Networks, Security, and Privacy

158.235

A/Prof. Julian Jang-Jaccard

Massey University

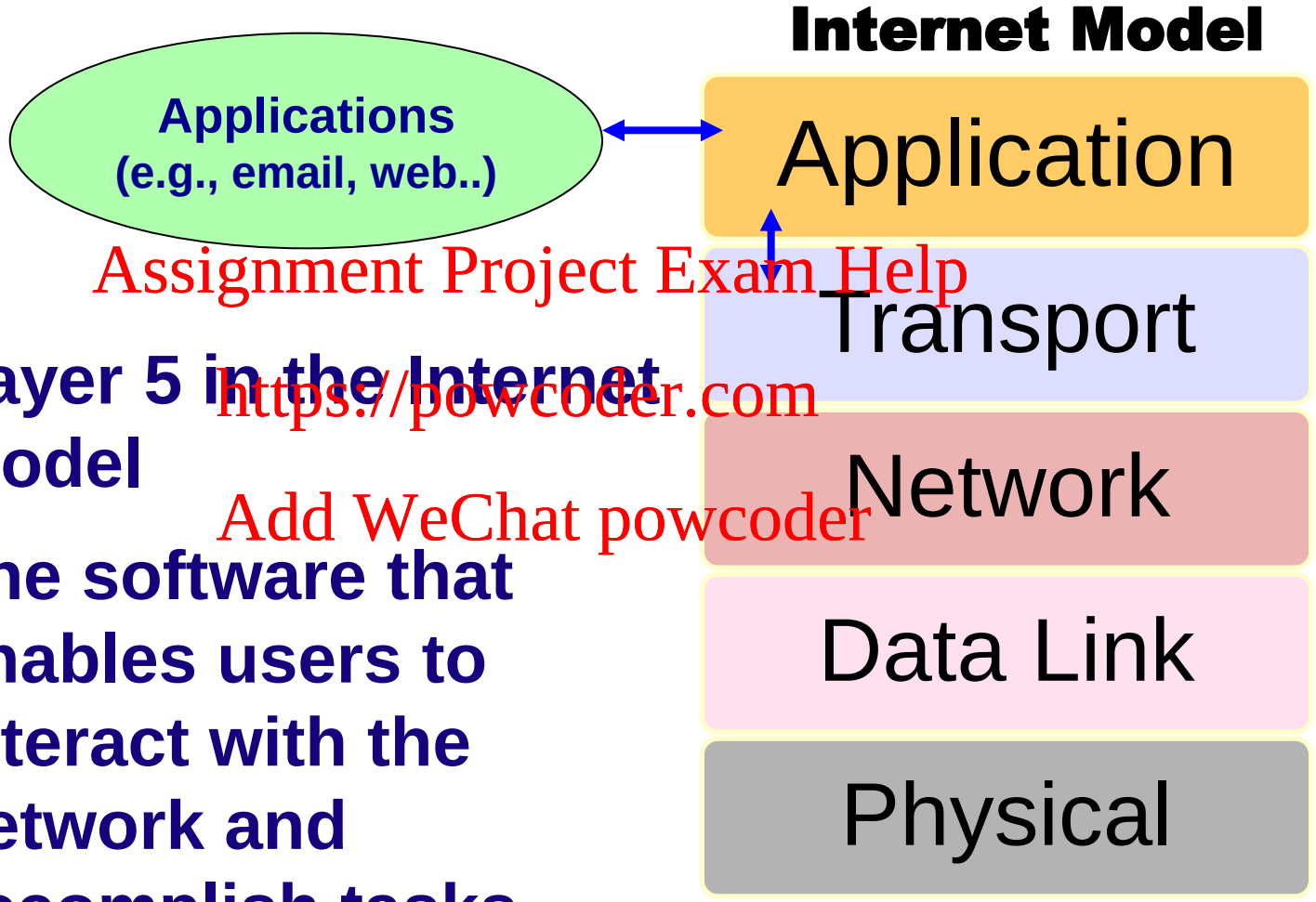
Application Layer

Reading: Chapter 2 in the prescribed textbook

Outline

- **Application Architecture**
- **Application Layer Services**
  - **The Web: HTTP**
  - **Email: SMTP, POP**
  - **Other Application-Layer Protocols**

Application Layer



- **Layer 5 in the Internet model**
- **The software that enables users to interact with the network and accomplish tasks**

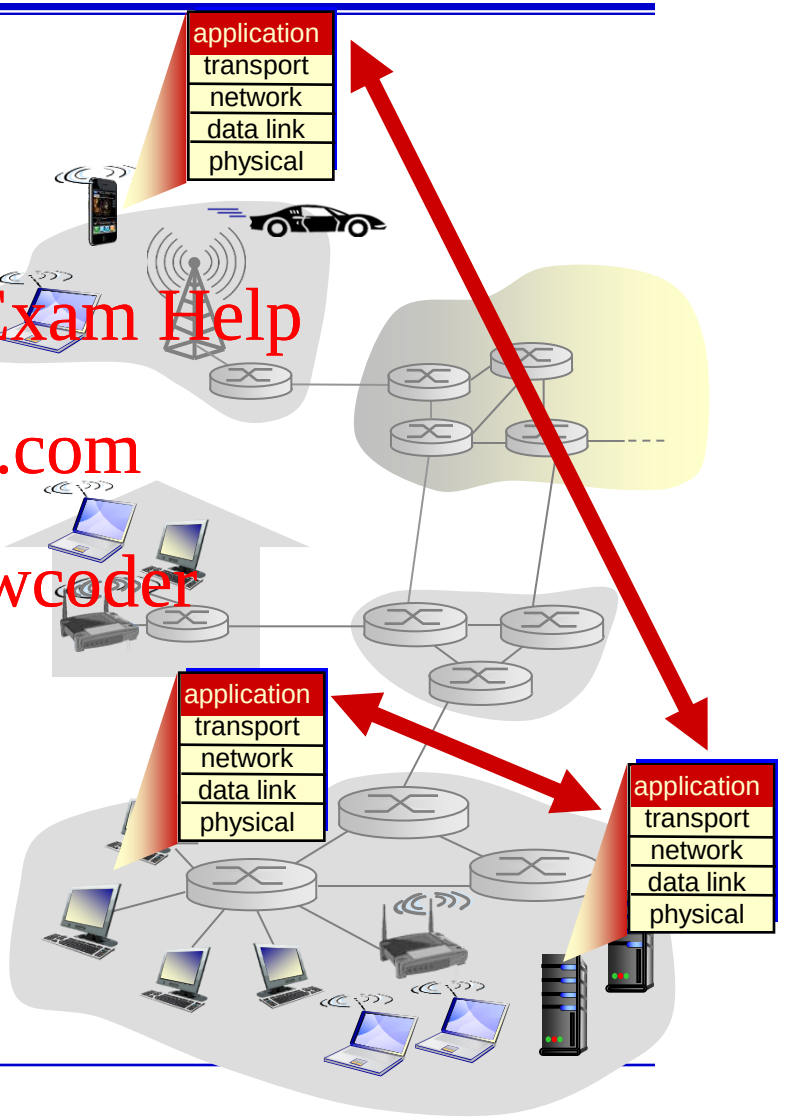
What is a network application?

