Business Data Communications & Networking:

12th Edition

Chapter 2

Sadiq M. Sait, PhD
Professor
College of Computer Sciences and Engineering
Director, Industry Collaboration

September 2023

APPLICATION LAYER

- The application layer (Layer 5) is the ***software*** that enables the user to perform useful work.
- The software at the application layer is the ***reason*** for having the network because it is this software that provides the ***business value***.
- This chapter examines the ***five fundamental types of application architectures*** used at the application layer (***host-based, client-based, client-server, cloud-based, and peer-to-peer)***.
- We then look at the Internet and the primary software application packages it enables: the Web, email, Telnet, and instant messaging.

LESSON OBJECTIVES

- Understand
  - host-based
  - client-based
  - client-server, and
  - cloud-based application architectures
- Understand
  - How the Web works?
  - How email works? and
  - Be aware of how Telnet and instant messaging work

2.1 INTRODUCTION

- Network applications are the ***software packages*** that run in the application layer.
- You should be quite familiar with many types of network software, because it is these application packages that you use when you use the network.
- In many respects, the only reason for having a network is to enable these applications.
- We first discuss five ***basic architectures*** for network applications and how each of those architectures affects the ***design*** of networks.
- We will use applications such as the Web and word processing, as examples of different application architectures.
- We then examine several common applications used on the Internet(e.g., Web, email) and use those to explain how application software interacts with the networks.
- By the end of this chapter, you should have a much better understanding of the application layer in the network model and what exactly we meant when we used the term protocol data unit (PDU).

2.2 APPLICATION ARCHITECTURES

- We now get a bit more specific about how the client computer and the server computer can work together to provide application software to the users.
- *An application architecture is how the functions of the application layer software are spread among the clients and servers in the network.*
- The work done by any application program can be divided into **four** general functions.
  - The first is **data storage**. Most application programs require data to be stored and retrieved, whether it is a small file such as a memo produced by a word processor or a large database.
  - The second function is **data access logic**, the processing required to access data, which often means database queries in SQL (structured query language).
  - The third function is **application logic** (sometimes called business logic), which also can be simple or complex, depending on the application.
  - The fourth function is the **presentation logic**, the presentation of information to the user and the acceptance of the user's commands.

2.2 APPLICATION ARCHITECTURES (contd).

- These four functions—data storage, data access logic, application logic, and presentation logic—are the basic building blocks of any application.
- There are many ways in which these four functions can be allocated between the client computers and the servers in a network.
- There are **five** fundamental application architectures in use today.
  - In host-based architectures, the server (or host computer) performs virtually all of the work.
  - In client-based architectures, the client computers perform most of the work.
  - In client-server architectures, the work is shared between the servers and clients.
  - In cloud-based architectures, the cloud provides services (software, platform, and/or infrastructure) to the client.
  - In peer-to-peer architectures, computers are both clients and servers and thus share the work.

Although the **client-server architecture** is the dominant application architecture, **cloud-based architecture is becoming the runner-up** because it offers **rapid scalability** and **deployability** of computer resources.

TF2-1 Cloud Computing Deployment Models

- When an organization decides to use cloud-based architecture, it needs to decide on **which deployment model will it use**. There are three deployment models from which to choose:

  - **Private Cloud:**
    - These are created for the exclusive use of a **single private organization**.
    - The cloud (hardware and software) would be **hosted** by the organization in a **private data center**.
    - This deployment model provides the **highest levels of control, privacy, and security**.
    - This model is often used by organizations needing to satisfy regulations posed by regulators, such as in the financial and health care industries.

TF2-1 Cloud Computing Deployment Models (contd.)

- **Public Cloud**:
  - This deployment model is **used by multiple organizations** that **share the same cloud resources**.
  - The level of control is lower than in private clouds, and many companies are concerned with the **security of their data**.
  - However, this deployment model doesn't require any **upfront capital investment**, and the ***selected service can be up and running in a few days***.
  - Public clouds are a good choice when a lot of people in the organization are using the same application.
    - Because of this, the most frequently used software as a service (SaaS) is email. For example, many universities have moved to this model for their students.

TF2-1 Cloud Computing Deployment Models (contd.)

- **Community Cloud**:
  - This deployment model is used by organizations that have a **common purpose**.
  - Rather than each organization creating its own private cloud, organizations decide to **collaborate** and pool their resources.
  - Although this cloud is not private, only a limited number of companies have access to it.
  - Community clouds are considered to be a **subset** of public clouds.
  - ***Therefore, community clouds realize the benefits from cloud infrastructure (such as speed of deployment) with the added level of privacy and security that private clouds offer.***
  - This deployment model is often used in the government, health care, and finance industries, members of which have similar application needs and require a very high level of security.
- Sometimes an organization will choose to use only one of these deployment models for all its cloud-based applications.
- This strategy is called a ***pure strategy***, such as a pure private cloud strategy or a pure public cloud strategy.

TF2-1 Cloud Computing Deployment Models (contd.)

- **Community Cloud (contd).**:
  - In other cases, the organization is best supported by a mix of public, private, and community clouds for different applications.
  - This strategy is called a **hybrid** cloud strategy.
  - A hybrid cloud strategy allows the organization to take advantage of the benefits that these different cloud deployment models offer.
    - For example, a hospital can use Gmail for its email application (public cloud) but a private cloud for patient data, which requires high security.
  - The downside of a hybrid cloud strategy is that an organization has to deal with different platforms and cloud providers. However, the truth is that this strategy offers the ***greatest flexibility***, so most organizations eventually end up with this strategy

2.2.1 Host-Based Architectures

- First DCNs developed in the 1960s were host-based, with the server (usually a large mainframe computer) performing all four functions.
- The (*dumb*) clients (usually terminals) enabled users to send and receive messages to and from the host computer.
- The *clients merely captured keystrokes*, sent them to the server for processing, and accepted instructions from the server on what to display.
  - This very simple architecture often works very well.
  - Application software is developed and stored on one server along with all data.
- If you've ever used a terminal, you've used a host-based application.
- There is one point of control, because all messages flow through one central server.
  - Economies of scale, because all computer resources are centralized.
- There are two fundamental problems with host-based networks.
  - As the demands for more and more network applications grow, many servers become overloaded and unable to quickly process all the users' demands.
  - Prioritizing users' access becomes difficult.
  - Response time becomes slower, and you have to spend more money to upgrade the server.
  - Unfortunately, upgrades: (i) Expensive; (ii) Come in large increments.

2.2.2 Client-Based Architectures

- In the late 1980s, there was an explosion in the use of personal computers.
- Today, more than 90% of most organizations' total computer processing power now resides on personal computers, not in centralized mainframe computers.
- Downsizing to reduce cost.
- Part of this expansion was fueled by many low-cost, highly popular applications such as word processors, spreadsheets, and presentation graphics programs.
- It was partly fueled by managers' frustrations with application software on host mainframe computers.
  - Most mainframe software is not as easy to use as personal computer software, is far more expensive, and can take years to develop.
- In the late 1980s, many large organizations had application development backlogs of 2 to 3 years; that is, getting any new mainframe application program written would take years. New York City, for example, had a 6-year backlog. In contrast, managers could buy personal computer packages or develop personal computer-based applications in a few months.

2.2.2 Client-Based Architectures (contd).

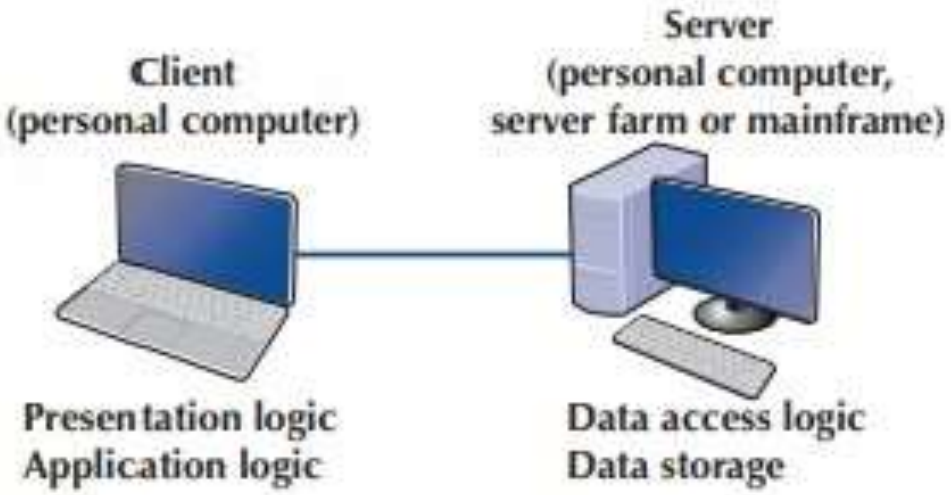
- With client-based architectures, the clients are PCs on a LAN, and the server is usually another personal computer on the same network.
- The ***application software on the client computers is responsible*** for the presentation logic, the application logic, and the data access logic; *the server simply stores the data.*
- This simple architecture often works very well. If you've ever used a word processor and stored your document file on a server (or written a program in Visual Basic or C that runs on your computer but stores data on a server), you've used a client-based architecture.
- The fundamental problem in client-based networks is that all data on the server must travel to the client for processing.
  - For example, suppose the user wishes to display a list of all employees with company life insurance. All the data in the database (or all the indices) must travel from the server where the database is stored over the network circuit to the client, which then examines each record to see if it matches the data requested by the user.
  - *This can overload the network circuits because far more data are transmitted from the server to the client than the client needs.*

2.2.3a Client-Server Architectures

- Most applications written today use client-server architectures.
- Client-server architectures attempt to ***balance the processing*** between the client and the server by having both do some of the logic.
- ***In these networks, the client is responsible for the presentation logic, whereas the server is responsible for the data access logic and data storage.***
- ***The application logic may either reside on the client, reside on the server, or be split between both.***
- Figure 2-3 shows the simplest case, with the presentation logic and application logic on the client and the data access logic and data storage on the server.
- In this case, the client software accepts user requests and performs the application logic that produces database requests that are transmitted to the server.
- The server software accepts the database requests, performs the data access logic, and transmits the results to the client.
- The client software accepts the results and presents them to the user.
  - When you used a Web browser to get pages from a Web server, you used a client-server architecture.
  - Likewise, if you've ever written a program that uses SQL to talk to a database on a server, you've used a client-server architecture.
  - For example, if the user requests a list of all employees with company life insurance, the client will accept the request, format it so that it could be understood by the server, and transmit it to the server. On receiving the request, the server searches the database for all requested records and then transmits only the matching records to the client, which would then present them to the user.

