequest API, which can be used to exchange data between a user agent and a server. The modern Fetch API provides the same features with a more powerful and flexible feature set.

Another API, server-sent events, is a one-way service that allows a server to send events to the client, using HTTP as a transport mechanism. Using the EventSource interface, the client opens a connection and establishes event handlers. The client browser automatically converts the messages that arrive on the HTTP stream into appropriate Event objects. Then it delivers them to the event handlers that have been registered for the events› type if known, or to the onmessage event handler if no type-specific event handler was established.

HTTP is an extensible protocol that is easy to use. The client-server structure, combined with the ability to add headers, allows HTTP to advance along with the extended capabilities of the Web.

Though HTTP/2 adds some complexity by embedding HTTP messages in frames to improve performance, the basic structure of messages has stayed the same since HTTP/1.0. Session flow remains simple, allowing it to be investigated and debugged with a simple HTTP message monitor.

HTTP GENERAL HEADER

HTTP header fields provide required information about the request or response, or about the object sent in the message body. There are four types of HTTP message headers:

- **General-header:** These header fields have general

applicability for both request and response messages.

- **Client Request-header:** These header fields have applicability only for request messages.
- **Server Response-header:** These header fields have applicability only for response messages.
- **Entity-header:** These header fields define meta information about the entity-body or, if no body is present, about the resource identified by the request.

"Website without visitors is like a ship lost in the horizon."

—Dr. Christopher Dayagdag

General Headers

Cache-Control

The Cache-Control general-header field is used to specify directives that MUST be obeyed by all the caching system. The syntax is as follows:

Cache-Control : cache-request-directive | cache-response-directive

An HTTP client or server can use the **Cache-control** general header to specify parameters for the cache or to request certain kinds of documents from the cache. The caching directives are specified in a comma-separated list. For example:

Cache-control: no-cache

The following table lists the important cache request directives that can be used by the client in its HTTP request:

S.N.	Cache Request Directive and Description
1	**no-cache** A cache must not use the response to satisfy a subsequent request without successful revalidation with the origin server.
2	**no-store** The cache should not store anything about the client request or server response.
3	**max-age = seconds** Indicates that the client is willing to accept a response whose age is not greater than the specified time in seconds.
4	**max-stale [= seconds]** Indicates that the client is willing to accept a response that has exceeded its expiration time. If seconds are given, it must not be expired by more than that time.
5	**min-fresh = seconds** Indicates that the client is willing to accept a response whose freshness lifetime is not less than its current age plus the specified time in seconds.
6	**no-transform** Does not convert the entity-body.
7	**only-if-cached** Does not retrieve new data. The cache can send a document only if it is in the cache, and should not contact the origin-server to see if a newer copy exists.

The following important cache response directives that can be used by the server in its HTTP response:

S.N.	Cache Response Directive and Description
1	**public** Indicates that the response may be cached by any cache.
2	**private** Indicates that all or part of the response message is intended for a single user and must not be cached by a shared cache.
3	**no-cache** A cache must not use the response to satisfy a subsequent request without successful re-validation with the origin server.
4	**no-store** The cache should not store anything about the client request or server response.
5	**no-transform** Does not convert the entity-body.

6	**must-revalidate** The cache must verify the status of the stale documents before using it and expired ones should not be used.
7	**proxy-revalidate** The proxy-revalidate directive has the same meaning as the must- revalidate directive, except that it does not apply to non-shared user agent caches.
8	**max-age = seconds** Indicates that the client is willing to accept a response whose age is not greater than the specified time in seconds.
9	**s-maxage = seconds** The maximum age specified by this directive overrides the maximum age specified by either the max-age directive or the Expires header. The s-maxage directive is always ignored by a private cache.

Connection

The Connection general-header field allows the sender to specify options that are desired for that particular connection and must not be communicated by proxies over further connections. Following is the simple syntax for using connection header:

Connection : "Connection"

HTTP/1.1 defines the "close" connection option for the sender to signal that the connection will be closed after completion of the response. For example:

Connection: close

By default, HTTP 1.1 uses persistent connections, where the connection does not automatically close after a transaction. HTTP 1.0, on the other hand, does not have persistent connections by default. If a 1.0 client wishes to use persistent connections, it uses the **keep-alive** parameter as follows:

Connection: keep-alive

Date

All HTTP date/time stamps MUST be represented in Greenwich Mean Time (GMT), without exception. HTTP applications are allowed to use any of the following three representations of date/time stamps:

Sun, 06 Nov 1994 08:49:37 GMT ; RFC 822, updated by RFC 1123

Sunday, 06-Nov-94 08:49:37 GMT ; RFC 850, obsoleted by RFC 1036

Sun Nov 6 08:49:37 1994 ; ANSI C's asctime() format

Here the first format is the most preferred one.