Is a program that:

- run on (different) *end systems*
- communicate over network

no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allow for rapid app development, propagation



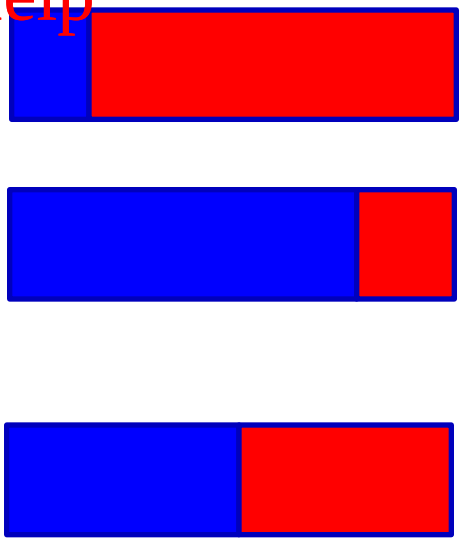
Application Architecture

- The way the functions of the application layer are spread out across the client and the server

- Four components of applications:
  1. Data Storage
  2. Data Access Logic
  3. Application Logic
  4. Presentation Logic

Application Architectures

- **Who is doing what between the clients and servers?**
  - **Host-based Architectures**
    - **Server performs almost all functions**
  - **Client-based architectures**
    - **Client performs most functions**
  - **Client-server architectures**
    - **Functions shared between client and server (including Cloud Computing)**
  - **Peer to peer architectures**
    - **Computers are both clients and servers**



Host-Based Architecture

- **Common in the 1960s with mainframes and terminals**
- **Server contains all components ("server-based")**

CLIENT

SERVER

Presentation Logic
Application Logic
Data Access Logic
Data Storage

Host-Based Architecture

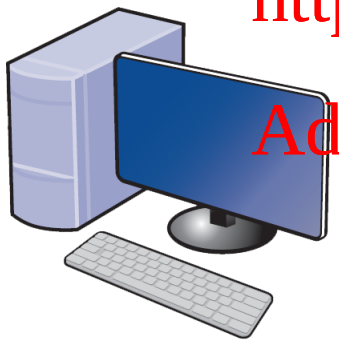
- **Advantages**
  - **Very simple**
  - **Single point of control**
- **Disadvantages**
  - **Host (server) can become a bottleneck**
  - **Upgrades typically expensive ('lumpy architecture')**

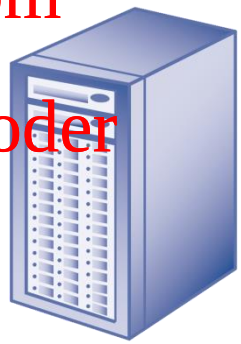
Client-Based Architecture

- **Most common in the 1980s with popularity of PC**
- **Client contains presentation, application, and data access logic while server stores the data**

CLIENT

SERVER





Presentation Logic
Application Logic
Data Access Logic

Data Storage

Client-Based Architecture

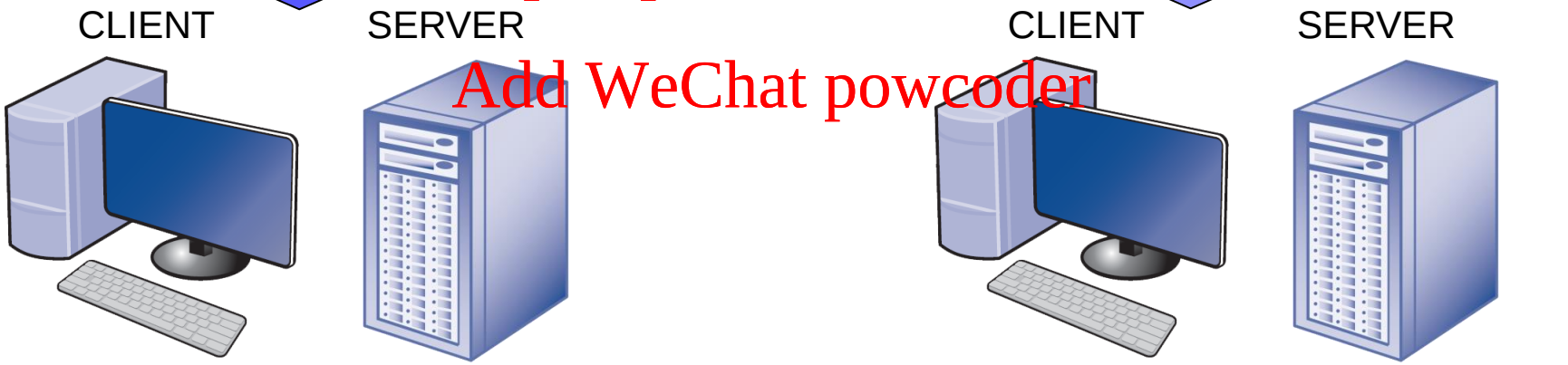
- **Advantages**
  - **Hardware and applications less expensive**
  - **Simple architecture**
- **Disadvantages**
  - **Data must travel back and forth between server and client**

Client-Server Architecture

- **Most common architecture today**
- **Thin clients are easier to manage, thick clients have more functionality**

Thick-Client Architecture

Thin-Client Architecture



Presentation Logic
Application Logic

Data Access Logic
Data Storage

Presentation Logic

Application Logic
Data Access Logic
Data Storage

Client-Server Architecture

- **Advantages**
  - **More efficient because of distributed processing**
  - **Allows hardware/software from different vendors to be used together**
  - **Less bandwidth required**
- **Disadvantages**
  - **May be challenges in configuring hardware/software from different vendors to work together**
  - **In many cases, middleware is required**

Client-Server Architecture

Middleware is software acts as an intermediary by "sitting between" client and server applications

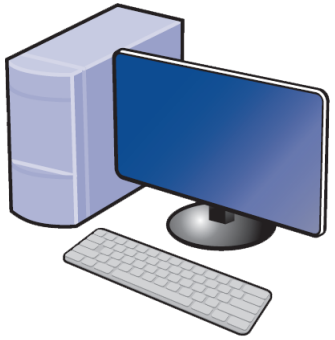
1.Provides a standard way of translating between software from different vendors

2.Manages message transfers. Insulates network changes from the clients (e.g., adding a new server)

Client-Server Architecture

- **Example of two-tier architecture**

CLIENT SERVER



Presentation Logic

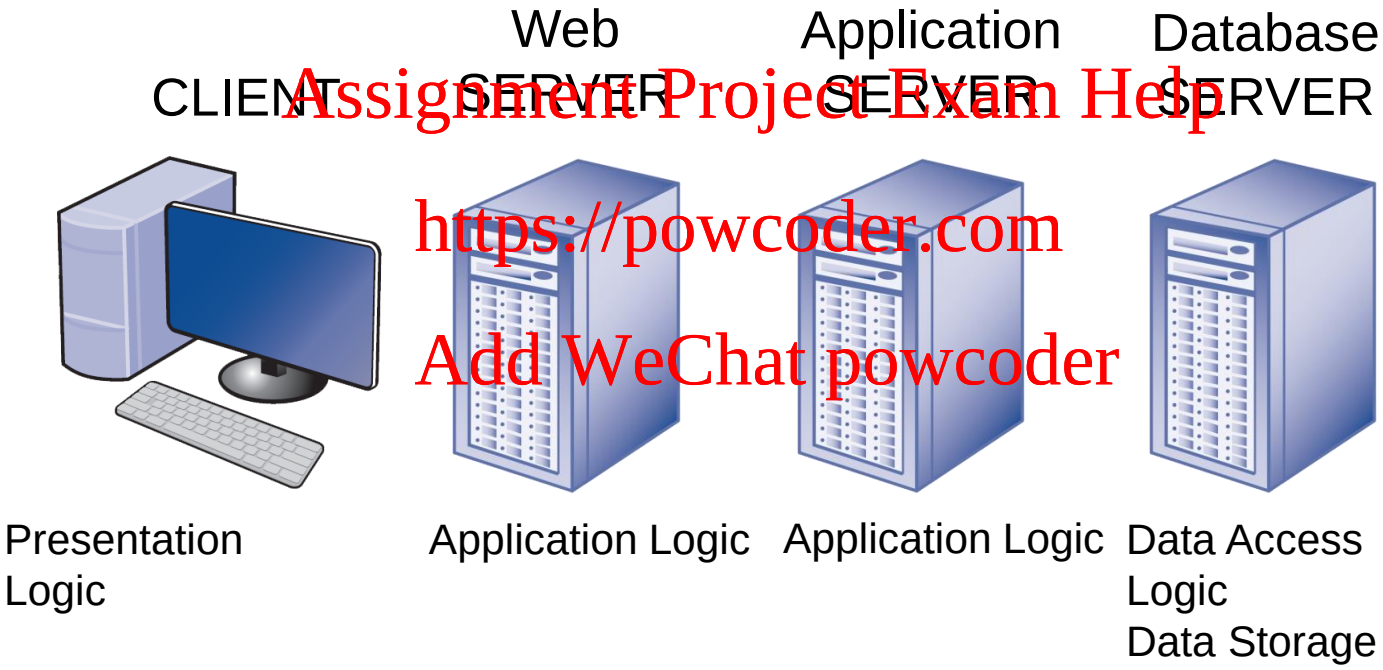
Application Logic
Data Access Logic
Data Storage

