Chapter 6
Application Layer

- ***Outline:***

  6.1 Web: HTTP & HTTPS
  6.2 File Transfer: FTP, PuTTY, WinSCP
  6.3 Electronic Mail: SMTP, POP3, IMAP
  6.4 DNS
  6.5 P2P Applications
  6.6 Socket Programming
  6.7 Application server concept: proxy caching, Web/Mail/DNS server optimization
  6.8 Concept of traffic analyzer: MRTG, PRTG, SNMP, Packet tracer, Wireshark.

Application Layer

- An **application layer** is an abstraction layer that specifies the shared communications protocols and interface methods used by hosts in a communications network.
- ***Network Applications Layer Protocols***
  - **DNS:** Matches domain names with IP addresses
  - **WWW - HTTP:** Used to transfer data between clients/servers using a web browser.
  - **SMTP & POP3 :** used to send email messages from clients to servers over the internet.
  - **FTP** : allows the download/upload of files between a client/server.
  - **Telnet:** allows users to login to a host from a remote location and take control as if they were sitting at the machine (virtual connection).
  - **DHCP:** assigns IP addresses, subnet masks, default gateways, DNS servers, etc. To users as they login the network.
  - **Multimedia.**

HTTP

- HTTP stands for **Hyper Text Transport Protocol.** Web's application-layer protocol, is at the heart of the Web.
- HTTP is an application-level protocol **for distributed, collaborative, hypermedia information systems.**
- This is the foundation for data communication for the **World Wide Web** (i.e. internet) since 1990
- Allows the **transmitting and receiving** of information across the Internet.
- Use TCP **at Transport Layer**
- Web server listen **at port number 80**
- HTTP is implemented in two programs: **a client program and server program**.

HTTP Operation

- The HTTP protocol is a **request/response protocol.**
- **Client**: A client **sends a request** to the server in the form of
  - a request method
  - URI, and
  - protocol version
  - followed by a MIME-like message containing
    - request modifiers
    - client information, and
    - possible body content over a connection with a server.

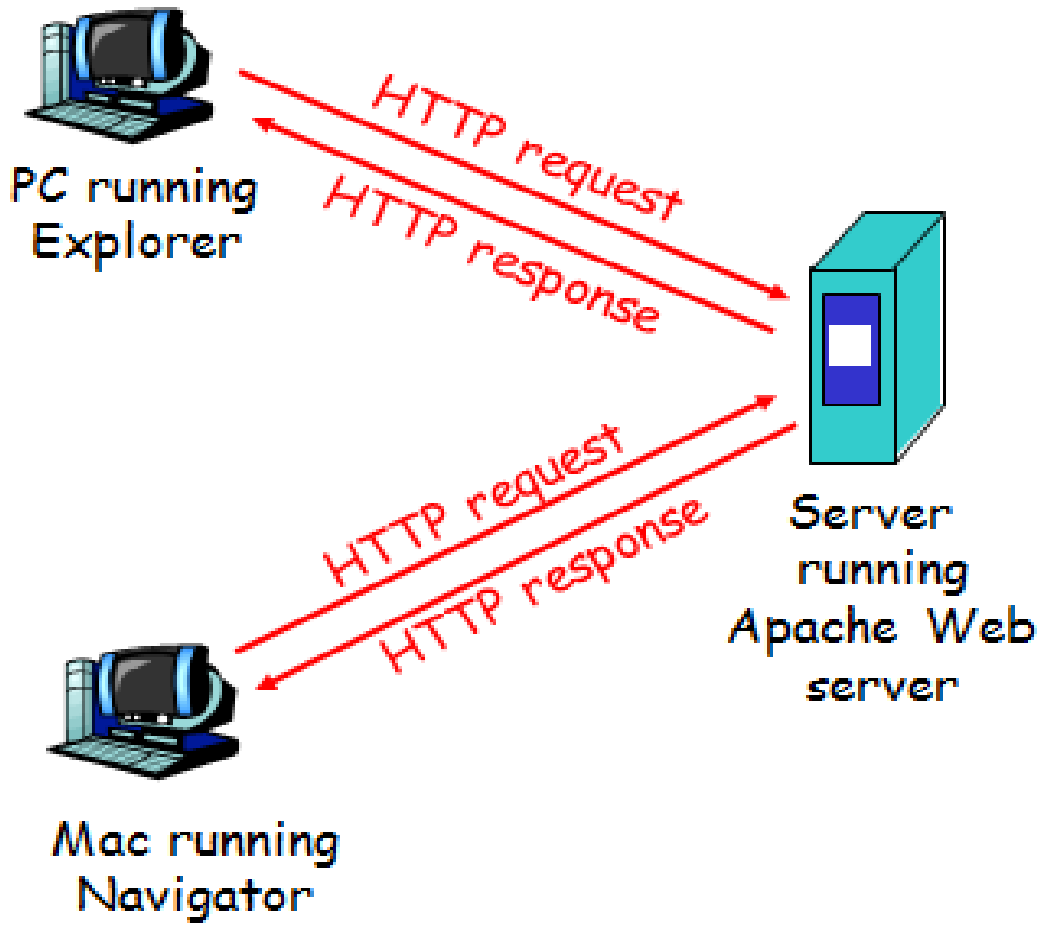
- ***MIME: Multipurpose Internet Mail Extensions***

HTTP Operation

- **Server:** The **server responds with**
  - a status line
  - the message's protocol version and
  - a success or error code,
  - followed by a MIME-like message containing
    - ✓ server information,
    - ✓ entity meta information, and
    - ✓ possible entity-body content

HTTP overview

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



HTTP Connection

Non-persistent HTTP

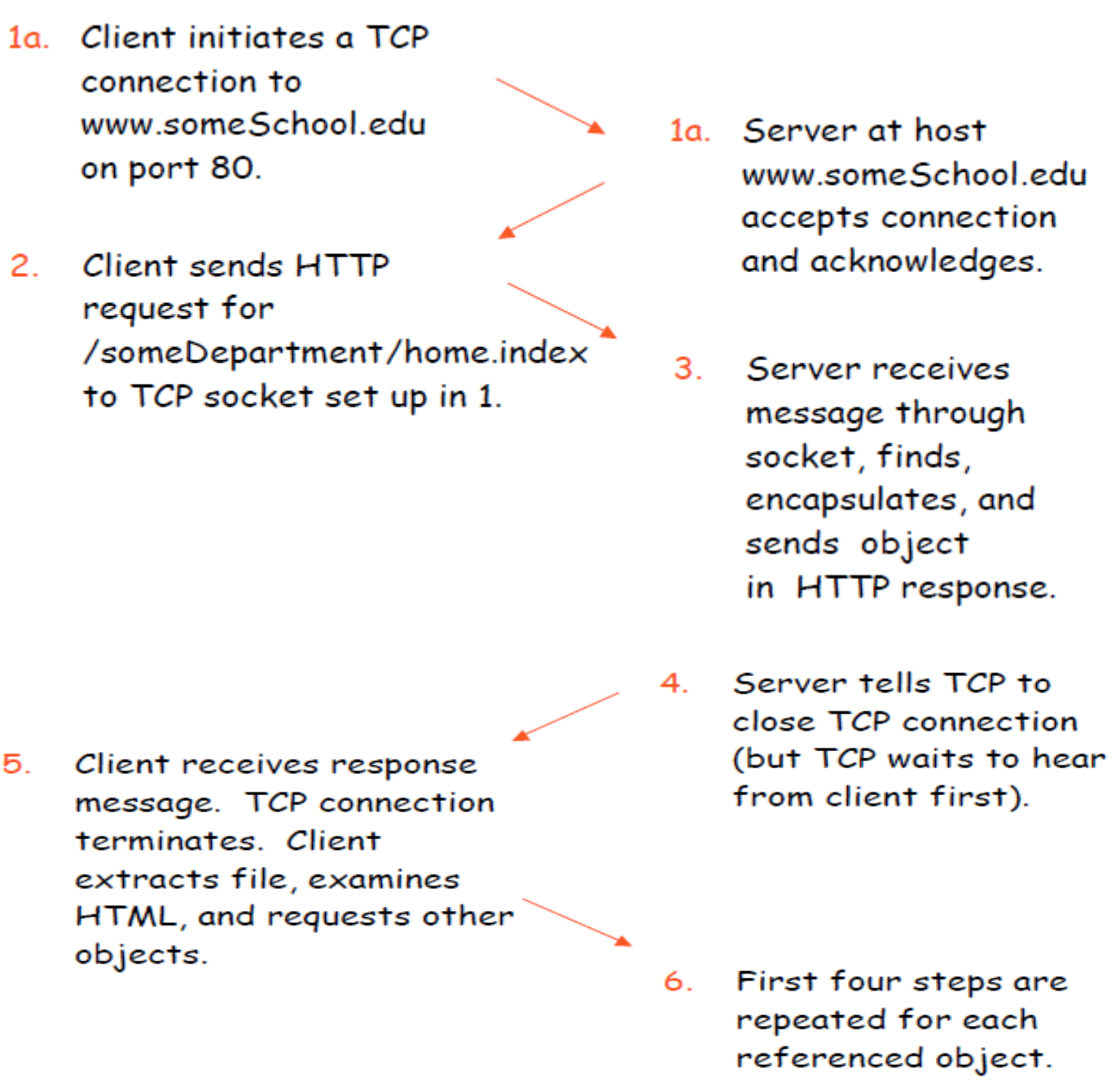
- HTTP/1.0 uses non-persistent HTTP
- In a **non-persistent connection**, one TCP connection is made for each request/response.