FIGURE 2-3

Two-tier client-server architecture



2.2.3b Client-Server Architectures (contd).

- The same would be true for database updates; the client accepts the request and sends it to the server. The server processes the update and responds (either accepting the update or explaining why not) to the client, which displays it to the user.
- One of the ***strengths*** of client-server networks is that they enable software and hardware from different vendors to be used together.
- But this is also one of their **disadvantages**, because it can be difficult to get software from different vendors to work together.
- One solution to this problem is middleware, software that sits between the application software on the client and the application software on the server.

2.2.3c Client-Server Architectures (contd).

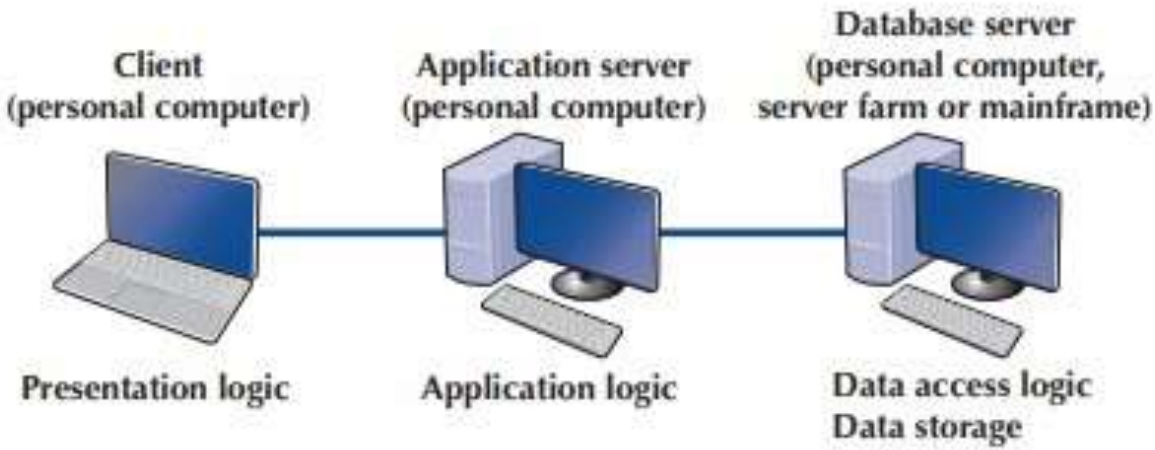
- **Middleware** does two things:
  - First, it provides a standard way of communicating that can **translate between software** from different vendors. *Many middleware tools began as translation utilities that enabled messages sent from a specific client tool to be translated into a form understood by a specific server tool.*
  - The second function of middleware is to manage the message transfer from clients to servers (and vice versa) **so that clients need not know the specific server that contains the application's data**.
  - The application software on the client sends all messages to the middleware, which forwards them to the correct server.
  - The application software on the client is therefore protected from any changes in the physical network.
    - If the network layout changes (e.g., a new server is added), only the middleware must be updated.
  - There are literally dozens of standards for middleware, each of which is supported by different vendors and each of which provides different functions.
  - Two of the most important standards are ***Distributed Computing Environment (DCE) and Common Object Request Broker Architecture (CORBA).*** Both of these standards cover virtually all aspects of the client-server architecture but are quite different.
  - Another important standard is Open Database Connectivity (ODBC), which provides a standard for data access logic.

2.2.3c Client-Server Architectures (contd).

- **Two-Tier and Three-Tier**
  - There are many ways in which the application logic can be partitioned between the client and the server. The example in Figure 2-3 is one of the most common.
  - In this case, the server is responsible for the data, and the client for the application and presentation.
  - This is called a two-tier architecture because it uses only two sets of computers, one set of clients and one set of servers.
  - A three-tier architecture uses three sets of computers, as shown in Figure 2-4.
    - In this case, the software on the client computer is responsible for presentation logic, an application server is responsible for the application logic, and a separate database server is responsible for the data access logic and data storage.

FIGURE 2-4

Three-tier client-server architecture

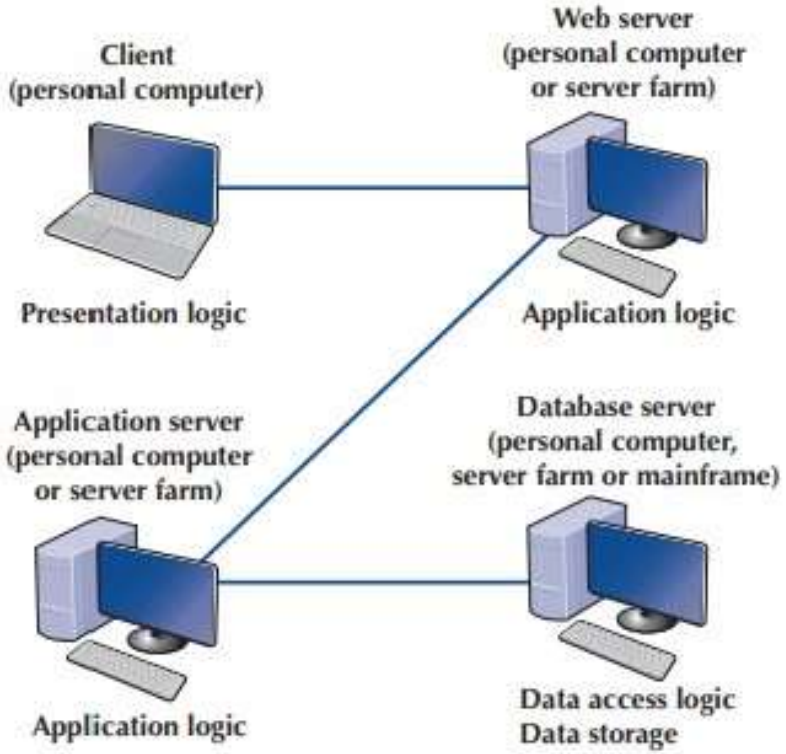


2.2.3c Client-Server Architectures (contd).

- **N-Tier Architectures**
  - N-tier architecture uses more than three sets of computers.
  - In this case, the client is responsible for presentation logic, a database server is responsible for the data access logic and data storage, **and the application logic is spread across two or more different sets of servers.**
  - Figure 2-5 shows an example of an n-tier architecture of a groupware product called TCB Works developed at the University of Georgia. TCB Works has four major components.
    - The first is the Web browser on the client computer that a user uses to access the system and enter commands (presentation logic).
    - The second component is a Web server that responds to the user's requests, either by providing HTML pages and graphics (application logic) or by sending the request to the third component.
    - The third component is a set of 28 C programs that perform various functions such as adding comments or voting (application logic).
    - The fourth component is a database server that stores all the data (data access logic and data storage).

    Each of these four components is separate, making it easy to spread the different components on different servers and to partition the application logic on two different servers.,

FIGURE 2-5
The *n*-tier client-server architecture



2.2.3c Client-Server Architectures (contd).

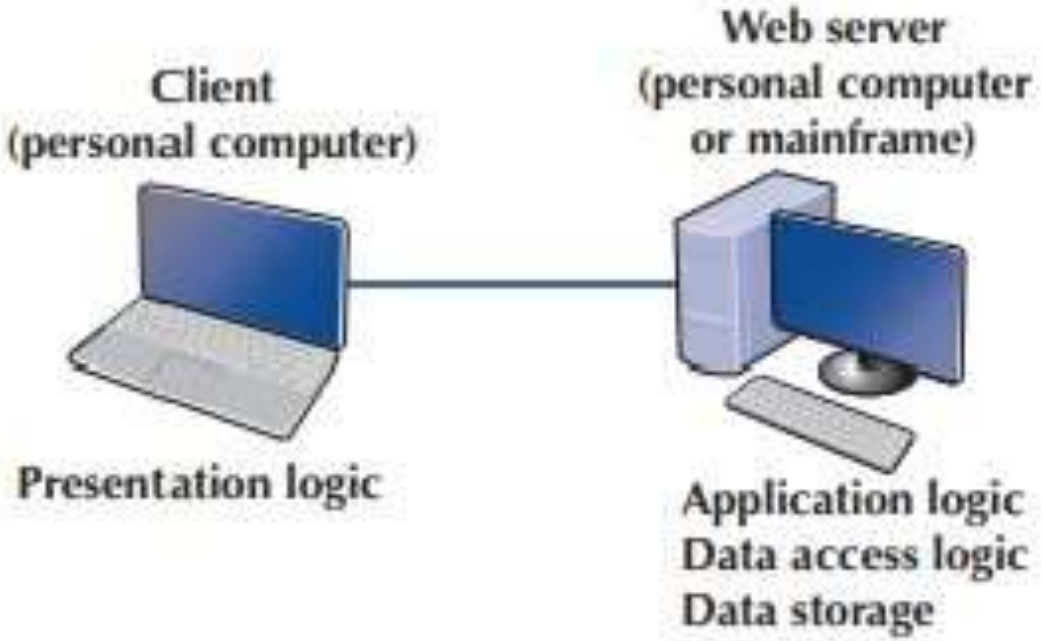
- **N-Tier Architectures: Advantages**
  - The primary advantage of an n-tier client-server architecture compared with a two-tier architecture (or a three-tier compared with a two-tier) is that it **separates the processing** that occurs to better balance the load on the different servers; it is more scalable.
    - In Figure 2-5, we have three separate servers, which provides more power than if we had used a two-tier architecture with only one server.
    - If we discover that the application server is too heavily loaded, we can simply replace it with a more powerful server, or even put in two application servers.
    - Conversely, if we discover the database server is underused, we could put data from another application on it.