Client-Server Architecture

- **Example of three-tier architecture**

Client-Server Architecture

- **Example of n-tier architecture**

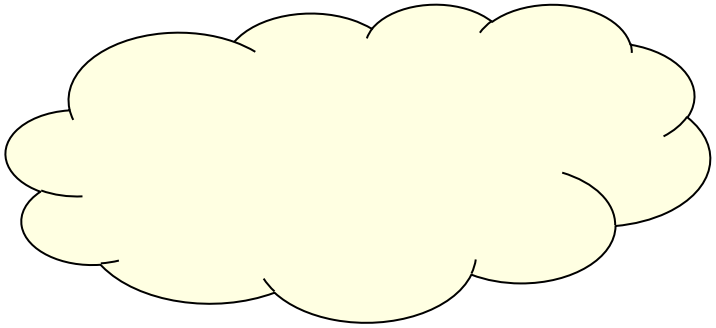


Tiered Client-Server Architecture

- **Advantages**
  - **Better load balancing: More evenly distributed processing**
  - **More scalable: Only servers experiencing high demand need be upgraded**
- **Disadvantages**
  - **Heavily loaded network: More distributed processing necessitates more data exchanges**
  - **Difficult to program and test due to increased complexity**

Cloud Computing

- **Cloud Computing is the general term for enabling access to computing services over the network (most commonly the Internet)**
- **Models of cloud computing define who manages each application function and associated hardware/software**



Case Study: Building a Business

OLD WAY

- Significant Upfront Investment
- Results in months
- Large total investment

CLOUD WAY

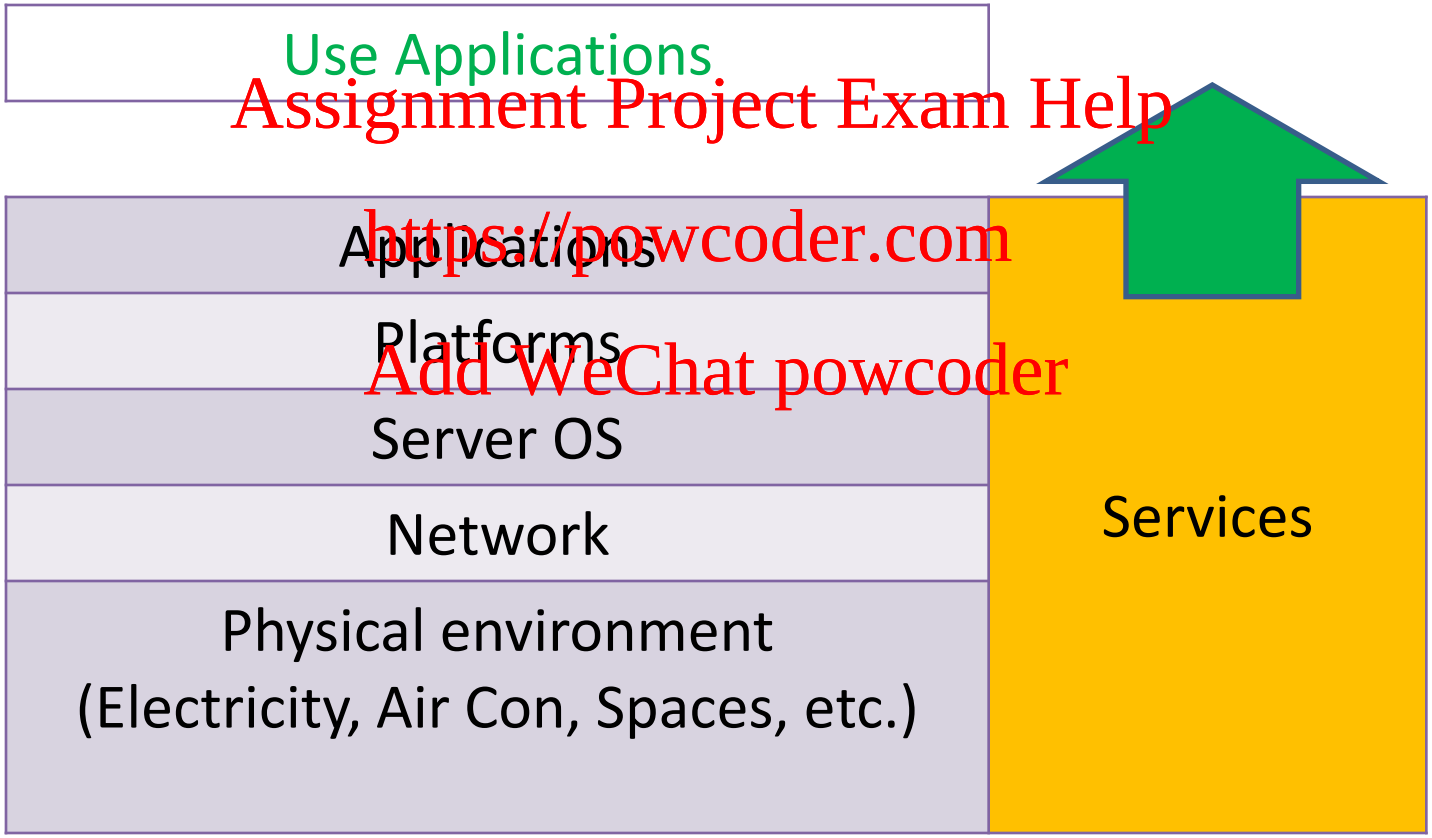
- No Upfront investment
- Results in days
- Trivial total investment

Cloud Computing: Key benefits

- **Huge Resources**
  - **Available for everyone with a small fee**
  - **Leasing model compared to buying model**
- **No Commitment**
  - **No over provisioning (waste of capital)**
  - **No under provisioning (waste of users)**
- **Pay by use**
  - **Pay only for actual resources consumed**