Pragma

The Pragma general-header field is used to include implementation specific directives that might apply to any recipient along the request/response chain. For example:

Pragma: no-cache

The only directive defined in HTTP/1.0 is the no-cache directive and is maintained in HTTP 1.1 for backward compatibility. No new Pragma directives will be defined in the future.

Trailer

The Trailer general field value indicates that the given set of header fields is present in the trailer of a message encoded with chunked transfer-coding. Following is the syntax of Trailer header field:

Trailer : field-name

Message header fields listed in the Trailer header field must not include the following header fields:

- Transfer-Encoding
- Content-Length
- Trailer

Transfer-Encoding

The *Transfer-Encoding* general-header field indicates what type of transformation has been applied to the message body in order to safely transfer it between the sender and the recipient. This is not the same as content-encoding because transfer-encodings

are a property of the message, not of the entity-body. The syntax of Transfer-Encoding header field is as follows:

Transfer-Encoding: chunked

All transfer-coding values are case-insensitive.

Upgrade

The *Upgrade* general-header allows the client to specify what additional communication protocols it supports and would like to use if the server finds it appropriate to switch protocols. For example:

Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11

The Upgrade header field is intended to provide a simple mechanism for transition from HTTP/1.1 to some other, incompatible protocol.

Via

The *Via* general-header must be used by gateways and proxies to indicate the intermediate protocols and recipients. For example, a request message could be sent from an HTTP/1.0 user agent to an internal proxy code-named «fred", which uses HTTP/1.1 to forward the request to a public proxy at nowhere.com, which completes the request by forwarding it to the origin server at www.ics.uci.edu. The request received by www.ics.uci.edu would then have the following Via header field:

Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)

The Upgrade header field is intended to provide a simple mechanism for transition from HTTP/1.1 to some other, incompatible protocol.

Warning

The *Warning* general-header is used to carry additional information about the status or transformation of a message which might not be reflected in the message. A response may carry more than one Warning header.

Warning : warn-code SP warn-agent SP warn-text SP warn-date

Client Request Headers

Accept

The *Accept* request-header field can be used to specify certain media types which are acceptable for the response. The general syntax is as follows:

Accept: type/subtype [q=qvalue]

Multiple media types can be listed separated by commas and the optional qvalue represents an acceptable quality level for accept types on a scale of 0 to 1. Following is an example:

Accept: text/plain; q=0.5, text/html, text/x-dvi; q=0.8, text/x-c

This would be interpreted as **text/html** and **text/x-c** and are the preferred media types, but if they do not exist, then send the **text/x-dvi** entity, and if that does not exist, send the **text/plain** entity.

Accept-Charset

The *Accept-Charset* request-header field can be used to indicate what character sets are acceptable for the response. Following is the general syntax:

Accept-Charset: character_set [q=qvalue]

Multiple character sets can be listed separated by commas and the optional qvalue represents an acceptable quality level for nonpreferred character sets on a scale of 0 to 1. Following is an example:

Accept-Charset: iso-8859-5, unicode-1-1; q=0.8

The special value "*", if present in the **Accept-Charset** field, matches every character set and if no **Accept-Charset** header is present, the default is that any character set is acceptable.

Accept-Encoding

The *Accept-Encoding* request-header field is similar to Accept, but restricts the content-codings that are acceptable in the response. The general syntax is:

Accept-Encoding: encoding types

Examples are as follows:

Accept-Encoding: compress, gzip

Accept-Encoding:

Accept-Encoding: *

Accept-Encoding: compress;q=0.5, gzip;q=1.0

Accept-Encoding: gzip;q=1.0, identity; q=0.5, *;q=0

Accept-Language

The *Accept-Language* request-header field is similar to Accept, but restricts the set of natural languages that are preferred as a response to the request. The general syntax is:

Accept-Language: language [q=qvalue]

Multiple languages can be listed separated by commas and the optional qvalue represents an acceptable quality level for non preferred languages on a scale of 0 to 1. Following is an example:

Accept-Language: da, en-gb;q=0.8, en;q=0.7

Authorization

The *Authorization* request-header field value consists of credentials containing the authentication information of the user agent for the realm of the resource being requested. The general syntax is:

Authorization : credentials

The HTTP/1.0 specification defines the BASIC authorization scheme, where the authorization parameter is the string of **username:password** encoded in base 64. Following is an

example:

Authorization: BASIC Z3Vlc3Q6Z3Vlc3QxMjM=

The value decodes into is **guest:guest123** where **guest** is user ID and **guest123** is the password.

Cookie

The *Cookie* request-header field value contains a name/value pair of information stored for that URL. Following is the general syntax:

Cookie: name=value

Multiple cookies can be specified separated by semicolons as follows:

Cookie: name1=value1;name2=value2;name3=value3

Expect

The *Expect* request-header field is used to indicate that a particular set of server behaviors is required by the client. The general syntax is:

Expect : 100-continue | expectation-extension

If a server receives a request containing an Expect field that includes an expectation-extension that it does not support, it must respond with a 417 (Expectation Failed) status.