2.2.3c Client-Server Architectures (contd).

- **Thin Clients versus Thick Clients:**
  - Another way of classifying client-server architectures is by examining. ***how much of the application logic is placed on the client computer***
  - A thin-client approach places ***little or no application logic*** on the client (e.g., Figure 2-5), whereas a thick-client (also called fat-client) approach places ***all or almost all of the application logic on the client*** (e.g., Figure 2-3).
  - There is no direct relationship between thin and fat clients, and two-, three- and n-tier architectures.
  - For example, Figure 2-6 shows a typical Web architecture: a two-tier architecture with a thin client.
    - **One of the biggest forces favoring thin clients is the Web.**
    - Thin clients are much easier to manage. If an application changes, only the server with the application logic needs to be updated.
    - **With a thick client, the software on all of the clients would need to be updated.** Conceptually, this is a simple task; one simply copies the new files to the hundreds of affected client computers. In practice, it can be difficult.
  - Thin-client architectures are the future. More and more application systems are being written to use a Web browser as the client software, with Java Javascriptor AJAX (containing some of the application logic) downloaded as needed.
  - **This application architecture is sometimes called the distributed computing model.**
  - **The thin-client architecture also enables cloud-based architecture.**

FIGURE 2-6

The typical two-tier thin-client architecture of the Web



2.2.4 Cloud Computing Architectures

- The traditional client-server architecture can be **complicated** and **expensive** to deploy.
- Every application has to be ***hosted on a server*** so that it can fulfill requests from potentially thousands of clients.
- An organization has hundreds of applications, so running a successful client-server architecture requires a variety of software and hardware and ***skilled*** personnel who can build and maintain this architecture.
- Cloud computing architectures are different because they ***outsource*** part or all of the infrastructure to other firms that specialize in ***managing that infrastructure***.
- There are **three** common cloud-based architecture models.
- Figure 2-7 summarizes these three models and compares them to the client-server architecture.

FIGURE 2-7

Cloud architecture models compared to thin-client client-server architecture

Source: Adapted from www.cbc.radio-canada.ca/en/reporting-to-canadians/sync/sync-issue-1-2012/cloud-services

	Thin-Client Client-Server		Infrastructure as a Service		Platform as a Service		Software as a Service	
Who manages which parts	Internal	Outsourced	Internal	Outsourced	Internal	Outsourced	Internal	Outsourced
Application Logic	X		X		X			X
Data Storage	X		X		X			X
Data Access Logic	X		X			X		X
Operating System	X		X			X		X
Virtualization Software	X		X			X		X
Server Hardware	X			X		X		X
Storage Hardware	X			X		X		X
Network Hardware	X			X		X		X

2.2.4 Cloud Computing Architectures (contd).

- The first column of this figure shows the thin-client client-server architecture, in which the organization manages the entire application software and hardware.
  - In addition to the software components we've discussed previously (the application logic, data access logic, and the data themselves), the servers need an operating system (e.g., Windows, Linux).
  - Most companies also use ***virtualization*** software to install many virtual or logical servers on the same physical computer.
  - ***This software (VMware is one of the leaders) creates a separate partition on the physical server for each of the logical servers.***
  - Each partition has its own operations system and its own server software and works **independently** from the other partitions.

2.2.4 Cloud Computing Architectures (contd).

- This software must run on some hardware, which includes a server, a storage device, and the network itself.
- The server may be a large computer or a server farm.
- A server farm is a **cluster** of computers linked together so that they act as **one computer**.
- Requests arrive at the server farm (e.g., Web requests) and are distributed among the computers so that no one computer is overloaded (load **balancing**).
- Each computer is separate so that if one fails, the server farm simply bypasses it (fault **tolerance**).
- Server farms are more complex than single servers because work must be quickly coordinated and shared among the individual computers.
- **Server farms are very scalable** because one can always add another computer.
- Figure 2-8 shows one row of a server farm at Indiana University.
  - There are seven more rows like this one in this room, and another room contains about the same number.
  - Many companies use **separate storage devices** instead of the hard disks in the servers themselves.
  - **These storage devices are special-purpose hard disks designed to be very large and very fast.**
  - The six devices on the left of Figure 2-8 comprise a special storage device called a storage area network (SAN)

FIGURE 2-8

One row of a server farm at Indiana University

Source: Courtesy of the author, Alan Dennis

2.2.4 Cloud Computing Architectures (SaaS)

- Software as a Service (SaaS) is one of the three cloud computing models.
- With SaaS, an organization **outsources** the entire application to the cloud provider (see the last column of Figure 2-7) and uses it as any other application that is available via a browser (thin client).
- SaaS is based on **multi-tenancy**. This means that rather than having many copies of the same application, there is only one application that everybody shares, yet everybody can customize it for his or her specific needs.
  - As an example, imagine a giant office building in which all people share the infrastructure (water, A/C, electricity) but can customize the offices they are renting.
- The customers can customize the app and **don't have to worry about upgrades, security, or underlying infrastructure because the cloud provider does it all**.
- The most frequently used SaaS application is **email**.
- At Indiana University, all student email is outsourced to Google's Gmail.
- Customer relationship management (CRM) from Salesforce.com is another very commonly used SaaS.

2.2.4 Cloud Computing Architectures (PaaS)

- PaaS is another of the three cloud computing models and is called Platform as a Service (PaaS).
- What if there is an application you need, but no cloud provider offers one you like?
  - ***Then you can build your own application and manage your own data on the cloud infrastructure provided by your cloud supplier.***
- *The developers in your organization decide what programming language to use to develop the application of choice*.
- **The needed hardware and software infrastructure, called the platform, is rented from the cloud provider (see Figure 2-7).**
- In this case, **the organization manages the application and its own data** *but uses the database software (data access logic) and operating system provided by the cloud provider.*
- PaaS offers a much faster development and deployment of custom applications at a fraction of the cost required for the traditional client-server architecture.
- **PaaS providers include Amazon Elastic Cloud Compute (EC2), Microsoft Windows Azure, and Google App Engine.**

2.2.4 Cloud Computing Architectures (IaaS)

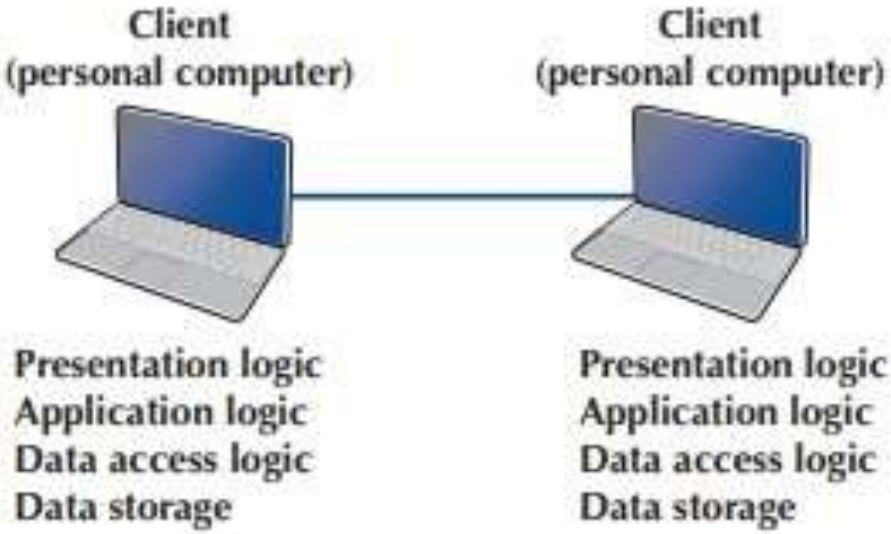
- As you can see in Figure 2-7, with **IaaS, the cloud provider manages the hardware, including servers, storage, and networking components.**
- The organization is responsible for all the software, including operating system (and virtualization software), database software, and its applications and data. IaaS is sometimes referred to also as HaaS, or Hardware as a Service because in this cloud model, only the hardware is provided; everything else is up to the organization.
- This model allows a decrease in capital expenditures for hardware and maintaining the proper environment (e.g., cooling) and redundancy and backups for data and applications.
- Providers of IaaS are Amazon Web Services, Microsoft Windows Azure, and Akamai.
- In conclusion, cloud computing is a technology that fundamentally changed the way we think about applications in that they are rented and paid for as a service.
- The idea is the same as for utilities—water, gas, cable, and phone.
  - The utility provider builds and is running the infrastructure; you plug in and sign up for a type of service. Sometimes you pay as you go (water, gas), or you sign up for a level of service (phone, cable).