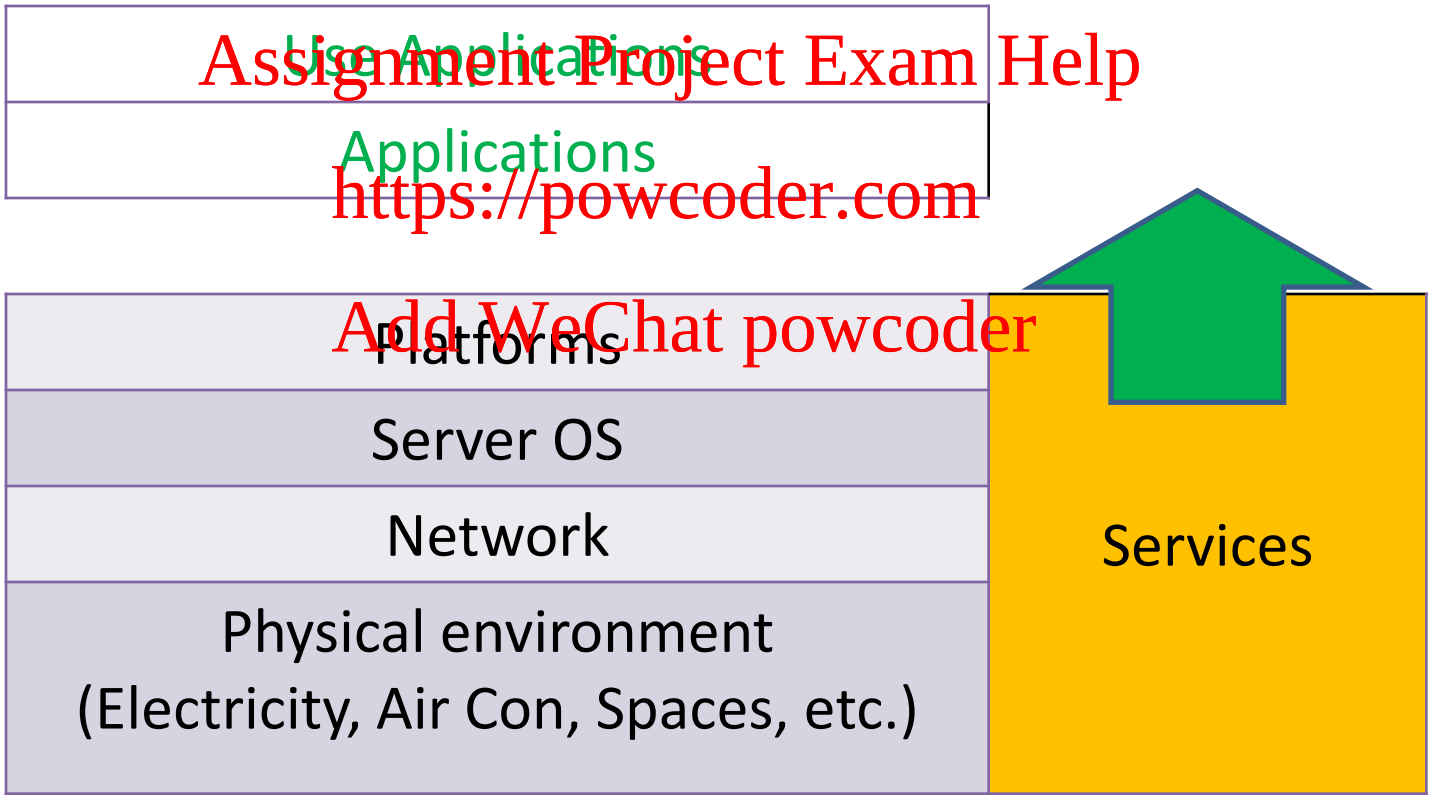
Software as a Service (SaaS)

- **Provides all application components and associated hardware/software**



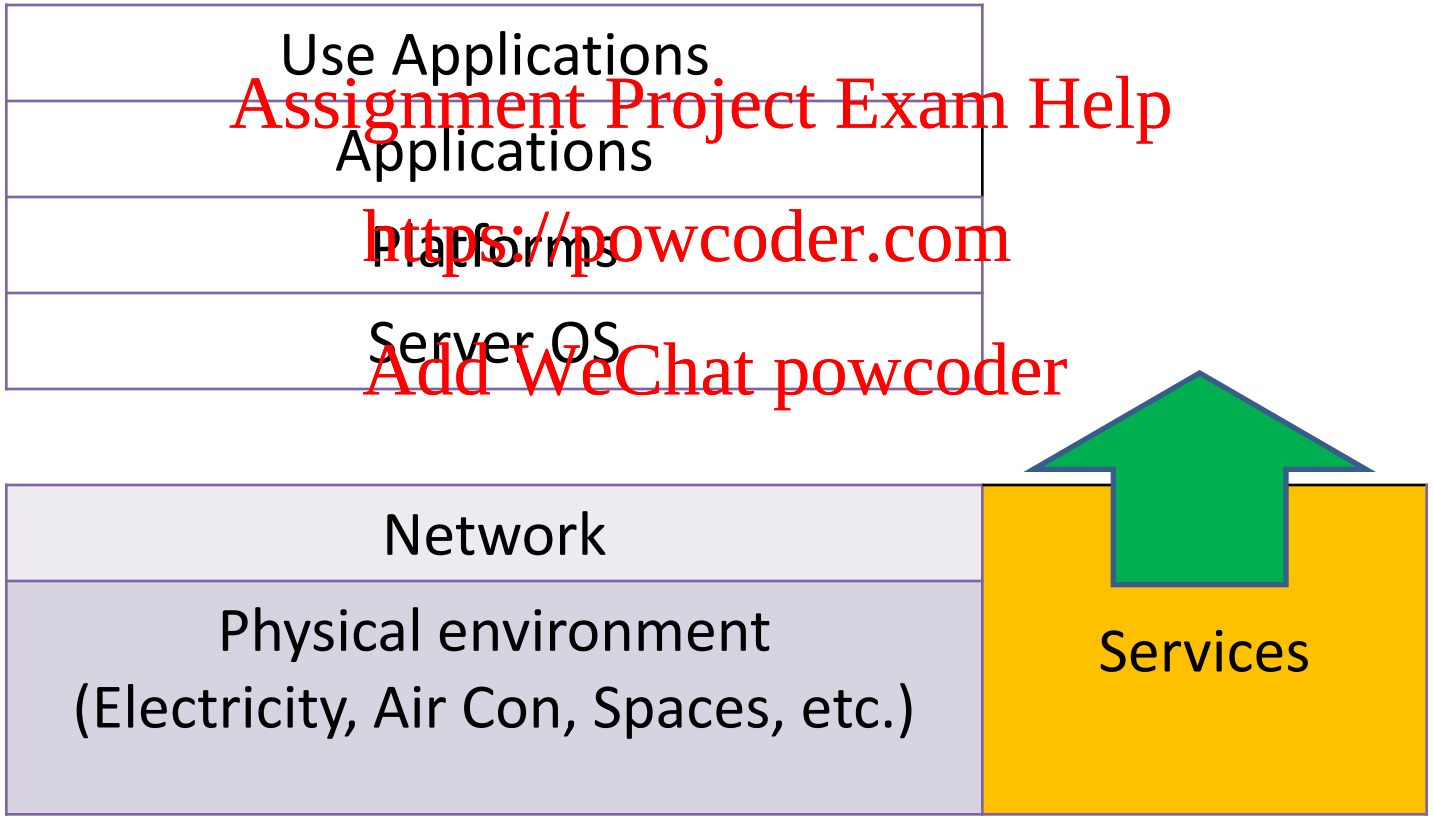
Platform as a Service (PaaS)