From

The *From* request-header field contains an Internet e-mail address for the human user who controls the requesting user agent. Following is a simple example:

From: webmaster@w3.org

This header field may be used for logging purposes and as a means for identifying the source of invalid or unwanted requests.

Host

The *Host* request-header field is used to specify the Internet host and the port number of the resource being requested. The general syntax is:

Host : "Host" ":" host [":" port] ;

A **host** without any trailing port information implies the default port, which is 80. For example, a request on the origin server for *http://www.w3.org/pub/WWW/* would be:

GET /pub/WWW/ HTTP/1.1

Host: www.w3.org

If-Match

The *If-Match* request-header field is used with a method to make it conditional. This header requests the server to perform the requested method only if the given value in this tag matches the given entity tags represented by **ETag**. The general syntax is:

If-Match : entity-tag

An asterisk (*) matches any entity, and the transaction continues only if the entity exists. Following are possible examples:

If-Match: "xyzzy"

If-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"

If-Match: *

If none of the entity tags match, or if "*" is given and no current entity exists, the server must not perform the requested method, and must return a 412 (Precondition Failed) response.

If-Modified-Since

The *If-Modified-Since* request-header field is used with a method to make it conditional. If the requested URL has not been modified since the time specified in this field, an entity will not be returned from the server; instead, a 304 (not modified) response will be returned without any message-body. The general syntax of if-modified-since is:

If-Modified-Since : HTTP-date

An example of the field is:

If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT

If none of the entity tags match, or if "*" is given and no current entity exists, the server must not perform the requested method, and must return a 412 (Precondition Failed) response.

If-None-Match

The *If-None-Match* request-header field is used with a method to make it conditional. This header requests the server to perform the requested method only if one of the given value in this tag matches the given entity tags represented by **ETag**. The general syntax is:

If-None-Match : entity-tag

An asterisk (*) matches any entity, and the transaction continues only if the entity does not exist. Following are the possible examples:

If-None-Match: "xyzzy"

If-None-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"

If-None-Match: *

If-Range

The *If-Range* request-header field can be used with a conditional GET to request only the portion of the entity that is missing, if it has not been changed, and the entire entity if it has been changed. The general syntax is as follows:

If-Range : entity-tag | HTTP-date

Either an entity tag or a date can be used to identify the partial entity already received. For example:

If-Range: Sat, 29 Oct 1994 19:43:31 GMT

Here if the document has not been modified since the given date, the server returns the byte range given by the Range header, otherwise it returns all of the new document.

If-Unmodified-Since

The *If-Unmodified-Since* request-header field is used with a method to make it conditional. The general syntax is:

If-Unmodified-Since : HTTP-date

If the requested resource has not been modified since the time specified in this field, the server should perform the requested operation as if the If-Unmodified-Since header were not present. For example:

If-Unmodified-Since: Sat, 29 Oct 1994 19:43:31 GMT

If the request results in anything other than a 2xx or 412 status, the *If-Unmodified-Since* header should be ignored.

Max-Forwards

The *Max-Forwards* request-header field provides a mechanism with the TRACE and OPTIONS methods to limit the number of proxies or gateways that can forward the request to the next inbound server. Here is the general syntax:

Max-Forwards : n

The Max-Forwards value is a decimal integer indicating the remaining number of times this request message may be forwarded. This is useful for debugging with the TRACE method, avoiding infinite loops. For example:

Max-Forwards : 5

The Max-Forwards header field may be ignored for all other methods defined in the HTTP specification.

Proxy-Authorization

The *Proxy-Authorization* request-header field allows the client to identify itself (or its user) to a proxy which requires authentication. Here is the general syntax:

Proxy-Authorization : credentials

The Proxy-Authorization field value consists of credentials containing the authentication information of the user agent for

the proxy and/or realm of the resource being requested.

Range

The *Range* request-header field specifies the partial range(s) of the content requested from the document. The general syntax is:

Range: bytes-unit=first-byte-pos "-" [last-byte-pos]

The first-byte-pos value in a byte-range-spec gives the byte-offset of the first byte in a range. The last-byte-pos value gives the byte-offset of the last byte in the range; that is, the byte positions specified are inclusive. You can specify a byte-unit as bytes. Byte offsets start at zero. Some simple examples are as follows:

- The first 500 bytes

Range: bytes=0-499

- The second 500 bytes

Range: bytes=500-999

- The final 500 bytes

Range: bytes=-500

- The first and last bytes only

Range: bytes=0-0,-1

Multiple ranges can be listed, separated by commas. If the first digit in the comma-separated byte range(s) is missing, the range is assumed to count from the end of the document. If the second digit is missing, the range is byte n to the end of the document.

Referer

The *Referer* request-header field allows the client to specify the address (URI) of the resource from which the URL has been requested. The general syntax is as follows:

Referer : absoluteURI | relativeURI

Following is a simple example:

Referer: http://www.tutorialspoint.org/http/index.htm

If the field value is a relative URI, it should be interpreted relative to the *Request-URI*.