Persistent HTTP

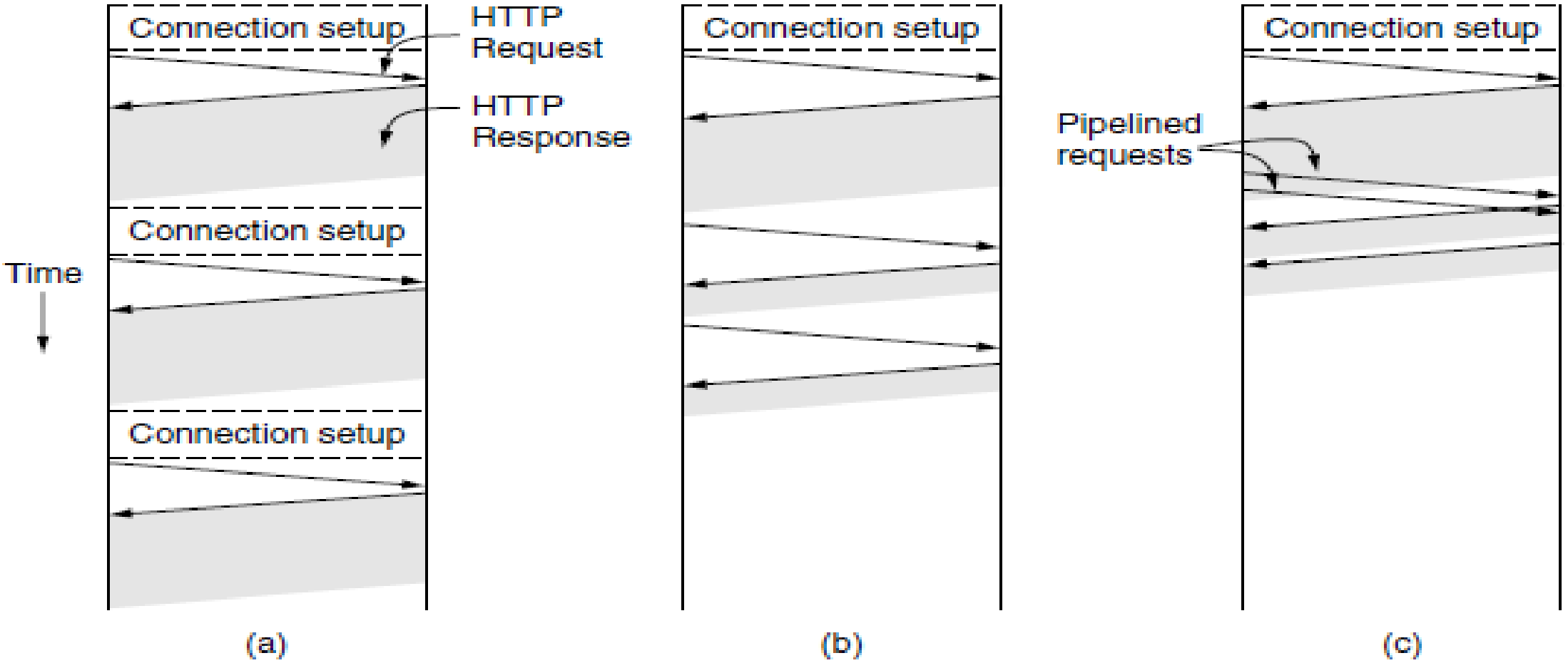
- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode
- In a persistent connection, the server leaves the connection open for more requests after sending a response.
- The server can close the connection at the request of a client or if a time-out has been reached.

HTTP/1.0 non-persistent connection

- www.someSchool.edu/someDepartment/home.index



HTTP Connection



HTTP with (a) multiple connections and sequential requests. (b) A persistent connection and sequential requests. (c) A persistent connection and pipelined requests.

HTTP Connection

- **Figure (a) shows** non-persistent connections
  - *Browser makes connection and request for each object*
  - *Server parses request, responds and closes connection*
  - *Performance problem*
- **Figure (b) shows** persistent connections
  - *On same TCP connection server parses request, responds, parses new request upon responds and so on*
  - *Improves performance*
- **Figure (c) shows** pipelining with persistent connections
  - *Send next request before previous response is received*
  - *Pipelining with persistent connection improves performance*

Response time modeling

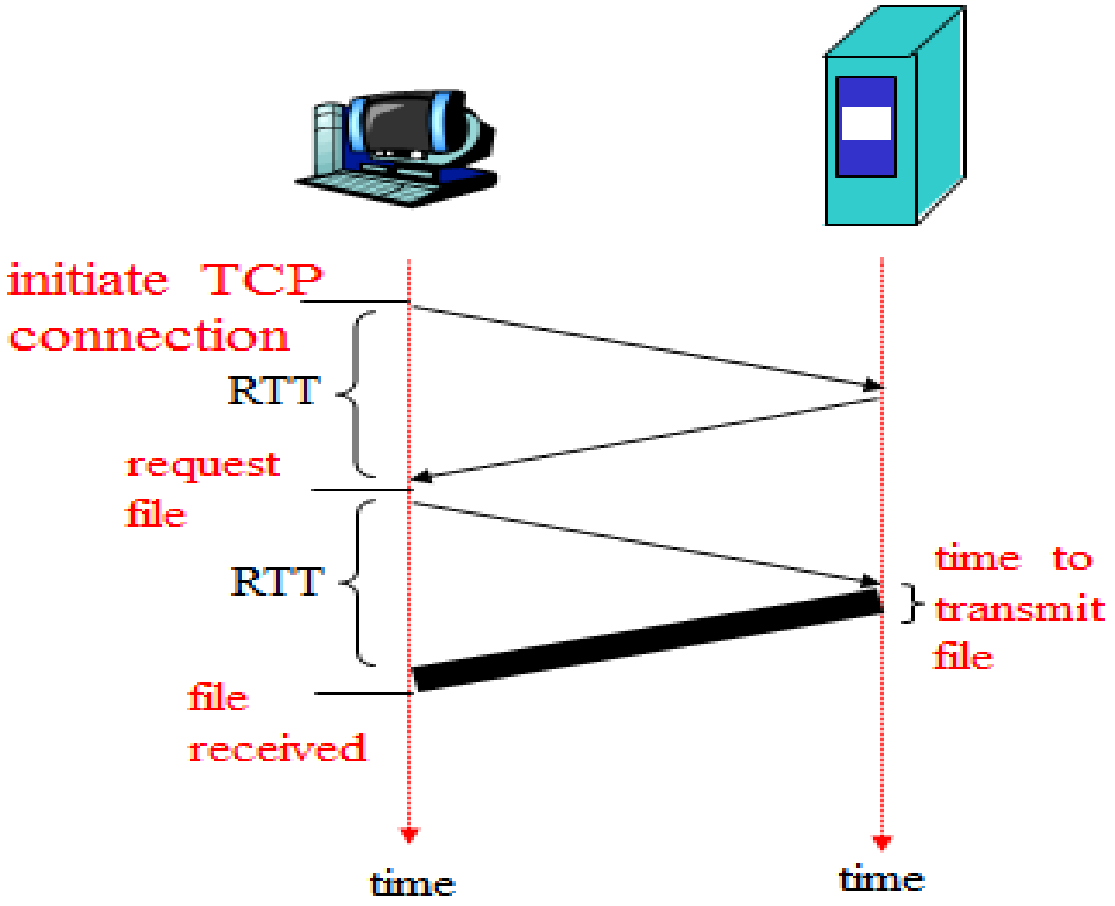
Definition of RRT: time to send a small packet to travel from client to server and back.