- **Provides computing platforms (OS, database, webserver etc.,)**

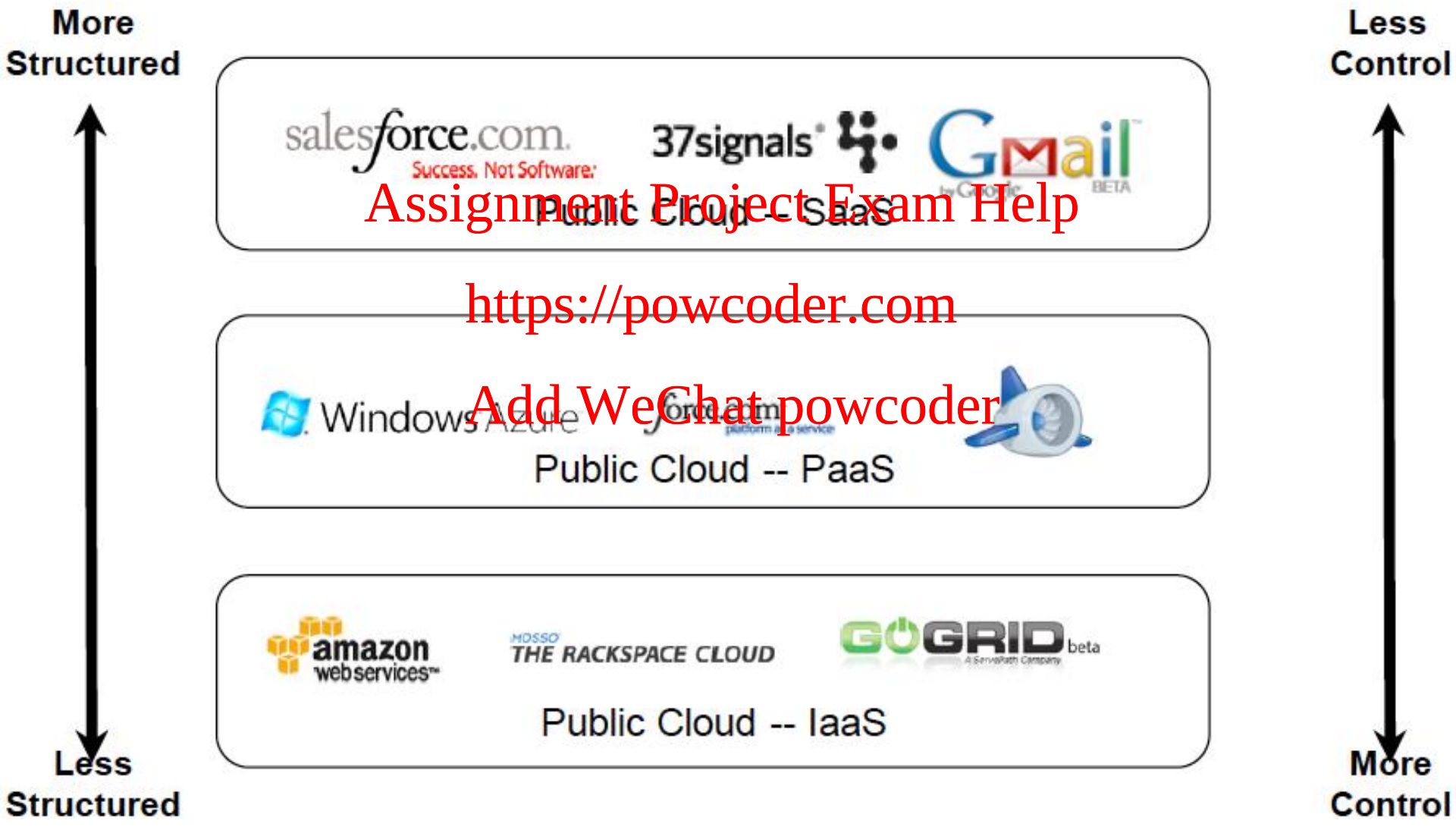


Infrastructure as a Service (IaaS)

- **All hardware is outsourced**



Cloud Computing Delivery

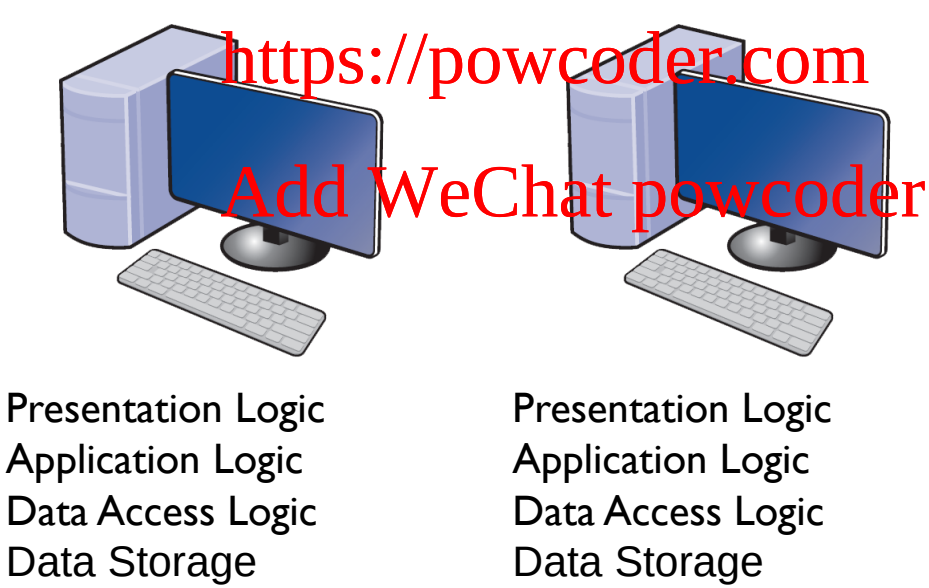


Cloud Computing

	Traditional Thin-Client Client-Server		Infrastructure as a Service (IaaS)		Platform as a Service (Paas)		Software as a Service (SaaS)	
	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

Peer to Peer Architecture

- **An older architecture that became popular again with Napster, BitTorrent, etc., in early 2000s.**
- **All devices can serve as a client and a server**



Peer to Peer Architecture

- **Advantages:**
  - **Data can be stored anywhere on the network**
  - **Very resilient to failure**
  - **Distributes bandwidth requirements**
- **Disadvantages:**
  - **Finding the stored data is hard (no centralised control)**
  - **Security (everything is everywhere)**

Criteria for Choosing Architecture

- **Development Costs**
  - **Tools, Software packages etc.,**
  - **Cost of servers, clients, and circuits (infra, platform etc.,)**
- **Scalability**
  - **Ability to increase (or decrease) in computing capacity as network demand changes**
  - **Easier in client-server architectures**
- **Reliable**
  - **Ability to recover from failures**

Outline

- Application Architecture
- Application Layer Services
  - The Web: HTTP
  - Email: SMTP, POP
  - Other Application-Layer Protocols

Web and HTTP

First, a review…

- ***web page*** **consists of** ***objects***
- **object can be HTML file, JPEG image, Java applet, audio file,…**
- **web page consists of** ***base HTML-file*** **which includes** ***several referenced objects***
- **each object is addressable by a** ***URL,*** **e.g.,**

`www.someschool.edu/someDept/pic.gif`

host name | path name

HTTP Overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - ***client:*** browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - ***server:*** Web server sends (using HTTP protocol) objects in response to requests

HTTP Overview

uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests

aside

protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

HTTP Connections

non-persistent HTTP

- **at most one object sent over TCP connection**
  - **connection then closed**
- **downloading multiple objects required multiple connections**

persistent HTTP

- **multiple objects can be sent over single TCP connection between client, server**

Non-persistent HTTP

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

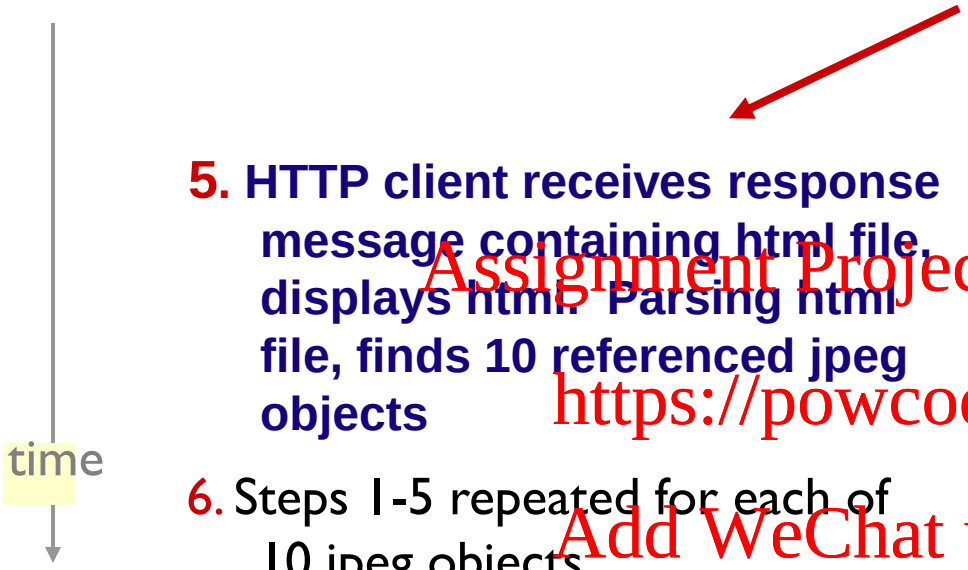
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time