TE

The *TE* request-header field indicates what extension *transfer-coding* it is willing to accept in the response and whether or not it is willing to accept trailer fields in a chunked *transfer-coding*. Following is the general syntax:

TE : t-codings

The presence of the keyword "trailers" indicates that the client is willing to accept trailer fields in a chunked transfer-coding and it is specified either of the ways:

TE: deflate

TE:

TE: trailers, deflate;q=0.5

If the TE field-value is empty or if no TE field is present, then only transfer-coding is *chunked*. A message with no transfer-coding is always acceptable.

User-Agent

The *User-Agent* request-header field contains information about the user agent originating the request. Following is the general syntax:

User-Agent : product | comment

Example:

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Server Response Headers

Accept-Ranges

The *Accept-Ranges* response-header field allows the server to indicate its acceptance of range requests for a resource. The general syntax is:

Accept-Ranges : range-unit | none

For example a server that accepts byte-range requests may send:

Accept-Ranges: bytes

Servers that do not accept any kind of range request for a resource may send:

Accept-Ranges: none

This will advise the client not to attempt a range request.

Age

The *Age* response-header field conveys the sender›s estimate of the amount of time since the response (or its revalidation) was generated at the origin server. The general syntax is:

Age : delta-seconds

Age values are non-negative decimal integers, representing time in seconds. Following is a simple example:

Age: 1030

An HTTP/1.1 server that includes a cache must include an Age header field in every response generated from its own cache.

ETag

The *ETag* response-header field provides the current value of the entity tag for the requested variant. The general syntax is:

ETag : entity-tag

Here are some simple examples:

ETag: "xyzzy"

ETag: W/"xyzzy"

ETag: ""

Location

The *Location* response-header field is used to redirect the recipient to a location other than the Request-URI for completion. The general syntax is:

Location : absoluteURI

Following is a simple example:

Location: http://www.tutorialspoint.org/http/index.htm

The Content-Location header field differs from Location in that the Content-Location identifies the original location of the entity enclosed in the request.

Proxy-Authenticate

The *Proxy-Authenticate* response-header field must be included as a part of a 407 (Proxy Authentication Required) response. The general syntax is:

Proxy-Authenticate : challenge

Retry-After

The *Retry-After* response-header field can be used with a 503 (Service Unavailable) response to indicate how long the service is expected to be unavailable to the requesting client. The general syntax is:

Retry-After : HTTP-date | delta-seconds

Examples:

Retry-After: Fri, 31 Dec 1999 23:59:59 GMT

Retry-After: 120

In the latter example, the delay is 2 minutes.

Server

The *Server* response-header field contains information about the software used by the origin server to handle the request. The general syntax is:

Server : product | comment

Following is a simple example:

Server: Apache/2.2.14 (Win32)

If the response is being forwarded through a proxy, the proxy application must not modify the Server response-header.

Set-Cookie

The *Set-Cookie* response-header field contains a name/value pair of information to retain for this URL. The general syntax is:

Set-Cookie: NAME=VALUE; OPTIONS

Set-Cookie response header comprises the token Set-Cookie, followed by a comma-separated list of one or more cookies. Here are the possible values you can specify as options:

S.N.	Options and Description
1	**Comment=comment** This option can be used to specify any comment associated with the cookie.
2	**Domain=domain** The Domain attribute specifies the domain for which the cookie is valid.
3	**Expires=Date-time** The date the cookie will expire. If it is blank, the cookie will expire when the visitor quits the browser.
4	**Path=path** The Path attribute specifies the subset of URLs to which this cookie applies.
5	**Secure** It instructs the user agent to return the cookie only under a secure connection.

Following is an example of a simple cookie header generated by the server:

Set-Cookie: name1=value1,name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT

Vary

The *Vary* response-header field specifies that the entity has multiple sources and may therefore vary according to the specified list of request header(s). Following is the general syntax:

Vary : field-name

You can specify multiple headers separated by commas and a value of asterisk "*" signals that unspecified parameters are not limited to the request-headers. Following is a simple example:

Vary: Accept-Language, Accept-Encoding

Here field names are case-insensitive.

WWW-Authenticate

The *WWW-Authenticate* response-header field must be included in 401 (Unauthorized) response messages. The field value consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable to the Request-URI. The general syntax is:

WWW-Authenticate : challenge

WWW- Authenticate field value might contain more than one challenge, or if more than one WWW-Authenticate header field is provided, the contents of a challenge itself can contain a comma-separated list of authentication parameters. Following is a simple example:

WWW-Authenticate: BASIC realm="Admin"

Entity Headers

Allow

The *Allow* entity-header field lists the set of methods supported by the resource identified by the Request-URI. The general syntax is:

Allow : Method

You can specify multiple methods separated by commas. Following is a simple example:

Allow: GET, HEAD, PUT

This field cannot prevent a client from trying other methods.