Q: Do you remember what contribute to the RTT?

Response time (assumed no packet losses):

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

total = 2RTT+transmit time



HTTP request message

HTTP request message: ASCII (human-readable format)

request line (GET, POST, HEAD commands)

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

header lines

Carriage return, line feed indicates end of message

(extra carriage return, line feed)

HTTP response message

status line (protocol status code status phrase)

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html
```

header lines

data, e.g., requested HTML file

```
data data data data data ...
```

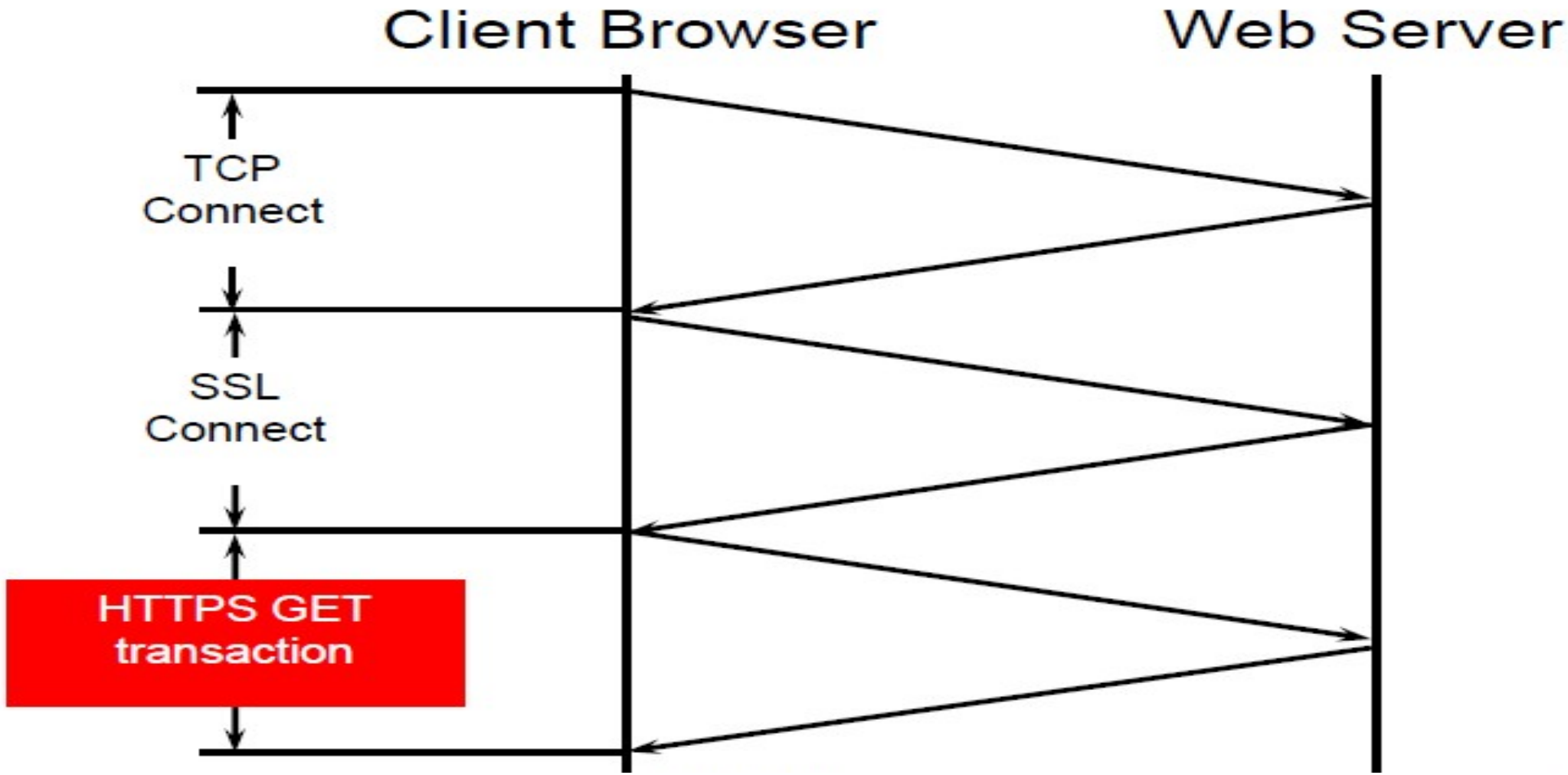
HTTPS

- Use **TCP at Transport layer**
- Listen at **port number 443**
- **Hypertext Transfer Protocol Secure (HTTPS)** is a widely-used communications protocol for secure communication over a computer network, with especially wide deployment on the Internet.
- HTTPS is http using a **Secure Socket Layer (SSL).**
- Technically, it is not a protocol in itself; rather, it is the result of simply layering the Hypertext Transfer Protocol (HTTP) on top of the SSL/TLS protocol, thus adding the security capabilities of **SSL/TLS** to standard HTTP communications.
- A secure socket layer is an **encryption protocol invoked on a Web server that uses HTTPS.**

HTTPS

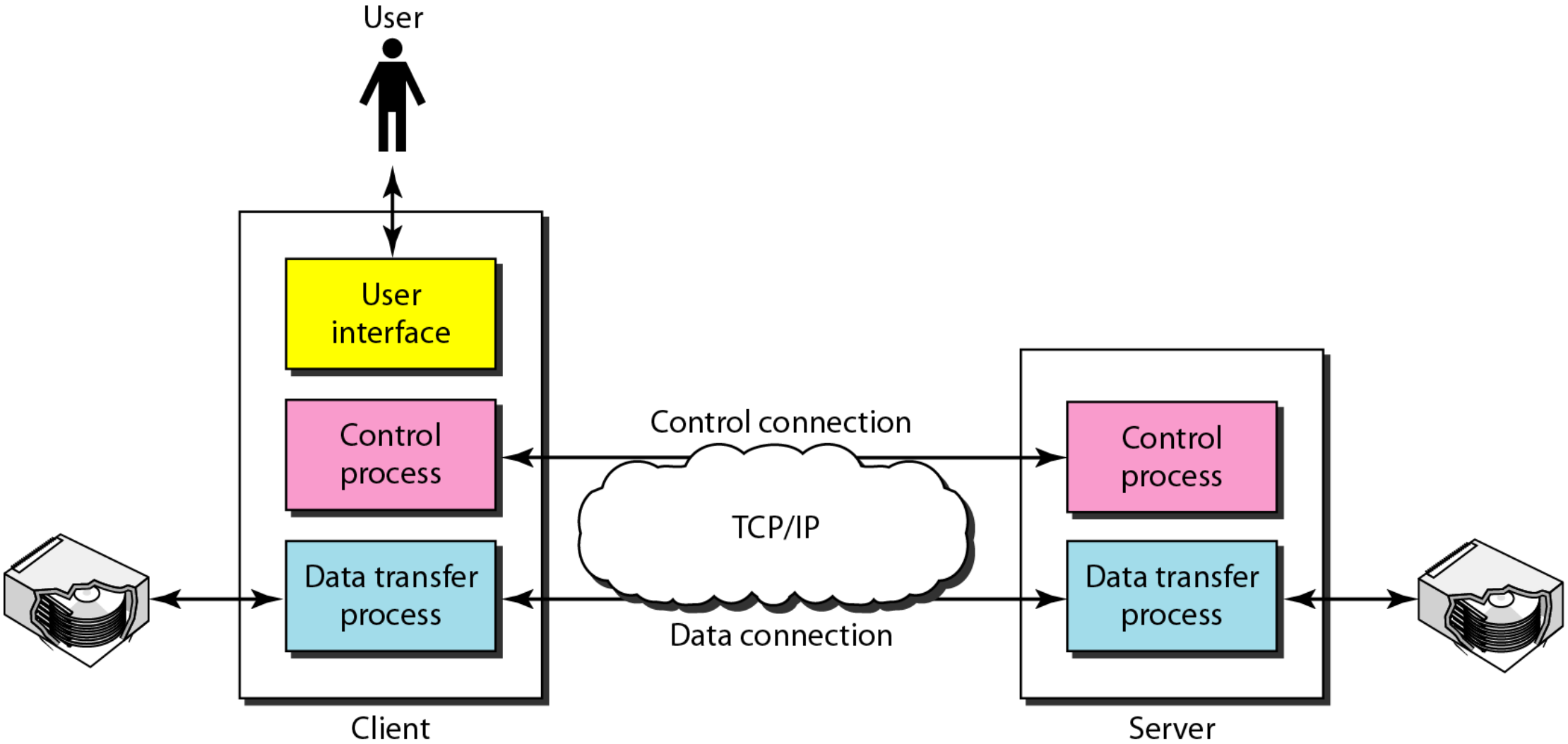
- HTTPS is one of the popular protocols to transfer sensitive data over the Internet.
- Most implementations of the HTTPS protocol involve **online purchasing or the exchange of private information.**
- HTTPS is only **slightly slower than HTTP**
- Why is **HTTPS not used for all web traffic**?
  - *Slows down web servers*
  - *Breaks Internet caching*
    - ISPs cannot cache HTTPS traffic
    - Results in increased traffic at web site

HTTPS Transaction



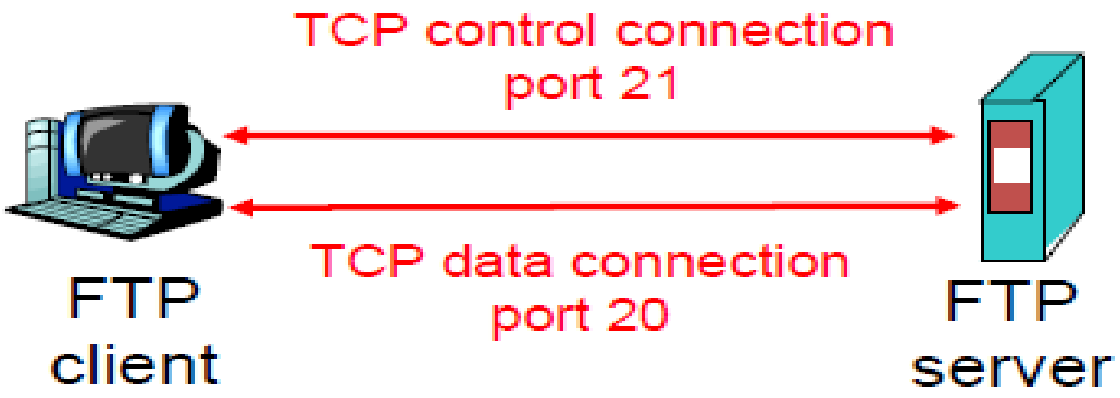
File Transfer Protocol (FTP)

- Allows a user to **copy files to/from remote hosts.**
- FTP uses the **services of TCP.**
- It needs **two TCP connections .**
  - The well known **port 21** is used for the **control connection**
  - and well known **port 20** for the **data connection**
- **Goal of FTP**
  - promote sharing of files
  - encourage indirect use of remote computers
  - shield user from variations in file storage
  - transfer data reliably and efficiently


User
User interface
Control process
Data transfer process
Client
Control connection
TCP/IP
Data connection
Control process
Data transfer process
Server

FTP: Working

- FTP client contacts FTP server at port 21, specifying TCP as transport protocol