Non-persistent HTTP



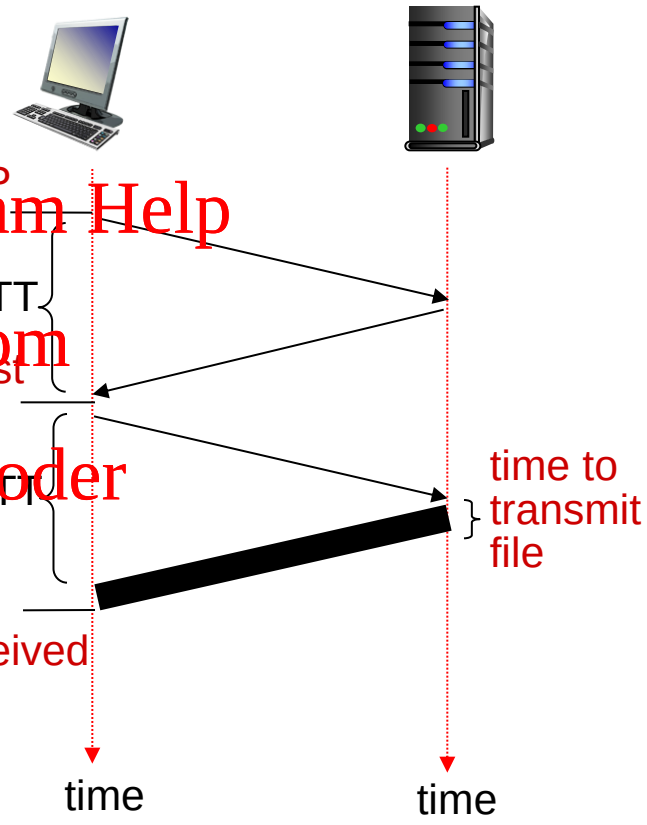
Non-persistent HTTP: response

RTT (definition): time for a small packet to travel from client to server and back

HTTP response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time
- non-persistent HTTP response time =

  2RTT+ file transmission time



Persistent HTTP

non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP Request Message

Request line
(command, URL, HTTP version number)

required

Request header
(information on the browser, date, and the referring page)

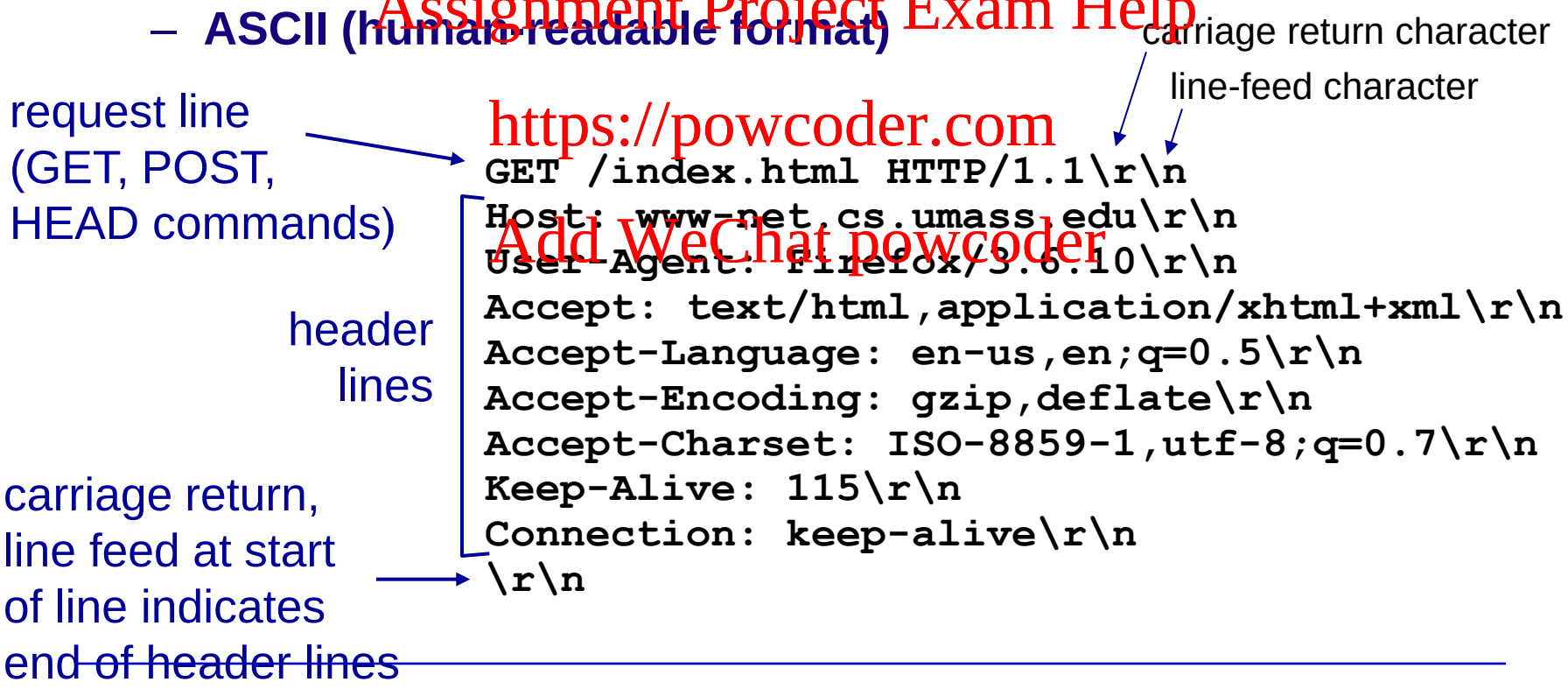
optional

Request body
(information sent to the server, such as from a form, mainly with POST command)