Content-Encoding

The *Content-Encoding* entity-header field is used as a modifier to the media-type. The general syntax is:

Content-Encoding : content-coding

The content-coding is a characteristic of the entity identified by the Request-URI. Following is a simple example:

Content-Encoding: gzip

If the content-coding of an entity in a request message is not acceptable to the origin server, the server should respond with a status code of 415 (Unsupported Media Type).

Content-Language

The *Content-Language* entity-header field describes the natural language(s) of the intended audience for the enclosed entity. Following is the general syntax:

Content-Language : language-tag

Multiple languages may be listed for content that is intended for multiple audiences. Following is a simple example:

Content-Language: mi, en

The primary purpose of Content-Language is to allow a user to identify and differentiate entities according to the user's own preferred language.

Content-Length

The *Content-Length* entity-header field indicates the size of the entity-body, in decimal number of OCTETs, sent to the recipient or, in the case of the HEAD method, the size of the entity-body that would have been sent, had the request been a GET. The general syntax is:

Content-Length : DIGITS

Following is a simple example:

Content-Length: 3495

Any Content-Length greater than or equal to zero is a valid value.

Content-Location

The *Content-Location* entity-header field may be used to supply the resource location for the entity enclosed in the message when that entity is accessible from a location separate from the requested resource›s URI. The general syntax is:

Content-Location: absoluteURI | relativeURI

Following is a simple example:

Content-Location: http://www.tutorialspoint.org/http/index.htm

The value of Content-Location also defines the base URI for the entity.

Content-MD5

The *Content-MD5* entity-header field may be used to supply an MD5 digest of the entity for checking the integrity of the message upon receipt. The general syntax is:

Content-MD5 : md5-digest using base64 of 128 bit MD5 digest as per RFC 1864

Following is a simple example:

Content-MD5 : 8c2d46911f3f5a326455f0ed7a8ed3b3

The MD5 digest is computed based on the content of the entity-body, including any content-coding that has been applied, but not including any transfer-encoding applied to the message-body.

Content-Range

The *Content-Range* entity-header field is sent with a partial entity-body to specify where in the full entity-body the partial body should be applied. The general syntax is:

Content-Range : bytes-unit SP first-byte-pos "-" last-byte-pos

Examples of byte-content-range-spec values, assuming that the entity contains a total of 1234 bytes:

- The first 500 bytes:

Content-Range : bytes 0-499/1234

- The second 500 bytes:

Content-Range : bytes 500-999/1234

- All except for the first 500 bytes:

Content-Range : bytes 500-1233/1234

- The last 500 bytes:

Content-Range : bytes 734-1233/1234

When an HTTP message includes the content of a single range, this content is transmitted with a Content-Range header, and a Content-Length header showing the number of bytes actually transferred. For example,

HTTP/1.1 206 Partial content

Date: Wed, 15 Nov 1995 06:25:24 GMT

Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT

Content-Range: bytes 21010-47021/47022

Content-Length: 26012

Content-Type: image/gif

Content-Type

The *Content-Type* entity-header field indicates the media type of the entity-body sent to the recipient or, in the case of the HEAD method, the media type that would have been sent, had the request been a GET. The general syntax is:

Content-Type : media-type

Following is an example:

Content-Type: text/html; charset=ISO-8859-4

Expires

The *Expires* entity-header field gives the date/time after which the response is considered stale. The general syntax is:

Expires : HTTP-date

Following is an example:

Expires: Thu, 01 Dec 1994 16:00:00 GMT

Last-Modified

The *Last-Modified* entity-header field indicates the date and time at which the origin server believes the variant was last modified. The general syntax is:

Last-Modified: HTTP-date

Following is an example:

Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT

REQUEST HEADER

A request header is an HTTP header that can be used in an HTTP request to provide information about the request context, so that the server can tailor the response. For example, the Accept-* headers indicate the allowed and preferred formats of the response. Other headers can be used to supply authentication credentials (e.g. Authorization), to control caching, or to get information about the user agent or referrer, etc.

"Learning HTML and CSS is a lot more challenging than it used to be. Responsive web design adds more layers of complexity to design and develop websites."

— Jacob Lett

Not all headers that can appear in a request are referred to as *request headers* by the specification. For example, the Content-Type header is referred to as a representation header.

In addition, CORS defines a subset of request headers as simple headers, request headers that are always considered authorized and are not explicitly listed in responses to preflight requests.

The HTTP message below shows a few request headers after a GET request:

GET /home.html HTTP/1.1

Host: developer.mozilla.org

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

Referer: https://developer.mozilla.org/testpage.html

Connection: keep-alive

Upgrade-Insecure-Requests: 1

If-Modified-Since: Mon, 18 Jul 2016 02:36:04 GMT

If-None-Match: "c561c68d0ba92bbeb8b0fff2a9199f722e3a621a"

Cache-Control: max-age=0

RESPONSE HEADER

An HTTP response header is a component of a network packet that is sent by a Web server to a Web browser or client machine in response to an HTTP request. It is used in Web communications to deliver webpage and other Web-based data from the server to the requesting end-user browsers.