2.2.5 Peer-to-Peer Architectures (P2P)

- Peer-to-peer (P2P) architectures are very old, but their modern design became popular in the early 2000s with the rise of **P2P file sharing applications** (e.g., Napster).
- With P2P architecture, **all computers act as both a client and a server.**
- **Therefore, all computers perform all four functions**: (i) presentation logic, (ii) application logic, (iii) data access logic, and (iii) data storage (see Figure 2-9).
- With a P2P file sharing application, a user uses the presentation, application, and data access logic installed on his or her computer to access the data stored on another computer in the network.
- With a P2P application sharing network (e.g., grid computing such as seti.org), other users in the network can use others' computers to access application logic as well.
- The advantage of P2P networks is that the data can be installed anywhere on the network. They spread the storage throughout the network, even globally, so they can be very ***resilient*** to the failure of any one computer.
- **The challenge is finding the data!!!.**
  - There must be some central server that enables you to find the data you need, so P2P architectures often are combined with a client-server architecture.
- **Security is a major concern in most P2P networks**, so P2P architectures are not commonly used except for specialized computing needs (e.g., grid computing).

FIGURE 2-9

Peer-to-peer architecture



2.2.6 Choosing Architectures

- Each of the preceding architectures has certain ***costs and benefits***, so how do you choose the “right” architecture?
- In many cases, the architecture is simply given; the organization has a certain architecture, and one simply has to use it.
- In other cases, the organization is acquiring new equipment and writing new software and has the opportunity to develop a new architecture, at least in some parts of the organization.
- ***Almost all new applications today are client-server applications***. Client-server architectures provide
  - **the best scalability**, and
  - **the ability to increase (or decrease) the capacity of the servers to meet changing needs**.
    - For example, we can easily add or remove application servers or database servers depending on whether we need more or less capacity for application software or database software and storage.

2.2.6 Choosing Architectures (contd.)

- Client-server architectures are also the **most reliable**.
  - We can use multiple servers to perform the same tasks so that if one server fails, the remaining servers continue to operate and users don't notice problems (FT).
- Finally, client-server architectures are usually the **cheapest** because many tools exist to develop them.
- And lots of client-server software exists for specific parts of applications so we can more quickly buy parts of the application we need.
  - For example, **no one writes Shopping Carts** for eCommerce sites anymore; it's **cheaper to buy** a Shopping Cart software application and put it on an application server than to write your own ***(can you think of another application?).***
- Client-server architectures **also enable cloud computing**.
  - As mentioned in Section 2.2.4, companies may choose to run the software as a service (SaaS) because of its **low price** and **high scalability** compared to traditional client-server architecture hosted at home.
- One major issue companies face when choosing SaaS is the **security** of the data.
- Each company has to evaluate the risk of its data being **compromised** and select its cloud provider carefully.
- However, SaaS is gaining **popularity**, and companies are becoming increasingly accustomed to this solution.

MF 2-1 Cloud Computing with salesforce.com

- Salesforce.com, the world's number one cloud platform, is the poster child for cloud computing.
- Companies used to buy and install software for customer relationship management (CRM), the process of identifying potential customers, marketing to them, converting them into customers, and managing the relationship to retain them.
- The software and needed servers were expensive and took a long time to acquire and install.
- Typically, only large firms could afford it.
- Salesforce.com changed this by offering a cloud computing solution.
- The CRM software offered by salesforce.com resides on the salesforce.com servers.
- There is no need to buy and install new hardware or software.
- Companies just pay a monthly fee to access the software over the Internet.
- Companies can be up and running in weeks, not months, and it is easy to scale from a small implementation to a very large one.
- Because salesforce.com can spread its costs over so many users, they can offer deals to small companies that normally wouldn't be able to afford to buy and install their own software.
- Salesforce is a very competitive organization that is keeping up with the mobile world too.
  - In Fall 2013, it announced the "Salesforce $1 Million Hackathon," where hundreds of teams competed to build the next killer mobile app on the Salesforce platform. Yup, the winning team will walk away with $1 million!
  - What is a killer app?

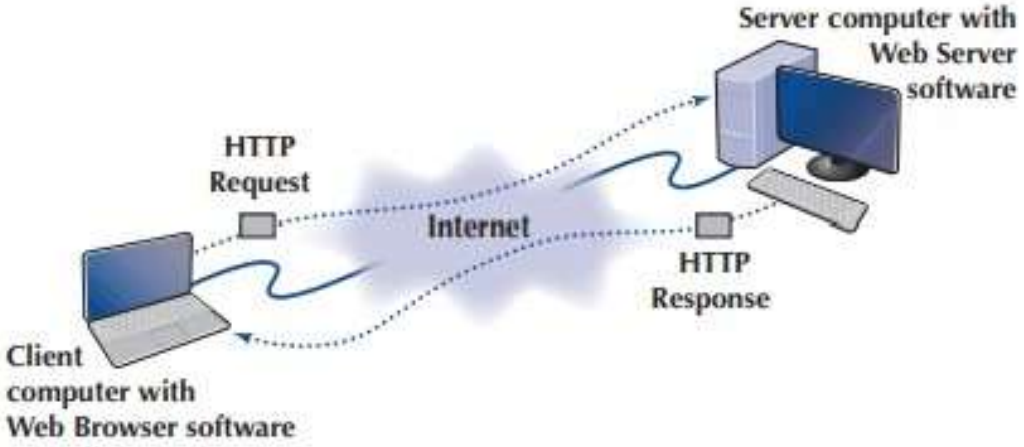
2.3 World Wide Web

- The Web was first conceived in 1989 by Sir Tim Berners-Lee at the European Particle Physics Laboratory (CERN) in Geneva.
- His original idea was to develop a database of information on physics research, but he found it difficult to fit the information into a traditional database.
- Instead, he decided to use a hypertext network of information.
- With hypertext, any document can contain a link to any other document ***(every file or document has a unique address in the world).***
- CERN's first Web browser was created in 1990, but it was 1991 before it was available on the Internet for other organizations to use.
- **By the end of 1992**, several browsers had been created for UNIX computers by CERN and several other European and American universities, and there were about **30 Web servers in the entire world**.
- In 1993, Marc Andreessen, a student at the University of Illinois, led a team of students that wrote **Mosaic**, the first graphical Web browser, as part of a project for the university's National Center for Supercomputing Applications (NCSA).
- **By the end of 1993**, the Mosaic browser was available for UNIX, Windows, and Macintosh computers, and there were about **200 Web servers in the world**.
- Today, no one knows for sure how many Web servers there are. There are more than 1.7 Billion separate Web sites, of which 400 Million are active.

2.3.1 How the Web Works?

- **The Web is a good example of a two-tier client-server architecture** (Figure 2-10).
- Each client computer needs an application layer software package called a web browser.
- There are many different browsers, such as Microsoft's Edge, Google's Chrome, etc.
- Each server (computer) on the network that will act as a Web server needs an application layer software package called a Web server.
- There are many different Web servers, such as those produced by Microsoft and Apache.
  - Other examples: (i) Internet Information Services, (ii) LiteSpeed Web Server, (iii) Jigsaw Server, (iv) Google Web Server, etc.
- To get a page from the Web, the user must type the Internet **uniform resource locator** (URL) for the page he or she wants (e.g., www.yahoo.com) or click on a link that provides the URL.
- **The URL specifies the Internet address of the Web server** and the directory and name of the specific page wanted.
- If no directory and page are specified, the Web server will provide whatever page has been defined as the site's **home page**.
- **For the requests from the Web browser to be understood by the Web server, they must use the same standard protocol or language.**
- If there were no standards and each Web browser used a different protocol to request pages, then it would be impossible for a Microsoft Web browser to communicate with an Apache Web server, for example.