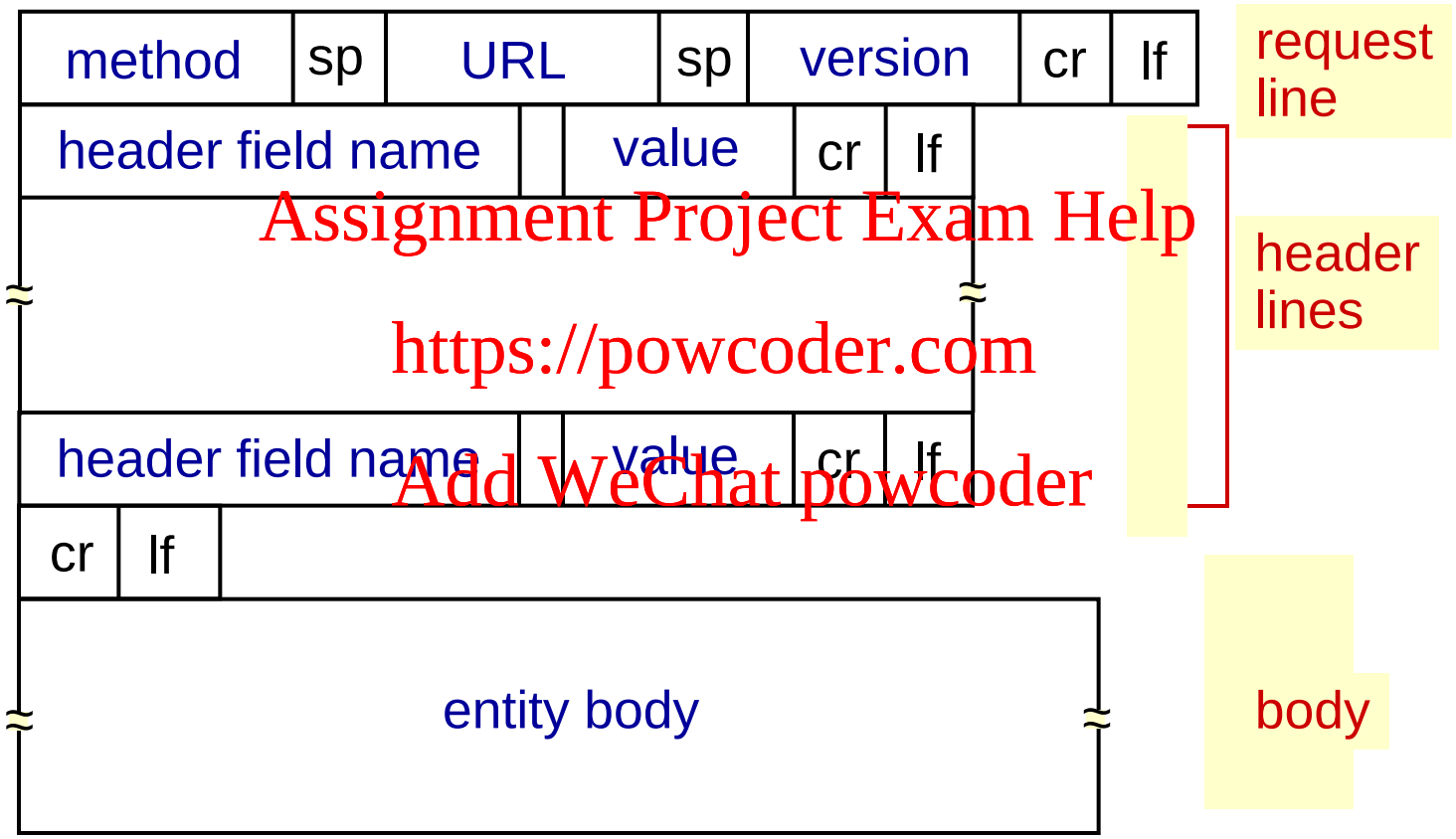
optional

HTTP request message

- **two types of HTTP messages: *request*, *response***
- **HTTP request message:**
  - **ASCII (human-readable format)**



HTTP request message: format



HTTP Response Message

Response status
(http version number, status code, reason)

optional

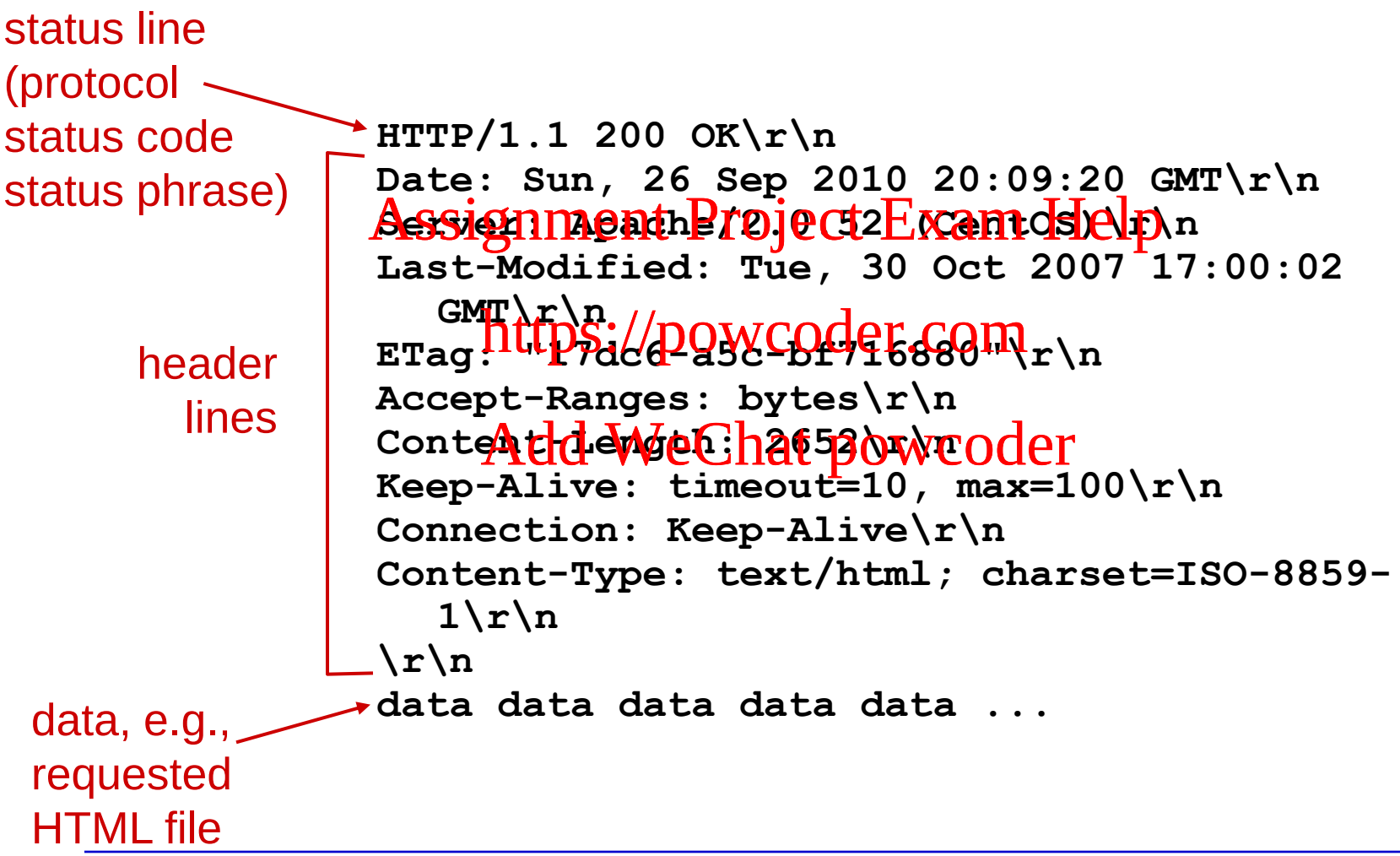
Response header
(information on the server, date, URL of the page retrieved, format used)

optional

Response body
(requested web page)

required

HTTP Response Message



HTTP Response: status codes

- status code appears in 1st line in server-to-client response message.
- some sample codes:

200 OK

- request succeeded, requested object later in this msg

301 Moved Permanently

- requested object moved, new location specified later in this msg (Location:)

400 Bad Request

- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Email

- **Mail User Agent (MUA): Mail Client**
  - Formal name for mail client software
  - e.g., Outlook, Apple Mail, Thunderbird
- **Mail Transfer Agent (MTA): Mail Server**
  - Formal name for mail server software
  - e.g., Sendmail, Postfix
- **Simple Mail Transfer Protocol (SMTP)**
  - Protocol used to send a message to a MTA
  - Originally only handled text files
- **Internet Message Access Protocol (IMAP) or Post Office Protocol (POP)**
  - Protocols used by a MUA to retrieve messages from an MTA

Email in Action

1) Alice uses MUA to compose message "to" `bob@someschool.edu`

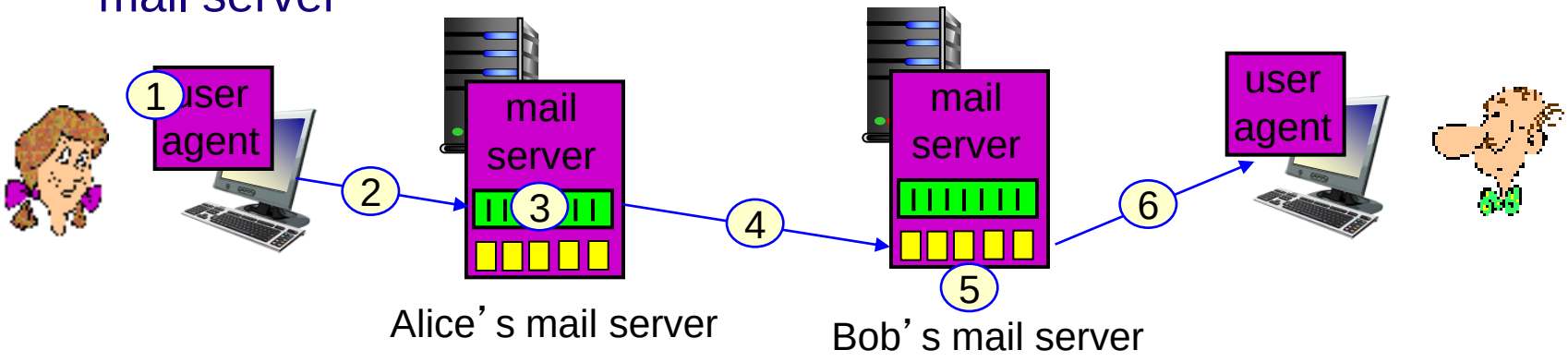
2) Alice's MUA sends message to her mail server; message placed in message queue