> *"Web development is difficult, only then it is fun to do. You just have to set your standards. If it were to be easy, would anyone do it?."*
>
> **—Olawale Daniel**

An HTTP response header primarily enables communicating and responding to user requests received on a Web server or the website. An HTTP response header works when a webpage or HTTP request is generated from the client's Web browser. This request is received in the form of an HTTP request header to the Web server containing the source address, requested data and its format and other data. The Web server responds back by creating an HTTP response header and attaching the requested data with it. The information embedded with the HTTP response header includes the destination IP address, data type, host address and more.

When configuring response headers, you must look at the definition for the HTTP header to determine how the client would handle the page content.

- For example, you configure an HTTP header for a specific page or all the pages with a Content-Security-Policy: frame-ancestors 'self' https://www.servicenow.com.
- When you invoke the page in a browser such as Chrome, you can review it in the Response Headers section of Chrome Developer Tools.



Warning: When using URLs with custom name-value pairs, proceed with caution because there is a potential security risk when doing so. The signed security amendment to the Now Platform contract has implied security. You may potentially or accidentally override it when you use custom name-value pairs in the resulting URLs.

- If you want to entirely disable HTTP response header configuration functions, set the glide.http.headers_config. enabled property to **false**.
- Once you set it to false, Now Platform does not use any of the header configurations you defined in the sys_response_header table.

Special handling of the Content-Security-Policy: frame-ancestor header

Normally, the Now Platform automatically includes the X-Frame-Options: SAMEORIGIN header.

- It supports use of this header in all types of browsers, based on the setting of the glide.set_x_frame_ options global property, which is enabled by default.
- When you configure a page with a Content-Security-Policy: frame-ancestor 'self' URL1 URL2 header, the Now Platform does not automatically include the X-Frame-Options: SAMEORIGIN header. Excluding it prevents the browser from being confused, because Content-Security-Policy: frame-ancestor 'self' already has a similar effect.

Special handling of Content-Security-Policy: frame-ancestor header for Internet Explorer

Using the Content-Security-Policy: frame-ancestor 'self' URL1 URL2 header enables you to configure multiple URL sources to include the page from within an iFrame rendered from a third-party site. However, Internet Explorer does not support this type of header.

- Instead, the Internet Explorer only supports the X-Frame-Options: ALLOW-FROM URL (ALLOW-FROM) directive in this header, although the restriction is for a single host URL.
- If you configure the frame-ancestor 'self' URL1 URL2 header, and Internet Explorer is in use, the Now Platform automatically uses the X-Frame-Options: ALLOW-FROM URL (ALLOW-FROM) header instead.

If the Internet Explorer request includes the referrer URL header:

- It attempts to match it with the host URLs (full or wildcard http://*.example.com type URL format only) configured in the Content-Security-Policy: frame-ancestor 'self' URL1 URL2 header.
- If there is a match, include the matched URL as X-Frame-Options: ALLOW-FROM URL1.
- If there is no referrer header, it uses the first non-wildcard based host URLs configured in the Content-Security-Policy: frame-ancestor 'self' URL1 URL2 header.

Note: When configuring URLs, do not include a forward slash at the end of the URL.

- This example of an incorrect configuration that may not work properly with this special handling:
  - Name: Content-Security-Policy
  - Value: frame-ancestors 'self' https://microsoft.com/
- Use this correct syntax instead:
- Name: Content-Security-Policy
- Value: frame-ancestors 'self' https://microsoft.com

CONCLUSION

Understanding URIs and URLs, MIME, and the details behind HTTP is important because they are the foundation for the mechanisms to identify, locate, code, and exchange information over the Internet and on intranets.

URIs and URLs make the naming, location, and identification of resources possible and global, which is fundamental to the World Wide Web and the applications that use the web infrastructure.

MIME is also a fundamental framework for the exchange of messages, given the wide variety of applications and either end of the communication exchange. The definition of standard rules for coding and the specification of a global character set allow consistency in the communication exchange.

HTTP and its vast communication structure not only allow but ease the adoption of web technology by providing rules for the actual communication process.

All these protocols make up a substantial portion of the fundamental structure for web communications, and understanding their specifications and the context in which they are likely to appear allows you to better understand the relation between the application world and the network infrastructure.