- Client obtains authorization over control connection
- Client browses remote directory by sending commands over control connection.
- When server receives a command for a file transfer, the server opens a TCP data connection to client
- After transferring one file, server closes connection.



- Server opens a second TCP data connection to transfer another file.
- Control connection: "out of band"
- FTP server maintains "state": current directory, earlier authentication

FTP Replies

- All replies are sent over control connection.
- Replies are a single line containing
  - 3 digit status code (sent as 3 numeric chars).
  - text message.
- FTP Reply Status Code
- First digit of status code indicates type of reply:

  1: Positive Preliminary Reply (got it, but wait).

  2: Positive Completion Reply (success).

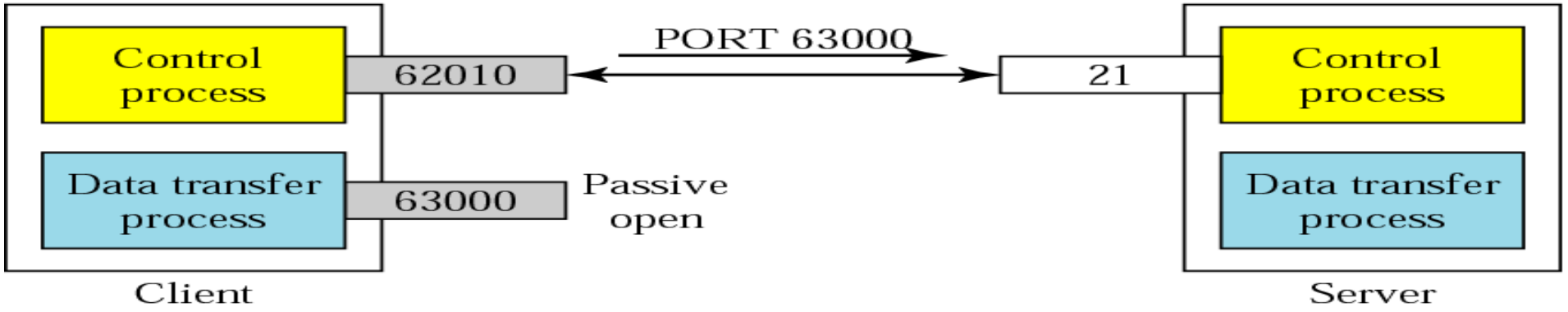
  3: Positive Intermediate Reply (waiting for more information).

  4: Transient Negative Completion (error - try again).

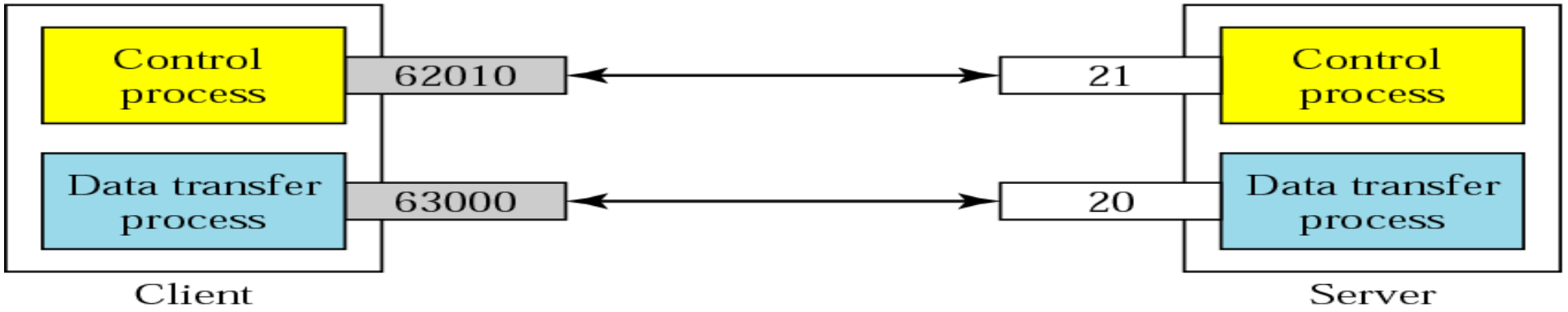
  5: Permanent Negative Reply (error - can’t do)
- 2nd digit indicates function groupings.
- 3rd digit indicates specific problem within function group.

FTP : The data connection

a. Passive open by client



b. Sending ephemeral port number to server



c. Active open by server

FTP: The Data Connection

- **Uses Server's well-known port 20**
  - Client issues a passive open on an ephemeral port, say *x*.
  - Client uses PORT command to tell the server about the port number *x*.
  - Server issues an active open from port 20 to port *x*.
  - Server creates a child server/ephemeral port number to serve the client
- **Independent of data type and file structure**
  - **Stream S:** file is transmitted as stream of bytes
  - **Block B:** file sent as sequence of blocks preceded by header information ; allows restart of an interrupted transfer
  - **Compressed C:** data compressed using simple compression techniques e.g., run length encoding

Trivial File Transfer Protocol (TFTP)

- **UDP port 69**
- Simple protocol, usually used to **transfer configuration files**
- Usually used for transferring boot file for **diskless hosts (X-Stations) or updating NVRAM**
- Typically used in **short distance, low noise environments**
- Server is usually implement in **firmware for updating things like routers, bios**.
- **Because of its compact size:**
  - no error recovery like TCP based FTP
  - no command structure like FTP
  - cannot list directories
  - transfers to server are to a single configured directory.

Putty, WinSCP