FIGURE 2-10
How the Web works



2.3.1 How the Web Works (contd).

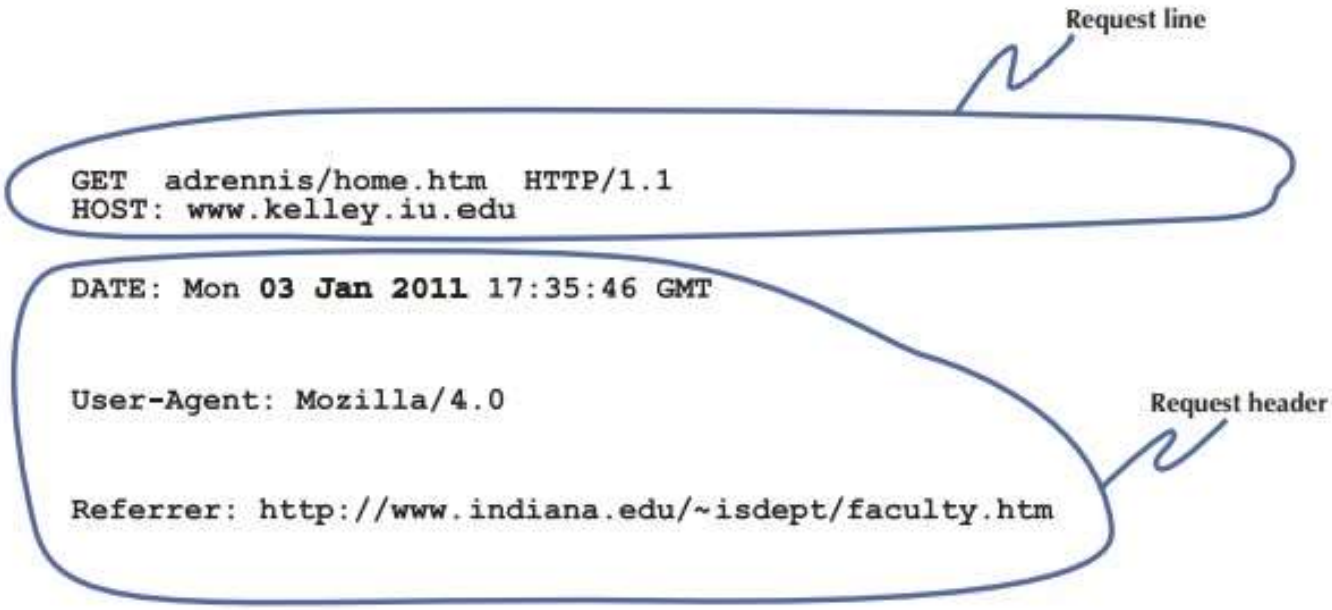
- **The standard protocol** for communication between a Web browser and a Web server is **Hypertext Transfer Protocol (HTTP).**
- To get a page from a Web server, the Web browser issues a **special packet called an 'HTTP Request'** that contains the URL and other information about the Web page requested (see Figure 2-10).
- Once the server receives the HTTP request it processes it and sends back an '**HTTP Response'**, which will be the requested page or an error message (see Figure 2-10).
- This request-response dialogue occurs for ***every*** file transferred between the client and the server.
  - For example, suppose the client requests a Web page with two graphic images.
  - Graphics are stored in separate files from the Web page itself using a different file format than the HTML used for the Web page (in JPEG [Joint Photographic Experts Group] format, for example).
  - In this case, there would be three request-response pairs.
    - First, the browser would issue a request for the Web page, and the server would send the response.
    - Then, the browser would begin displaying the Web page and notice the two graphic files.
    - The browser would then send a request for the first graphic and a request for the second graphic, and the server would reply with two separate HTTP responses, one for each request.

2.3.2 Inside an HTTP Request

- The HTTP request and HTTP response are examples of the packets we introduced in Chapter 1 that are produced by the application layer and sent down to the transport, network, data link, and physical layers for transmission through the network.
- The HTTP response and HTTP request are **simple text files** that take the information provided by the application (e.g., the URL to get) and format it in a structured way so that the receiver of the message can clearly understand it.
- An HTTP request from a Web browser to a Web server has three parts.
- The first two parts are required; the last is optional. The parts are:
  - The ***request line***, which starts with a command (e.g., get), provides the Web page, and ends with the HTTP version number that the browser understands; the version number ensures that the Web server does not attempt to use a more advanced or newer version of the HTTP standard that the browser does not understand.
  - The ***request header***, which contains a variety of optional information such as the Web browser being used (e.g., Chrome) and the date.
  - The ***request body***, which contains information sent to the server, such as information that the user has typed into a form

FIGURE 2-11

An example of a request from a Web browser to a Web server using the HTTP (Hypertext Transfer Protocol) standard



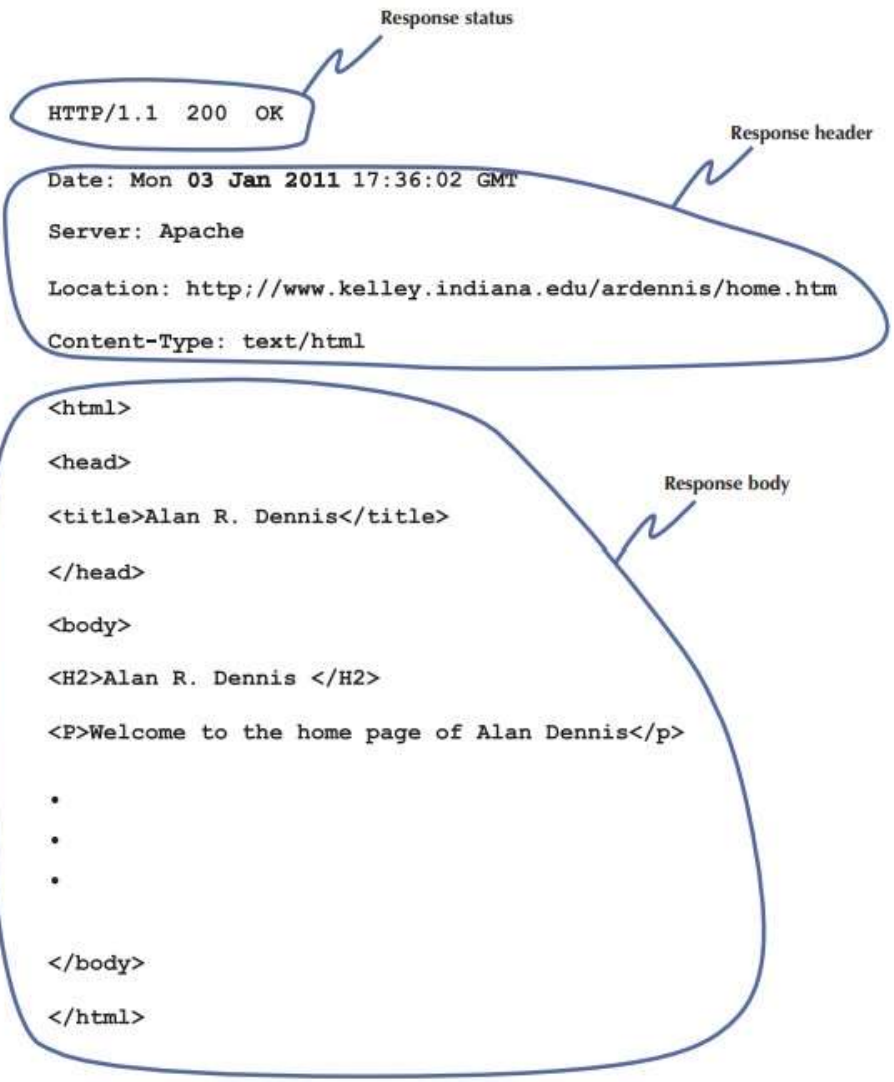
2.3.2 Inside an HTTP Request (contd)

- Figure 2-11 shows an example of an HTTP request for a page on our Web server, formatted using version 1.1 of the HTTP standard.
- This request has only the ***request line*** and the ***request header*** because ***no request body*** is needed for this request.
- This request includes the date and time of the request (expressed in GMT, the time zone that runs through London) and the name of the browser used (Mozilla is the code name for the browser).
- The "***Referrer***" field means that the user obtained the URL for this Web page by clicking on a link on another page, which is a list of faculty at Indiana University (i.e., www.indiana.edu/~isdept/faculty.htm).
- **If the referrer field is blank, then it means the user typed the URL himself or herself.**
- You can see inside HTTP headers yourself at www.rexswain.com/httpview.html

2.3.3 Inside an HTTP Response…

- The format of an HTTP response from the server to the browser is very similar to the HTTP request.
- It, too, has three parts, with the **first required and the last two optional**:
  - The response status, which contains the HTTP version number the server has used, a status code (e.g., 200 means “OK”; 404 means “not found”), and a reason phrase (a text description of the status code).
  - The response header, which contains a variety of optional information, such as the Web server being used (e.g., Apache), the date, and the exact URL of the page in the response.
  - The response body, which is the Web page itself. Figure 2-12 shows an example of a response from our Web server to the request in Figure 2-11.
  - This example has all three parts.
    - The response status reports “OK,” which means the requested URL was found and is included in the response body.
    - The response header provides the date, the type of Web server software used, the actual URL included in the response body, and the type of file.

FIGURE 2-12
An example of a response from a Web server to a Web browser using the HTTP standard



2.3.3 Inside an HTTP Response (contd)

- In most cases, the actual URL and the requested URL are the same, but not always.
  - For example, if you request an URL but do not specify a file name (e.g., www.indiana.edu), you will receive whatever file is defined as the home page for that server, so the actual URL will be different from the requested URL
- The response body in this example shows a Web page in HTML.
- **The response body can be in any format, such as text, Microsoft Word, Adobe PDF, or a host of other formats, but the most commonly used format is HTML.**
- HTML was developed by CERN at the same time as the first Web browser and has evolved rapidly.
- HTML is covered by standards produced by the **IETF (Internet Engineering Task Force)**, but Microsoft keeps making new additions to HTML with every release of its browser, so the **HTML standard keeps changing**

MF 2-2 Top Players in Cloud eMail

- Among the wide variety of applications that organizations use, email is most frequently deployed as SaaS.
- Four major industry players provide email as SaaS: Google, Microsoft, USA.NET, and Intermedia.
- Although cloud-based email seems to appeal more to smaller companies, it provides a cost-effective solution for organizations with up to 15,000 users (as a rule of thumb).
- Google was the first company to enter this market and offered Google Apps, Calendar, and 30 Gb of storage in addition to email.
- Microsoft entered this market in 2008 and offered Microsoft Office 365.
- Microsoft offers not only email but the whole MS Office suite. And, of course, all office applications are accessible from multiple devices.

MF 2-2 Top Players in Cloud eMail (contd).