KEY TERMS

Web page	Multipurpose Internet Mail Extensions
URIs	Uniform Resource Identifier
Data Identifiers	Uniform Resource Locator

DISCUSSION QUESTIONS

1. Give an example of a uniform resource identifier?
2. What is HTTP Uniform Resource Identifier?
3. What is URI in API testing?
4. Give an example of URI and URL?
5. Why is data uniform resource identifier used?
6. How do you use Uniform Resource Name?
7. What is the other name of uniform resource?
8. What is a URN identifier?
9. What are the parts of a URN?
10. What are the four groups of HTTP header?
11. What is General header in HTTP?
12. What is HTTP header format?
13. What kind of data is passed via HTTP headers?
14. What's in an HTTP request header?

REFERENCES

1. Douglas E. Comer. Internetworking with TCP/IP. Principles, Protocols, and Architectures (4th Edition), volume 1. Prentice Hall, 2000.
2. James F. Kurose and Keith W. Ross. Computer Networking: A Top-Down Approach Featuring the Internet (2nd Edition). Addison Wesley, 2003.
3. Larry L. Petterson and Bruce S. Davie. Computer Networks: A System Approach (2nd Edition). Morgan Kaufmann, 2000.
4. Sun Microsystems, Inc., http://www.rfc-editor.org/rfc/rfc3010.txt. RFC 3010. Network File System (NFS) version 4 Protocol, 2003.

INDEX

A

Access control lists (ACLs) 99
acknowledged (ACK) 278
acknowledgement number field 284
ActiveX Data Objects (ADO) 118, 123
Address Resolution Protocol (ARP) 17
American National Standards Institute (ANSI) 122
Anti-passback 195
anti-virus programs 188
Application architectures 7
Application program interface (API) 112
Application Specific Integrated Circuit (ASIC) 246
Atomic, Consistent, Isolation, Durable (ACID) 117
attribute-based access control (ABAC) 206
authentication 191, 200, 207, 213
Availability 191

B

Basic data transfer 277
biometrics 197, 199, 206, 207, 212

C

card key 201, 207, 209, 212
checksum field 276, 286, 289, 290
Client Request-header 321
Client-server architecture 49
Client-server model 8
cloud computing 189
coarse wavelength division multiplexing (CWDM) 22
Common Internet File System (CIFS) 95
Common Object Request Broker Architecture (CORBA) 99
Context Based Access Control (CBAC) 120
Customer Information Control System (CICS) 99
Customer relationship management (CRM) 90

D

Data center 2, 3, 4, 5, 6, 7, 10, 12, 13, 14, 23, 24, 41, 44, 48, 51, 52, 53, 87, 238
Data Center Facility 4
Data center security 188, 189
Data center virtualization 6, 7
data link 191
Data server 52
Demilitarized zone (DMZ) 145
denial-of-service (DoS) 191
Dense wave division multiplexing (DWDM) 130
dense wavelength division multiplexing (DWDM) 22
destination port field 289
Direct Server Return (DSR) 242
Distributed Component Object Model (DCOM) 99

DNS-based load balancing 225, 226, 227, 259
domain name 301, 302, 304, 305
Domain Name System (DNS) 93
Dynamic link libraries (DLLs) 111

E

Enterprise Application Integration (EAI) 98
Enterprise Java Beans (EJB) 99
Enterprise resource planning (ERP) 90
Entity-header 321
Ethernet controller 62, 63
Ethernet drivers 61, 62
Explicit Congestion Notification (ECN) 269

F

Fail-Over Cable 237
Fast EtherChannel distributes 249
Fiber Channel, ESCON, and fiber connectivity (FICON) 22
fingerprint scanner 206
Firewall load balancing 229, 230
firewalls 188, 191
flags field 271, 276
Flow control 277, 278
fragment offset field 272

G

General-header 320
Gigabit Ethernet (GE) 22
Global Server Load Balancing (GSLB) 223
Greenwich Mean Time (GMT) 323

H

header checksum field 274
Header Length Field 268, 285
heating, ventilation and air conditioning (HVAC) 203, 214
Hot Standby Routing Protocol (HSRP) 237
HTTP header fields 320
HTTP response header 343, 344
Hypertext Transfer Protocol (HTTP) 262

I

identifier field 271
identity, credential and access controls management (ICAM) 205
initial sequence number (ISN) 283
Internal Gateway Protocols (IGPs) 254
International Organization for Standardization [ISO] 95
Internet Assigned Numbers Authority (IANA) 268
Internet Control Message Protocol (ICMP) 133
Internet Data Centers (IDCs) 149
Internet Information Server (IIS) 94
Internet Protocol Television (IPTV) 140
Intrusion Detection Systems (IDSs) 169
IP header 267, 268, 269, 274, 275, 276, 279, 285, 289

J

Java Database Connectivity (JDBC) 112
Java Server Pages (JSP) 102
Java virtual machine (JVM) 112

L

Leased-line scenario 238, 239
length field 275, 276, 285, 287, 289
Local Disk Manager (LDM) 129
Local network 52
Logical Link Control [LLC] 271
loose source route (LSR) 275

M

Manageability 193
Maximum Transmittable Unit (MTU) 248
Multilayer Switch Feature Card (MSFC) 171
Multiplexing 277, 278
Multipurpose Internet Mail Extensions (MIME) 110, 297