- WinSCP (Windows Secure Copy) is a free and open source SFTP, SCP, and FTP client for Microsoft Windows.
- Its main function is secure file transfer between a local and a remote computer. Beyond this, WinSCP offers basic file manager and file synchronization functionality.
- For secure transfers, it uses Secure Shell (SSH) and supports the SCP protocol in addition to SFTP
- WinSCP is based on the implementation of the SSH protocol from PuTTY and FTP protocol from FileZilla.
- For downloading and other information refer to:
  - http://winscp.net http://www.putty.org/ https://filezilla-project.org/

Electronic Mail

- It is one of the most widely used and popular internet applications.
- User can communicate with each other across the network
- Every user owns his own mailbox which he uses to send, receive and store messages from other users
  - Every user can be uniquely identified by his unique email address
- ***Mail Send Protocol***
  - **Simple Mail Transfer Protocol(SMTP)**
- ***Mail Access Protocol:*** retrieval from server
  - **Post Office Protocol v3 (POP3)**
  - **Internet Mail Access Protocol (IMAP)**

The Architecture of email System

- It consists of two kinds of subsystems
  - User agents
  - Message transfer agents

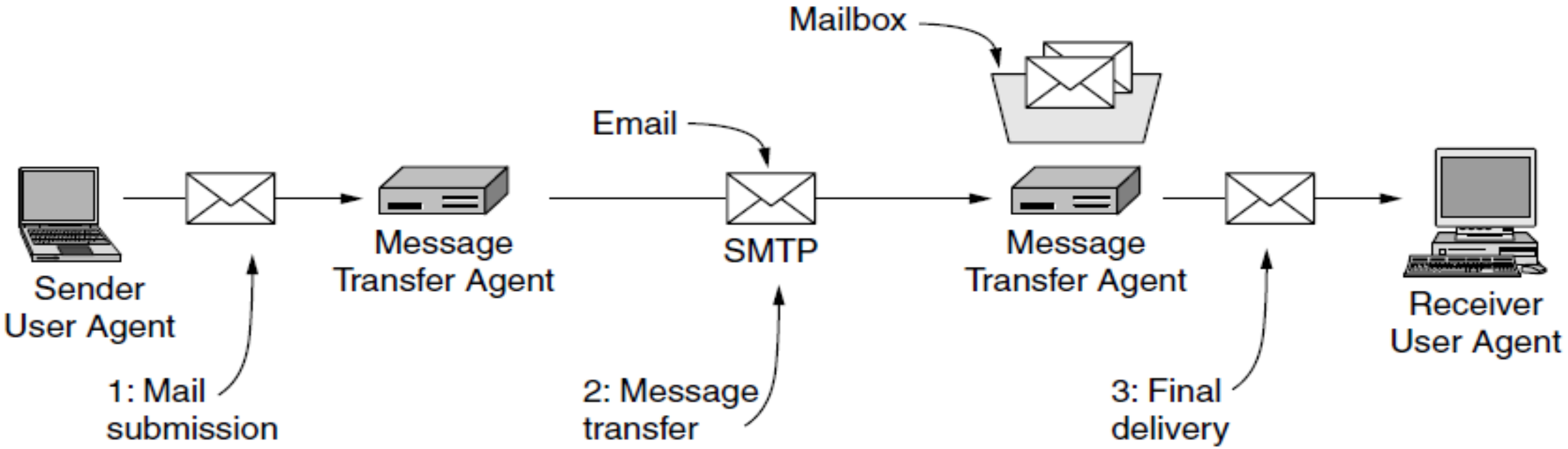


Figure 7-7. Architecture of the email system.

The Architecture of email System

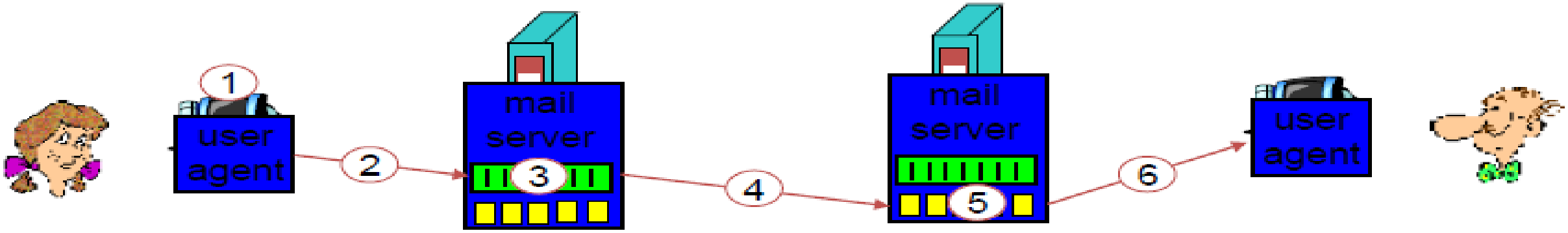
- **User Agent**
  - Which allow people **to read and send email.**
  - A user agent is a program (sometimes called an **email reader**).
  - It includes a means to **compose messages** and **replies to messages**, **display** incoming messages, and **organize messages by filing**, **searching, and discarding them.**
  - The **act of sending new messages** into the mail system for delivery is called **mail submission.**
  - There are many popular user agents, including **Google gmail, Microsoft Outlook, Mozilla Thunderbird, and Apple Mail.**
  - They can vary greatly in their appearance. Most user agents have a **menu- or icon driven graphical interface.**

The Architecture of email System

- **Message transfer agents**
  - **Move the messages** from the **source to the destination.**
  - Also refer **message transfer agents** informally as **mail servers**.
  - They run in the **background on mail server machines** and are intended to be **always available.**
  - Their job is to automatically move email through the system from the originator to the recipient with **SMTP** (**Simple Mail Transfer Protocol**). It sends mail over connections and **reports back the delivery status and any errors**.