- USA.NET is a SaaS company that offers Microsoft Exchange and **robust security features** that meet the federal and industry regulations, such as FINRA and HIPAA (what are these?).
- It services approximately 6,000 organizations worldwide that provide financial, health care, energy, and critical infrastructure services.
- In addition, USA.NET offers Security-as-a-Service platform from the cloud.
- Finally, Intermedia, founded in 1995, is the largest Microsoft-hosted Exchange Provider.
  - This was the first company to offer Hosted Microsoft Exchange, and today, it has 90,000 customers and more than 700,000 users.
- Just like Microsoft, **Intermedia delivers the Office Suite in the cloud (Wow)**.
- The prices for the services these companies offer differ quite a bit.
- The cheapest of these four companies is Google, starting at $4.17 per user per month.
- However, these are basic prices that increase with the number of features and services added

2.4 Electronic Mail

- One of the earliest applications on the Internet and is still among the most heavily used today.
- With email, users create and send messages to one user, several users, or all users on a distribution list.
- Most email software enable users to send text messages and attach files from word processors, spreadsheets, graphics programs, and so on.
- Many email packages also permit you to filter or organize messages by priority.
- Several standards have been developed to ensure compatibility between different email software packages.
- Any software package that conforms to a certain standard can send messages that are formatted using its rules.
- Any other package that understands that particular standard can then relay the message to its correct destination; however, if an email package receives a mail message in a different format, it may be unable to process it correctly.
- Many email packages send using one standard but can understand messages sent in several different standards.
- ***The most commonly used standard is SMTP (Simple Mail Transfer Protocol)*** which we will use in this course.
- **Other common standards are X.400 and CMC (Common Messaging Calls).**

2.4.1 How Email Works?

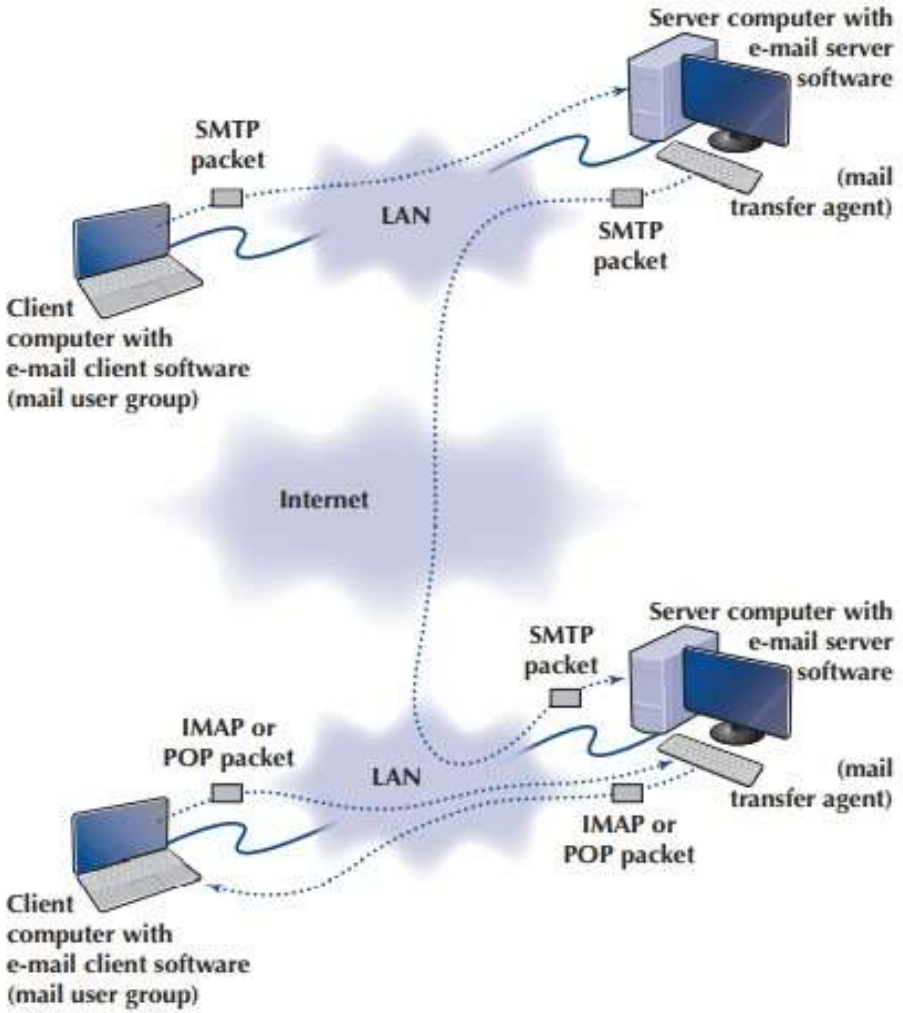
- The Simple Mail Transfer Protocol (SMTP) is the most commonly used email standard simply because it is the email standard used on the Internet.
- Email works similarly to the Web, but it is a bit more complex.
- SMTP email is usually implemented as a **two-tier thick client-server application**, but not always.
- Let's first learn how the normal two-tier thick client architecture works and then quickly contrast that with two alternate architectures.

2.4.1a Two-Tier *Email* Architecture

- With a two-tier thick client-server architecture, each client computer runs an application layer software package called a **mail user agent**, which is usually more commonly called an **email client** (Figure 2-12).
- There are many common email client software packages such as Eudora and Outlook.
- The user creates the email message using one of these email clients, which formats the message into an **SMTP packet** **containing information such as the sender's address and the destination address.**
- The user agent then sends the SMTP packet to a mail server that runs a special application layer software package called **a mail transfer agent**, which is more commonly called mail server software (see Figure 2-13).
- This email server reads the SMTP packet to find the destination address and then sends the packet on its way through the network—often over the Internet—from mail server to mail server, until it reaches the mail server specified in the destination address (see Figure 2-13).
- The mail transfer agent on the destination server then stores the message in the **receiver's mailbox on that server**. The message sits in the mailbox assigned to the user who is to receive the message until he or she checks for new mail.

FIGURE 2-13

How SMTP (Simple Mail Transfer Protocol) email works. IMAP = Internet Message Access Protocol; LAN = local area network



2.4.1a Two-Tier Email Architecture (contd).

- The SMTP standard covers message transmission between mail servers (i.e., mail server to mail server) and between the originating email client and its mail server.
- A different standard is used to communicate between the receiver's email client and his or her mail server.
- Two commonly used standards for communication between email client and mail server are **Post Office Protocol (POP) and Internet Message Access Protocol (IMAP).**
- Although there are several important technical differences between POP and IMAP, **the most noticeable difference is:**
  - That, **before a user can read a mail message with a POP (version 3) email client, the email message must be copied to the client computer's hard disk and *deleted* from the mail server.**
  - **With IMAP, email messages can remain stored on the mail server** after they are read. **IMAP, therefore, offers considerable benefits to users who read their email from many different computers (e.g., home, office, computer labs) because they no longer need to worry about having old email messages scattered across several client computers; all email is stored on the server until it is deleted.**

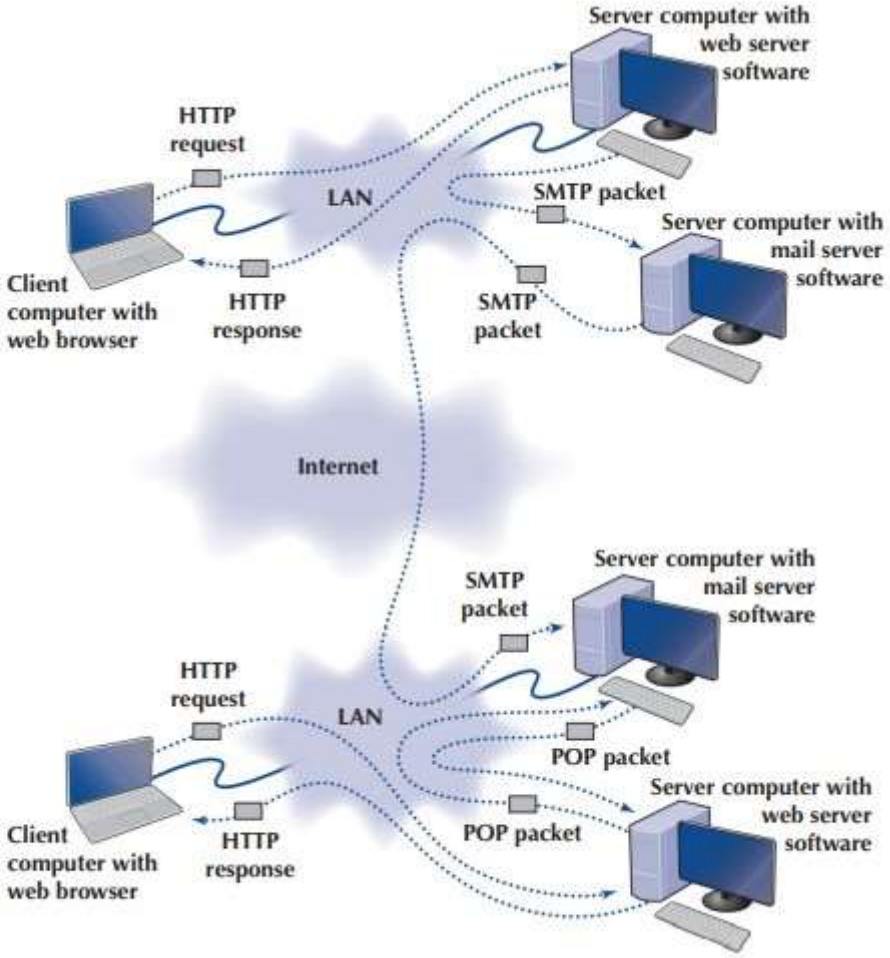
2.4.1a Two-Tier Email Architecture (contd).