3) client side of SMTP opens TCP connection with Bob's mail server

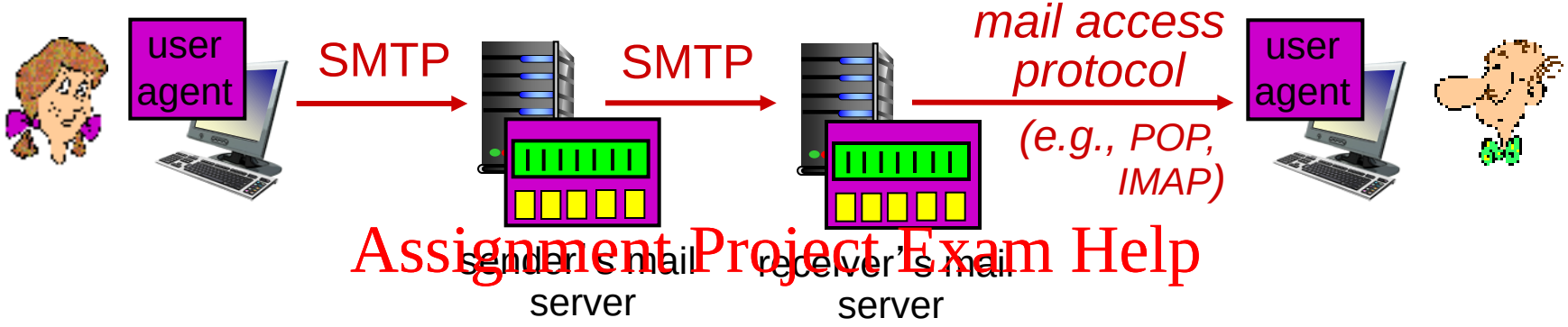
4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message



Mail Access Protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored msgs on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 vs IMAP

POP3

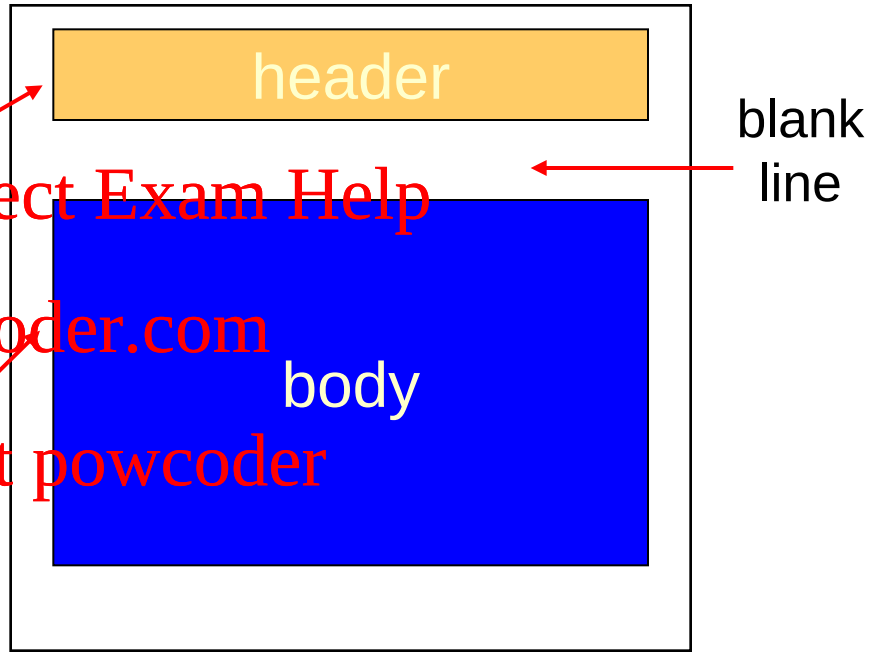
- previous example uses POP3 “download and delete” mode
  - Bob cannot re-read e-mail if he changes client
- POP3 “download-and-keep”: copies of messages on different clients
- POP3 is stateless across sessions

IMAP

- keeps all messages in one place: at server
- allows user to organize messages in folders
- keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name

Email Message Format

- **SMTP Message format**
  - **RFC 822: standard for text message format**
- **Header lines**
  - **Contains information about the message (e.g., to, from, subject)**
- **Body Section**
  - **Contains the 'content of the message'**
  - **Begins with the 'DATA' keyword**
  - **Only uses ASCII characters**



POP3 protocol

authorization phase

- client commands:
  - `user:` declare username
  - `pass:` password
- server responses
  - `+OK`
  - `-ERR`

transaction phase, client:

- `list:` list message numbers
- `retr:` retrieve message by number
- `dele:` delete
- `quit`

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

MIME

- **Multipurpose Internet Mail Extension**
  - **A graphic-capable mail transfer agent protocol (to send graphical information in addition to text)**
    - **SMTP was designed years ago for text transfer only**
  - **MIME software is included as part of an e-mail client**
  - **Superimposes a format for the body text, so a graphic can be represented using text, and then sent via SMTP (as a special attachment)**
  - **Receiver's e-mail client then translates the MIME attachment from text back into graphical format**

Telnet/SSH

- **Allows one computer to log into another computer**
  - **Remote login enabling full control of the host**
- **Requires account name and password**
  - **Anonymous sites similar to FTP approach**
- **Most popular Telnet software is PuTTY**
  - **Open source**
  - **Uses SSH encryption for security**
- **Remote Desktop (windows)**
  - **Most advanced, connecting Window-based machines, provide full access to Window interface**

Instant Messaging (IM)

- One of the fastest growing Internet applications
- Allows users to exchange real-time typed messages or chat with friends

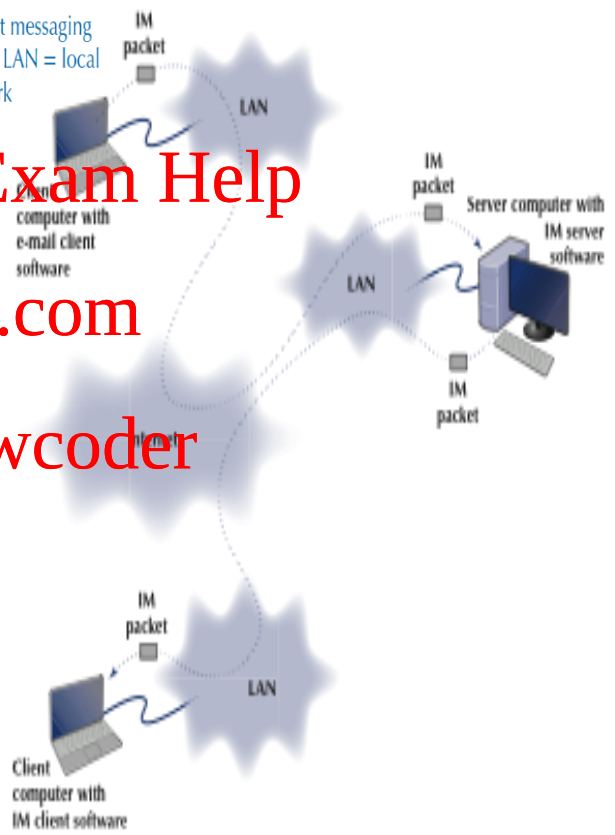


FIGURE 2-16
How instant messaging (IM) works. LAN = local area network

Videoconferencing

- Provides real-time transmission of video and audio signals
- Combined video/audio signals sent via WAN (Wide Area Network)
- Desktop videoconferencing is fast growing (Skype, FaceTime etc.,)
- Require a lot of network capacity thus use data compression
- Most often compatibility is an issue

Thank You