  - Message transfer agents also implement **mailing lists**, in which an identical **copy of a message is delivered to everyone** on a list of email addresses.
  - **Mailboxes** store the email that is received for a user. They are maintained by mail servers.

Scenario: Alice sends message to Bob

1. Alice uses UA to compose message and send “to” bob@someschool.edu
2. Alice’s UA 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

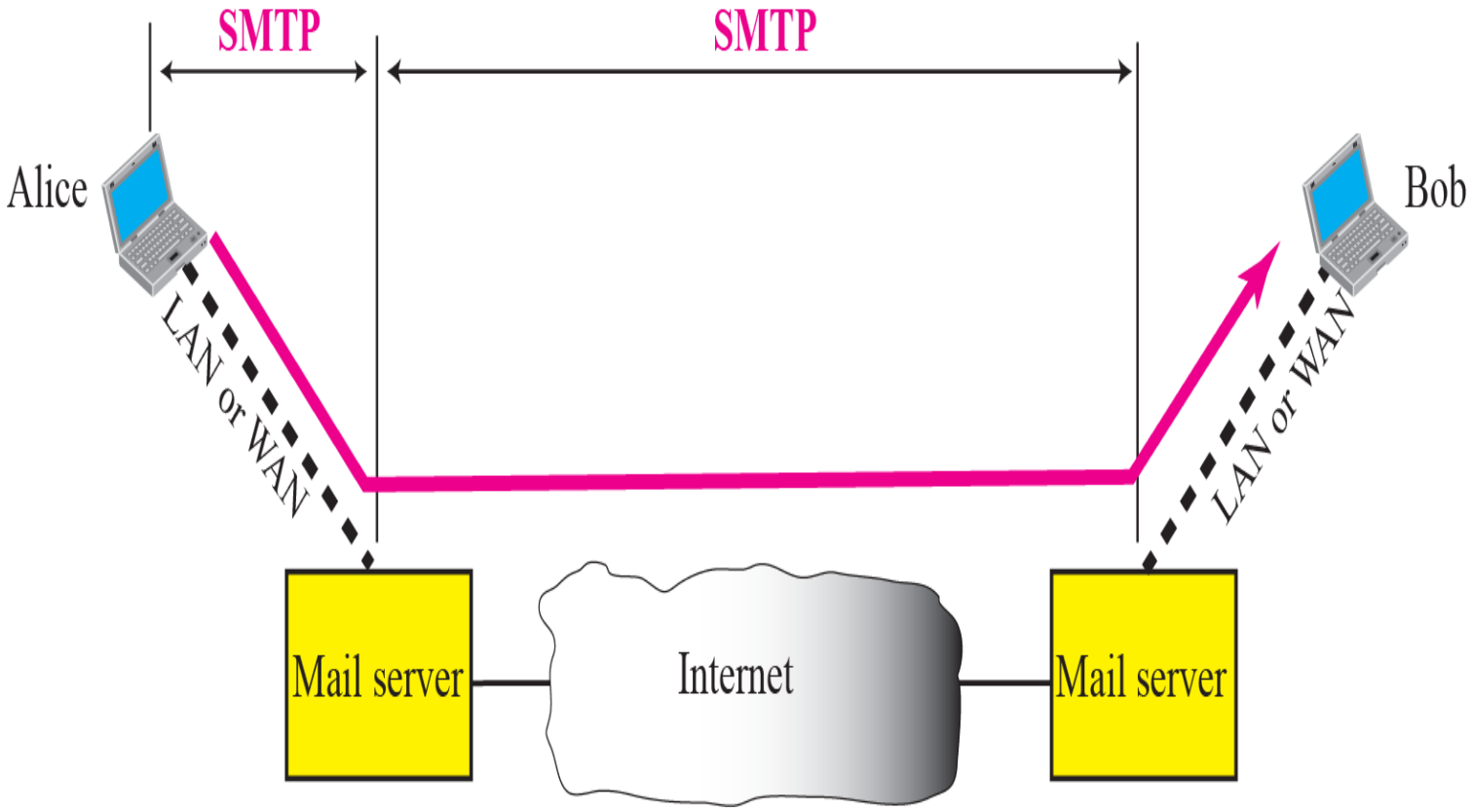


SMTP

- **Simple Mail Transfer Protocol** (**SMTP**) is an Internet standard for electronic mail (email) transmission.
- First defined by **RFC 821 in 1982**, it was last updated in 2008 with Extended SMTP **additions by RFC 5321**, which is the protocol in widespread use today.
- User-level client mail applications typically **use SMTP only for sending messages** to a mail server for relaying.
- SMTP communication between mail servers **uses TCP port 25.**
- SMTP connections secured by T**ransport Layer Security** (**TLS**). **Secure Sockets Layer** (**SSL**) is the predecessor of TLS.

SMTP

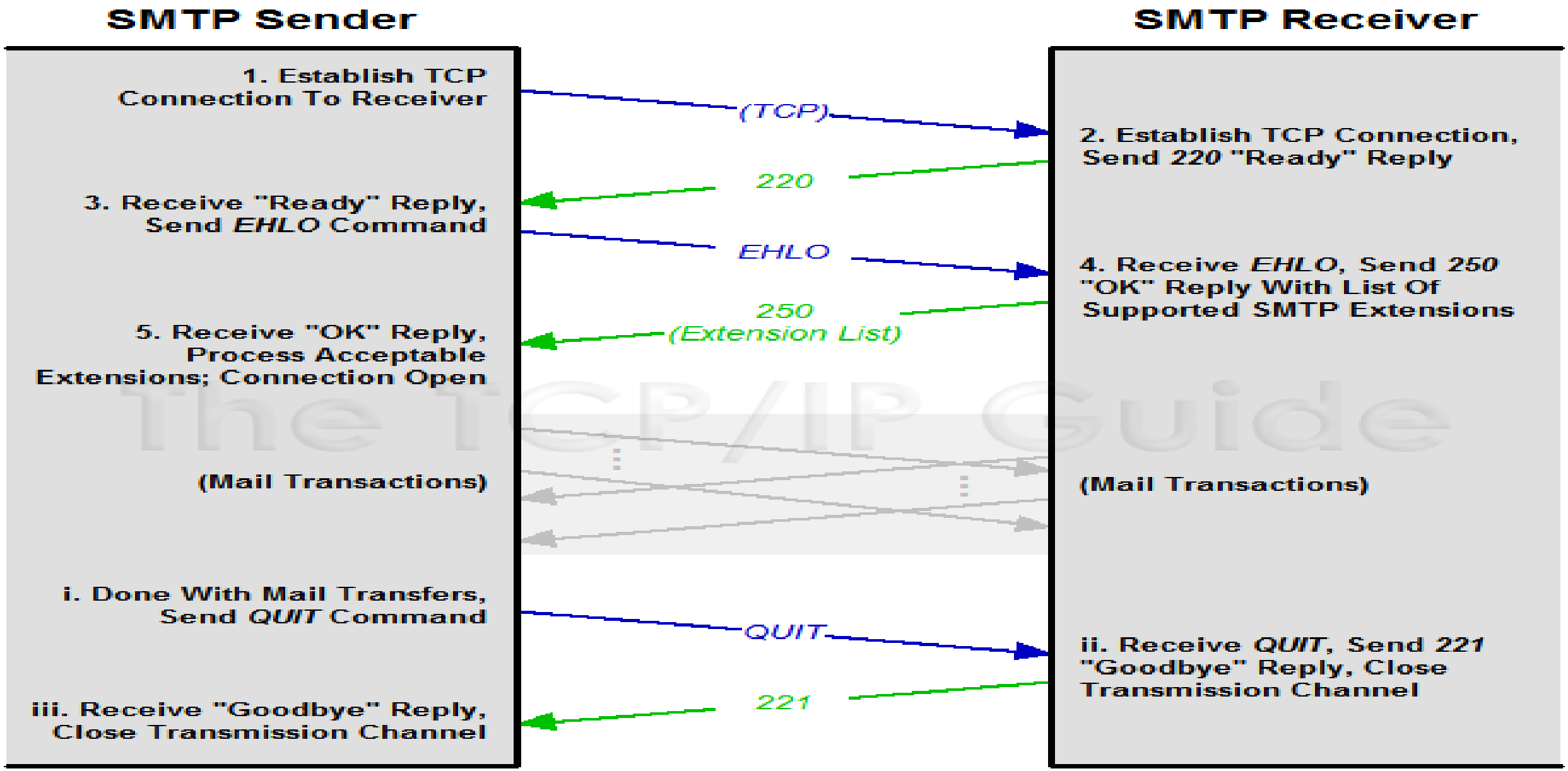
- **Three phases of transfer**
  - *Handshaking (greeting)*
  - *Transfer of messages*
  - *Closure*
- **Command/response interaction**
  - *Commands: ASCII text*
  - *Response: status code and phrase*
  - *Messages must be in 7-bit ASCII*



Commands of SMTP

- **HELO** : Request to initiate SMTP session
- **MAIL FROM** : Sender’s E-Mail address
- **RCPT TO** : Receiver’s E-Mail address
- **DATA** : Body of message
- **QUIT** : Terminates SMTP connection
- **RSET** : Aborts mail transaction
- **VRFY** : Asks receiver to verify the validity of the mailbox
- **EXPN** : Asks receiver to identify mailing list
- **HELP** : Causes receiver to send help information
- **NOOP** : Forces server to verify the communication with SMTP receiver

Session Establishment and Termination

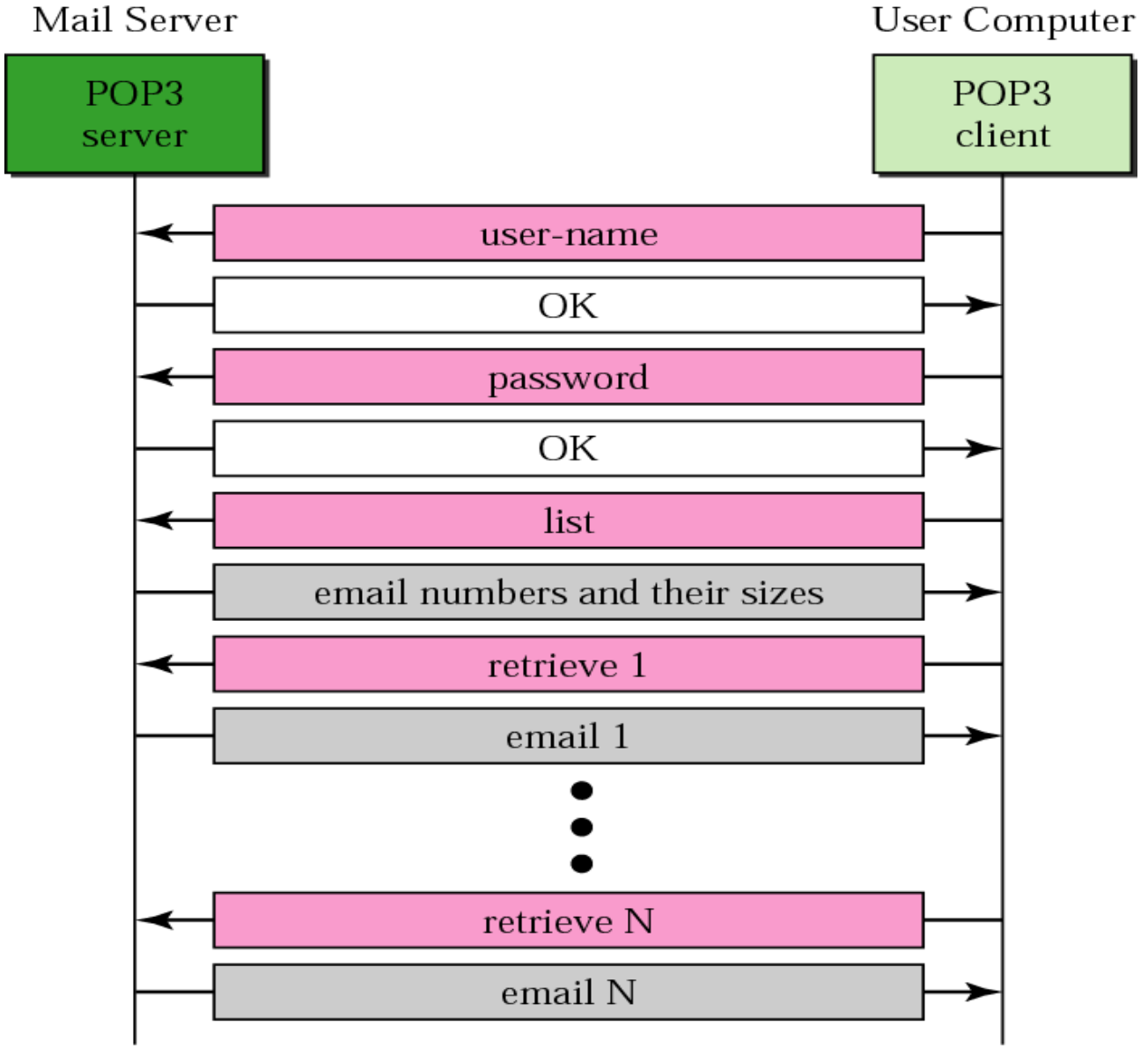


POP3

- Post Office Protocol version 3 (POP3) is a standard mail protocol used to **receive emails** from a remote server to a local email client.
- POP3 allows you to download email messages on your local computer and read them even when you are offline.
- POP has largely been superseded by the Internet Message Access Protocol (IMAP).
- By default, the POP3 protocol works on two ports:
  - **Port 110** - this is the default POP3 non-encrypted port
  - **Port 995** - this is the port you need to use if you want to connect using POP3 securely
- The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.

POP3

- POP3 has two modes: **the delete mode and the keep mode.**