- In our example in Figure 2-13, when the receiver next accesses his or her email, the email client on his or her computer contacts the mail server by sending an IMAP or a POP packet that asks for the contents of the user's mailbox.
- When the mail server receives the IMAP or POP request, it converts the original SMTP packet created by the message sender into a POP or an IMAP packet that is sent to the client computer, which the user reads with the email client.
- Therefore, any email client using POP or IMAP must also understand SMTP to create messages.
- POP and IMAP provide a host of functions that enable the user to manage his or her email, such as creating mail folders, deleting mail, creating address books, and so on.
- If the user sends a POP or an IMAP request for one of these functions, the mail server will perform the function and send back a POP or an IMAP response packet that is much like an HTTP response packet.

2.4.1b Three-Tier Email Architecture

- The three-tier think client-server email architecture **uses a Web server and Web browser to provide access to your email.**
- With this architecture, you do not need an email client on your computer. Instead, you use your Web browser.
- This type of email is sometimes called Web-based email and is provided by a variety of companies such as Hotmail and Yahoo!
- You use your browser to connect to a page on a Web server that lets you write the email message by filling in a **form**.
- When you click the send button, your Web browser sends the form information to the **Web server inside an HTTP request** (Figure 2-14).
- The Web server runs a program (written in C or Perl, for example) that takes the information from the HTTP request and builds an SMTP packet that contains the email message.
- Although not important to our example, it also sends an HTTP response back to the client.
- The Web server then sends the SMTP packet to the mail server, which processes the SMTP packet as though it came from a client computer.

FIGURE 2-14
Inside the Web. HTTP = Hypertext Transfer Protocol; IMAP = Internet Message Access Protocol; LAN = local area network; SMTP = Simple Mail Transfer Protocol



2.4.1b Three-Tier Email Architecture (contd).

- The SMTP packet flows through the network in the same manner as before. When it arrives at the destination mail server, it is placed in the receiver's mailbox.
- When the receiver wants to check his or her mail, he or she uses a Web browser to send an HTTP request to a Web server (see Figure 2-14).
- A program on the Web server (in C or Perl, for example) processes the request and sends the appropriate POP request to the mail server.
- The mail server responds with a POP packet, which a program on the Web server converts into an HTTP response and sends to the client.
- The client then displays the email message in the Web browser *Web-based email*

2.4.1b Three-Tier Email Architecture (contd).

- A simple comparison of Figures 2-13 and 2-14 will quickly show that the three-tier approach using a Web browser is much more complicated than the normal two-tier approach.
- So why do it?
- Well, it is simpler to have just a Web browser on the client computer rather than to require the user to install a special email client on his or her computer and then set up the special email client to connect to the correct mail server using either POP or IMAP.
- It is simpler for the user to just type the URL of the Web server providing the mail services into his or her browser and begin using mail.
- This also means that users can check their email from a public computer anywhere on the Internet.

2.4.1b Three-Tier Email Architecture (contd).

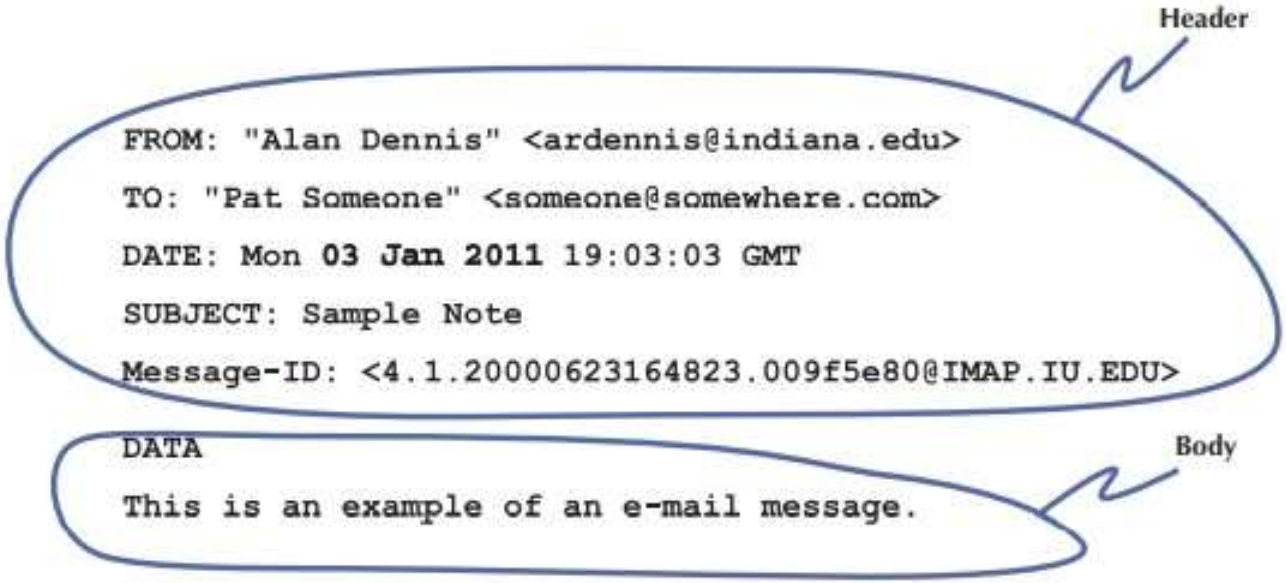
- It is also important to note that the sender and receiver do not have to use the same architecture for their email.
  - The sender could use a two-tier client-server architecture, and the receiver, a host-based or three-tier client-server architecture.
  - Because all communication is standardized using SMTP between the different mail servers, how the users interact with their mail servers is unimportant.
  - Each organization can use a different approach.
  - In fact, there is nothing to prevent one organization from using all three architectures simultaneously.
  - At Indiana University, email is usually accessed through an email client (e.g., Microsoft Outlook) but is also accessed over the Web because many users travel internationally and find it easier to borrow a Web browser with Internet access than to borrow an email client and set it up to use the Indiana University mail server.
  - *What is the setup at KFUPM?*

2.4.2 Inside an SMTP Packet

- SMTP defines how message transfer agents operate and how they format messages sent to other message transfer agents.
- An SMTP packet has two parts:
  - The **header**, which lists source and destination email addresses (possibly in text form [e.g., “Pat Smith”]) as well as the address itself (e.g., psmith@somewhere.com), date, subject, and so on.
  - The **body**, which is the word DATA, followed by the message itself.
- Figure 2-15 shows a simple email message formatted using SMTP.
  - The header of an SMTP message has a series of fields that provide specific information, such as the sender’s email address, the receiver’s address, date, and so on.
  - The information in quotes on the ***from*** and ***to*** lines is ignored by SMTP; only the information in the angle brackets is used in email addresses.
  - The message ID field is used to provide a unique identification code so that the message can be tracked.
  - The message body contains the actual text of the message itself.

FIGURE 2-15

An example of an email message using the SMTP (Simple Mail Transfer Protocol) standard



2.4.3 Attachments in Multipurpose Internet Mail Extension (MIME)

- As the name suggests, SMTP is a **simple** standard that permits only the transfer of text messages.
- It was developed in the early days of computing, when no one had even thought about using email to transfer nontext files such as graphics or word processing documents.
- Several standards for nontext files have been developed that can operate together with SMTP, such as Multipurpose Internet Mail Extension (MIME), uuencode, and binhex.
- Each of the standards is different, but all work in the same general way.
- The MIME software, which **exists** as part of the email client, takes the nontext file such as a PowerPoint graphic file, and translates each byte in the file into a special code that looks like regular text.
- This encoded section of “text” is then labeled with a series of special fields understood by SMTP as identifying a MIME-encoded attachment and specifying information about the attachment (e.g., name of file, type of file).
- When the receiver’s email client receives the SMTP message with the MIME attachment, it recognizes the MIME “text” and uses its MIME software (that is part of the email client) to translate the file from MIME “text” back into its original format.

2.5 OTHER APPLICATIONS

- There are literally thousands of applications that run on the Internet and on other networks. Most application software that we develop today, whether for sale or for private internal use, runs on a network. We could spend years talking about different network applications and still cover only a small number.
- Fortunately, most network application software works in much the same way as the Web or email. In this section, we will briefly discuss only three commonly used applications: Telnet, instant messaging (IM), and video conferencing

2.5.1 Telnet

- Telnet enables users to log in to servers (or other clients).
- It requires an **application** layer program on the client computer and an application layer program on the server or host computer.
- Once Telnet makes the **connection** from the client to the server, ***you must use the account name and password of an authorized user to log in.***
- Although Telnet was developed in the very early days of the Internet (actually, the very first application that tested the connectivity on ARPANET was Telnet), it is still widely used today.
- Because it was developed so long ago, Telnet assumes a host-based architecture.
- Any keystrokes you type using Telnet are sent to the server for processing, and then the server instructs the client what to display on the screen.
- One of the most frequently used Telnet software packages is PuTTY.
  - PuTTY is open source
  - Although TTY is a commonly used abbreviation for “terminal” in UNIX-based systems). The very first Telnet applications posed a great security threat because every key stroke was sent over the network as plain text.
  - PuTTY uses secure shell (SSH) encryption when communicating with the server so that no one can read what is typed.
  - An additional advantage of PuTTY is that it can run on multiple platforms, such as Windows, Mac, or Linux.
  - Today, PuTTY is routinely used by network administrators to log in to servers and routers to make configuration changes