N

namespace-specific string (NSS) 308
Near Filed communications (NFC) 208
Network address translation (NAT) 100

Network Attached Storage (NAS) 21
Network interface cards (NICs) 53
network operations center (NOC) 193
network protocol 266
N-tier architecture 8, 10

O

Object Management Group (OMG) 118
Open Database Connectivity (ODBC) 117
Open Systems Interconnection (OSI) 122
options field 275, 287

P

Packet processing 59
Peer-to-Peer Remote Copy (PPRC) 129
Peripheral component interconnect-extended (PCI-X) 58
Personal Identification Number 201
physical security 188, 194, 200, 201, 202, 203, 204, 205, 206, 209, 210
protocol field 273, 289
protocol stack 263, 264, 265, 266, 277, 283
Public-key infrastructure (PKI) 96

Q

quality of service (QoS) 270

R

radio frequency identification (RFID) 208
Read-only access 233
record route (RR) 275
Redundancy 213
Redundant Array of Independent Disks (RAID) 127
Reliability 277, 278
Remote Procedure Calls (RPC) 100
request header 295, 337, 342, 343, 347
reverse proxy 192

S

Scalability 192
Secure Socket Layer (SSL) 262
Security 187, 190, 201, 202, 204, 205, 209
selective acknowledgement (SACK) 287
sequence number field 283, 287
Sequence number (SEQ) 277
Server-based load balancer 245, 246
server farm 262, 265, 279, 293
Server Load Balancing (SLB) 223
Server Response-header 321
Service Checking 238
service-level agreements (SLAs) 193
Simple Object Access Protocol (SOAP) 100
Small Computer System Interface (SCSI) 127
smart cards 207
source port field 288
Source routing 275
Spanning Tree Protocol (STP) 152, 254
Stateful Fail-Over 237
strict source route (SSR) 275
Subnetwork Access Protocol [SNAP] 271
Superuser 233
Switch-based load balancer 246
Switch fabric 251, 253, 254
Systems Network Architecture (SNA) 97

T

TCP header length field 285
timestamp 276, 287
Time-To-Live field (TTL) 272
Total concurrent connections 247
Total Length Field 271
Traffic distribution 225, 227
Transaction Processing (TP) 117
Transport Control Protocol (TCP) 262

U

UDP header compression 290
Uniform Resource Characteristics (URC) 297
Uniform Resource Locator (URL) 297, 298, 308
Uniform Resource Name (URN) 297, 298
urgent pointer field 286, 287
URI syntax 307
User-access level 233
User Datagram Protocol (UDP) 120, 262

3G E-LEARNING

V

Version Field 267
Virtual IP (VIP) 232
virtualization 189
Virtual private network (VPN) 96, 147
Virtual Router Redundancy Protocol (VRRP) 236
Voice over Internet Protocol (VoIP) 140
vulnerability 189, 190

W

Web browser 311, 312, 313, 314, 343
Web Cache Communication Protocol (WCCP) 170
Webpage request 222
Web server 225, 238
Web stores 250
window size field 286
Wireless Session Protocol (WSP) 125
Worldwide Web Consortium (W3C) 300

Level: Beginner to Advanced
Subject: Computer and Information Science

Data Centre Management

3rd Edition

Times have changed and the demand for information is higher than ever. Just a few decades ago, society was content with simple TV shows with few channels to choose from. Even communication is easy, and snail mail is widely accepted. However, this is no longer the case. Ever since the internet became an everyday necessity and almost everyone owns a smartphone, we spend most of our waking moments online. Whether for work or socialization, the internet plays a huge role in our lives. Of course, the demand for real-time data transmission is at an all-time high. This need for computers and other network devices to handle these demands is the catalyst for immersion in the modern data center. The management of large amounts of data and hardware systems that are part of a widespread network is data center management. Data centers are the powerhouses of the industrial world. As technological advances continue to transform the way companies do business, the need for data and performance increases. Oftentimes, this significant increase in resources is too much for many organizations to handle internally or with existing corporate infrastructure. Many hours and staff are required to keep up the high demand. Using a data center allows a company to outsource its energy needs. Businesses benefit from data centers in many ways. In particular, they allow a company to focus on its consumer goals rather than technological maintenance.

This edition is thoroughly revised and new topics are added. A separate chapter is also added. The book in this edition shows how data management systems and processes help to provide accurate and reliable data to support strategic decisions and day-to-day operations. It examines all steps to be taken with a view to formulating a strategic data center strategy; leverages awareness of data center developments and technology to be applied to businesses; involves all stakeholders and implements effective cost modeling; analyses efficiency, costs and capabilities to build an integrated data center strategy. This edition is organized into eight chapters.

ISBN 978-1-98468-211-6

BIBLIOTEX
Digital Library

www.3ge-learning.com
email: info@3ge-learning.com