- **In the delete mode**, the mail is deleted from the mailbox after each retrieval.
- **In the keep mode**, the mail remains in the mailbox after retrieval.
- The **delete mode** is normally used when the user is working **at her permanent computer.**
- The **keep mode** is normally used when the user accesses her mail **away from her primary computer.**



Internet Mail Access Protocol, version 4 (IMAP4).

- Version 4 of the protocol is defined in **RFC 3501.**
- IMAP uses **TCP/IP port 143.**
- **IMAP4 provides the following extra functions:**
  - A user can check the e-mail header prior to downloading.
  - A user can search the contents of the e-mail for a specific string of characters prior to downloading.
  - A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
  - A user can create, delete, or rename mailboxes on the mail server.
  - A user can create a hierarchy of mailboxes in a folder for e-mail storage.

POP3 Vs. IMAP4

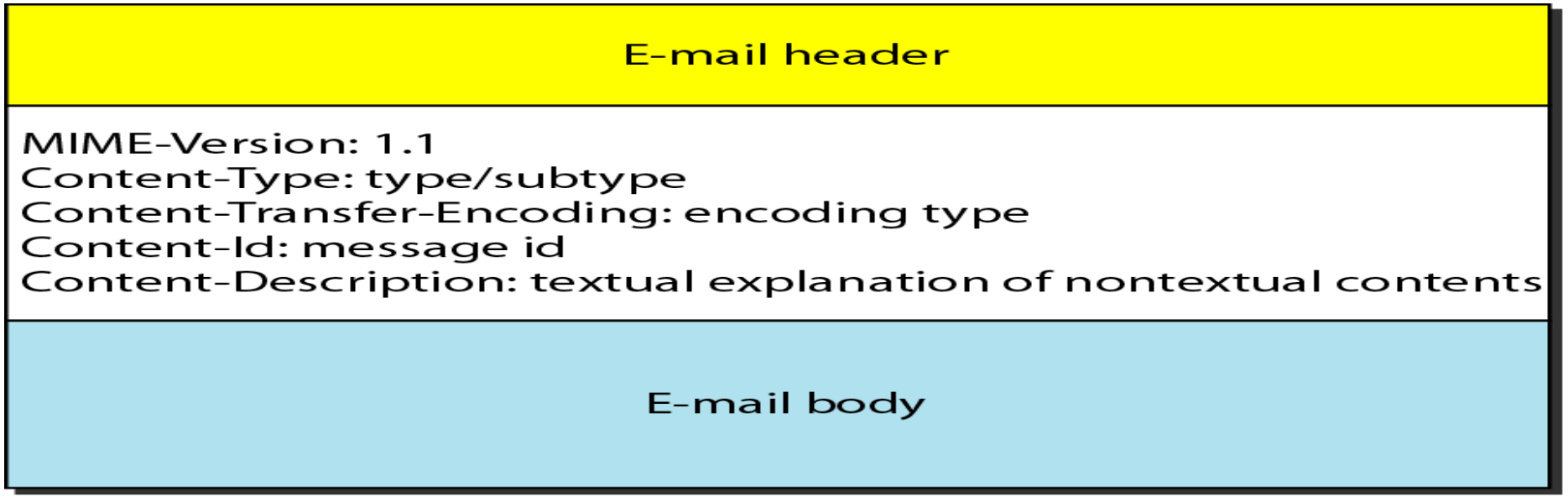
Feature	POP3	IMAP
Where is protocol defined?	RFC 1939	RFC 2060
Which TCP port is used?	110	143
Where is e-mail stored?	User's PC	Server
Where is e-mail read?	Off-line	On-line
Connect time required?	Little	Much
Use of server resources?	Minimal	Extensive
Multiple mailboxes?	No	Yes
Who backs up mailboxes?	User	ISP
Good for mobile users?	No	Yes
User control over downloading?	Little	Great
Partial message downloads?	No	Yes
Are disk quotas a problem?	No	Could be in time
Simple to implement?	Yes	No
Widespread support?	Yes	Growing

MIME

- **MIME—The Multipurpose Internet Mail Extensions**
- **Electronic mail** has a simple structure. It can send messages only in **NVT 7-bit ASCII forma**t.
- *For example, it **cannot be used** for languages that are **not supported by 7-bit ASCII characters** (such as French, German, Hebrew, Russian, Chinese, and Japanese).*
- *Also, it cannot be used **to send binary files or video or audio data**.*
- Multipurpose Internet Mail Extensions are defined **for transmission of non-ASCII data.** It allows ordinary data **to be encoded in ASCII and transmitted in standard email.**
- MIME is a mechanism for specifying and describing the format of message **bodies in a standardized way**.

MIME

- MIME defines **five new message headers:**



Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

- ***MIME allows seven different types of data***

Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to mixed subtypes, but the default is message/ RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	IPEG	Image is in IPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single-channel encoding of voice at 8 kHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (8-bit bytes)