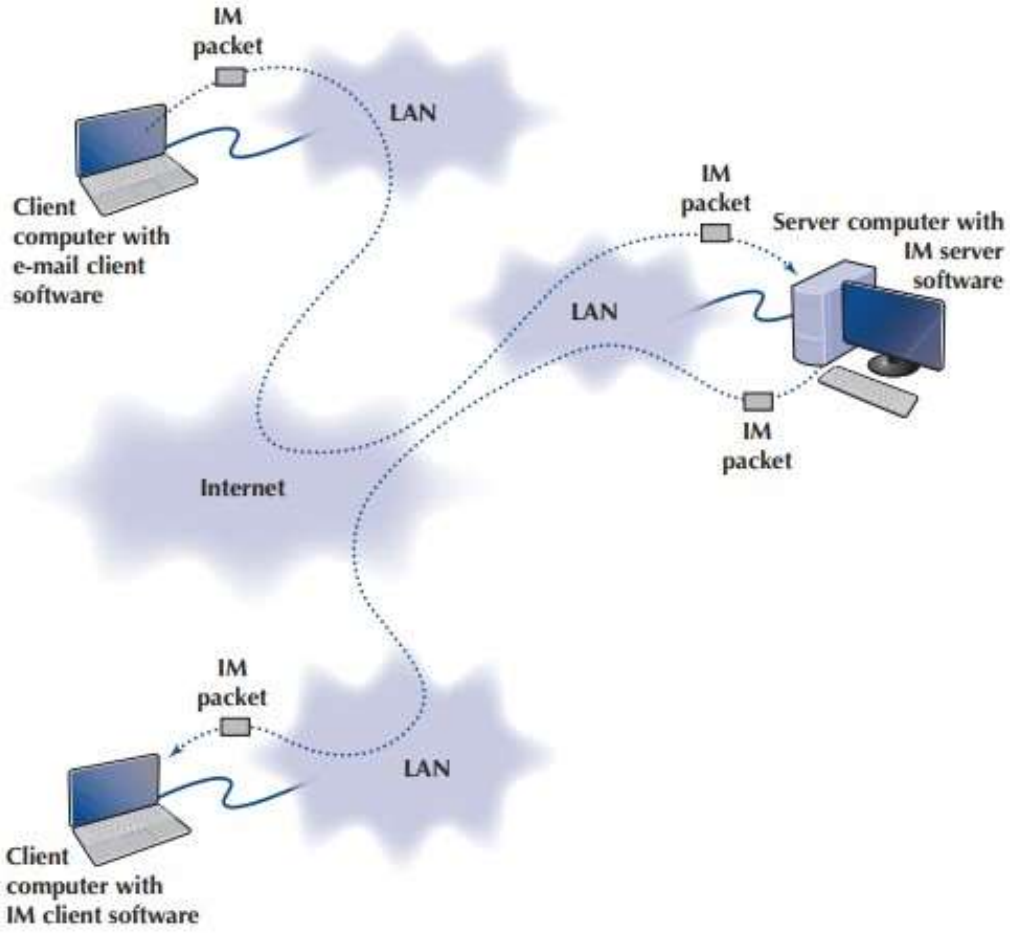
2.5.2 Instant Messaging

- One of the fastest growing Internet applications has been instant messaging (IM).
- With IM, you can exchange real-time typed messages or chat.
- Some IM software also enables you to verbally talk with your friends in the same way you might use the telephone or to use cameras to exchange real-time video in the same way you might use a videoconferencing system.
- Several types of IM currently exist, including Google Talk and AOL Instant Messenger.
- Instant messaging works in much the same way as the Web.
- The client computer needs an IM client software package, which communicates with an IM server software package that runs on a server.
- When the user connects to the Internet, the IM client software package sends an IM request packet to the IM server, informing it that the user is now online.
- The IM client software package continues to communicate with the IM server to monitor what other users have connected to the IM server.

2.5.2 Instant Messaging (contd).

- When one of your friends connects to the IM server, the IM server sends an IM packet to your client computer so that you now know that your friend is connected to the Internet.
- The server also sends a packet to your friend's client computer so that he or she knows that you are on the Internet.
- With the click of a button, you can both begin chatting.
- When you type text, your IM client creates an IM packet that is sent to the IM server (Figure 2-16).
- The server then retransmits the packet to your friend.
- Several people may be part of the same chat session, in which case the server sends a copy of the packet to all of the client computers.
- IM also provides a way for different servers to communicate with one another, and for the client computers to communicate directly.
- Additionally, IM will do voice and video.

FIGURE 2-16
How instant messaging (IM) works. LAN = local area network



2.5.3 Video Conferencing

- Videoconferencing provides real-time transmission of video and audio signals to enable people in two or more locations to have a meeting.
- Sometimes, videoconferences are held in special-purpose meeting rooms with one or more cameras and several video display monitors to capture and display the video signals (Figure 2-17).
- Special audio microphones and speakers capture and play audio signals.
- The audio and video signals are combined into one signal transmitted through a MAN or WAN to people at the other location.
- Most of this type of videoconferencing involves two teams in two separate meeting rooms, but some systems can support conferences of up to eight separate meeting rooms.
- Some advanced systems provide **telepresence**, which is of such high quality that you feel you are face-to-face with the other participants.
- The fastest growing form of videoconferencing is desktop videoconferencing.
  - Small cameras installed on top of each computer permit meetings to take place from individual offices (Figure 2-18).
  - Special application software (e.g., Yahoo! IM, Skype, Net Meeting, Zoom, MSTeams) is installed on the client computer and transmits the images across a network to application software on a videoconferencing server.
  - The server then sends the signals to the other client computers that want to participate in the videoconference

2.5.3 Video Conferencing (contd).

- In some cases, the clients can communicate with one another without using the server.
- The cost of desktop videoconferencing ranges from less than $20 per computer for inexpensive systems to more than $1,000 for high-quality systems.
- Some systems have integrated conferencing software with desktop videoconferencing, enabling participants to communicate verbally and, by using applications such as white boards, to attend the same meeting while they are sitting at the computers in their offices.
- The transmission of video requires a lot of network **capacity**.
- Most videoconferencing uses **data compression** to reduce the amount of data transmitted.
- ***Surprisingly, the most common complaint is not the quality of the video image but the quality of the voice transmissions.***
- Most videoconferencing systems were originally developed by vendors using different formats, so many products were **incompatible**.
- The best solution was to ensure that all hardware and software used within an organization was supplied by the same vendor and to hope that any other organizations with whom you wanted to communicate used the same equipment.

2.5.3 Video Conferencing (contd).

- Today, three standards are in common use:
  - H.320, H.323, and MPEG-2 (also called ISO 13818-2).
  - Each of these standards was developed by different organizations and is supported by different products.
  - They are incompatible, although some **application** software packages understand more than one standard.
  - H.320 is designed for room-to-room videoconferencing over high-speed telephone lines.
  - H.323 is a family of standards designed for desktop videoconferencing and just simple audio conferencing over the Internet.
  - MPEG-2 is designed for faster connections, such as a LAN or specially designed, privately operated WAN.
  - **Webcasting** is a special type of one-directional videoconferencing in which content is sent from the server to the user.
    - The developer creates content that is downloaded as needed by the users and played by a plug-in to a Web browser.
    - At present, there are **no standards** for Webcast technologies, but the products by RealNetworks.com are the *de facto* standards.

MF-2-4 Cloud-Hosted Virtual Desktops

- While cloud computing started on the server side, it quickly is moving to the client side—the desktop.
- Imagine that you work for a multinational organization and fly several times a year to different parts of the world to do your job.
- Your organization doesn't want you to travel with a laptop because they fear that you can lose the laptop with the data on it but they want you to be able to log in to any desktop in any office around the world and have your desktop appear on the screen.
- Well, with the cloud technology, this is possible, and many companies are taking advantage of this new service.
- Could you guess its name? Yes, Desktop-as-a-Service (DaaS).
- Several companies offer DaaS without the infrastructure cost and with reduced complexity of deploying desktops.
- This service works as a monthly subscription service and includes data center hardware and facilities and also security.
- Dell DaaS on Demand and Amazon WorkSpaces are among the service providers of Daas

2.6 IMPLICATIONS FOR MANAGEMENT

- The first implication for management from this chapter is that the primary purpose of a network is to provide a **worry-free** environment in which applications can run.
  - **The network itself does not change the way an organization operates; it is the applications that the network enables that have the potential to change organizations.**
  - If the network does not easily enable a wide variety of applications, this can severely limit the ability of the organization to compete in its environment.

2.6 IMPLICATIONS FOR MANAGEMENT (contd)

- The second implication is that over the past few years, there has been a dramatic increase in the number and type of applications that run across networks.
  - In the early 1990s, networks primarily delivered email and organization-specific application traffic (e.g. accounting transactions, database inquiries, inventory data).
  - Today's traffic contains large amounts of email, Web packets, videoconferencing, telephone calls, instant messaging, music, and organization-specific application traffic.
  - Traffic has been growing much more rapidly than expected, and each type of traffic has different implications for the best network design, making the job of the network manager much more complicated. Most organizations have seen their network operating costs grow significantly even though the cost per packet (i.e., the cost divided by the amount of traffic) has dropped significantly over the last 10 years. Experts predicted that by 2015, video will be the most common type of traffic on the Web, passing email and Web, which are the leading traffic types today.

SUMMARY

- The Application Architectures
  - There are four fundamental application architectures.
- World Wide Web One of the fastest growing Internet applications is the Web, which was first developed in 1990.
- Electronic Mail.