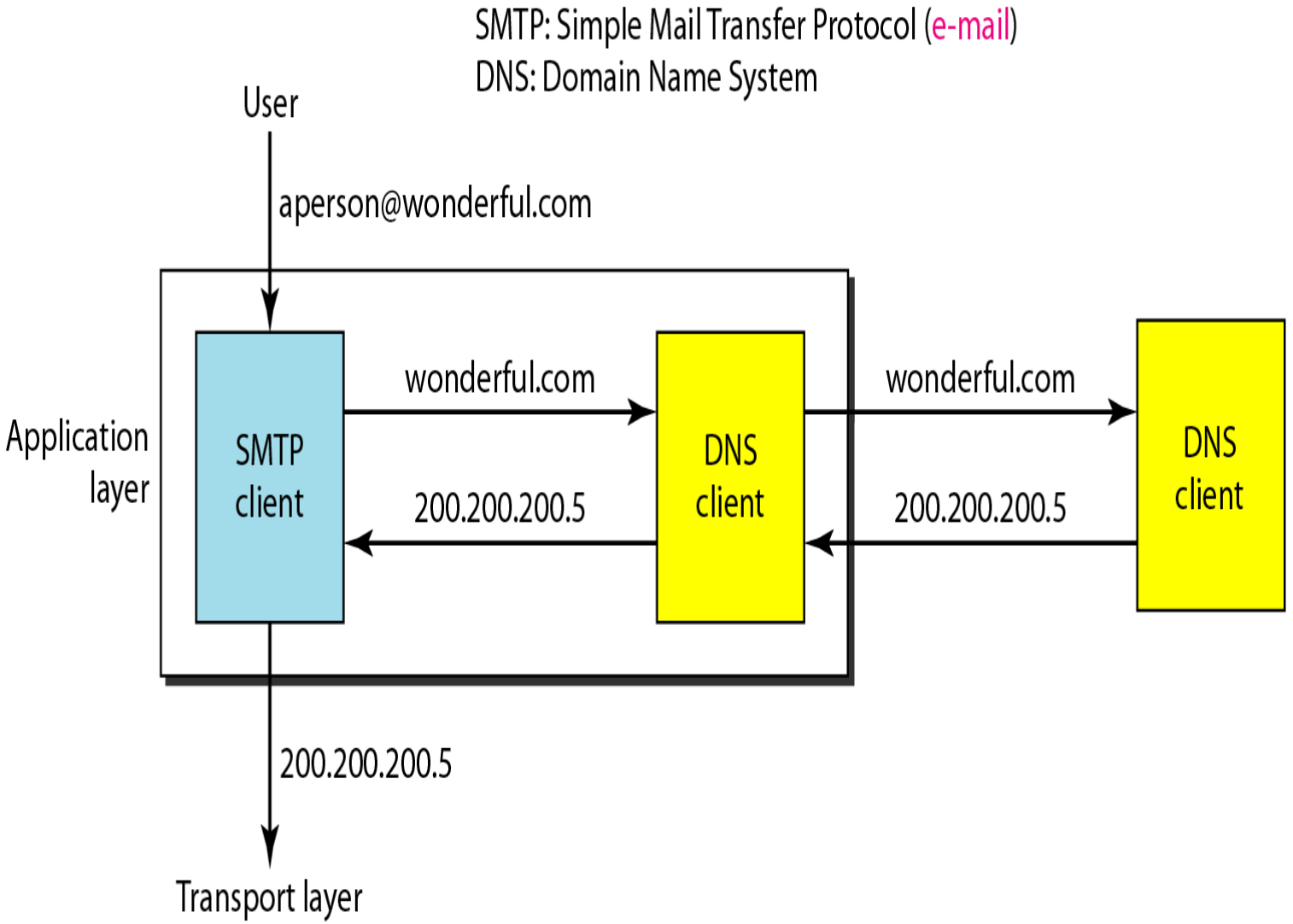
DNS (Domain Name System)

- DNS is usually used to **translate a host name into an IP address**
- **Automatically converts** the names we type in our Web browser address bar to the IP addresses of Web servers hosting those sites.
- Domain names **comprise a hierarchy** so that names are unique, yet easy to remember.
- DNS implements a **distributed database to store this name and address** information for all public hosts on the Internet.
- Most network operating systems **support configuration of primary, secondary, and tertiary DNS servers**, each of which can service initial requests from clients.
- Uses **UDP port number 53**
- The essence of DNS is the invention of a **hierarchical,** domain-based naming scheme and a **distributed database system** for implementing this naming scheme.

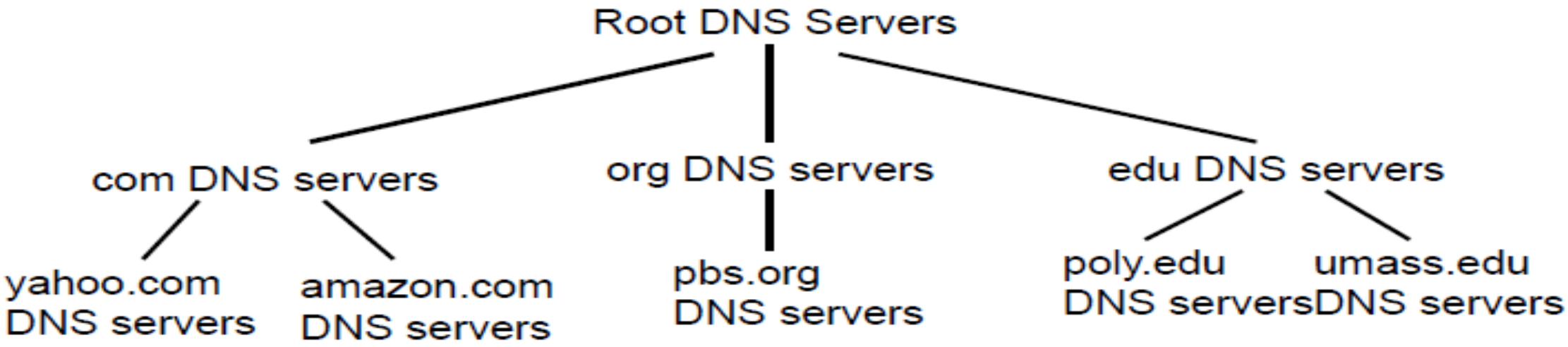
DNS (Domain Name System)

- **Which applications use DNS?**
  - **HTTP**
    - Browser extracts hostname
    - Sends hostname to DNS
    - DNS does lookup and returns IP address
    - Browser sends HTTP GET to IP address
- **Why not centralize DNS?**
  - single point of failure
  - traffic volume
  - distant centralized database
  - maintenance
  - doesn't *scale!*

Example of using the DNS service



Distributed, Hierarchical Database



- **Client wants IP for www.amazon.com:**
  - Client queries a root server to find com DNS server
  - Client queries com DNS server to get amazon.com DNS server
  - Client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: Root name server

- A **root name server** is a name server for the **root zone of the Domain Name System** of the Internet. It **directly answers requests for records** in the root zone and answers other requests **by returning a list of the authoritative name servers** for the appropriate top-level domain (TLD).
- The root name servers are a critical part of the Internet infrastructure because they are the first step in translating (resolving) human readable host names into IP addresses that are used in communication between Internet hosts.
- **Top-level domain (TLD) servers:**
  - **Generic top level domains:** responsible for .com, .org, .net, .edu, .gov, etc
  - **Countries have a 2 letter top level domain:** Example., uk, fr, np, ca, jp
- **Authoritative DNS servers:**
  - **Organization's DNS servers:** providing authoritative hostname to IP mappings for organization's servers (e.g., Web and mail)

Domain Name Space

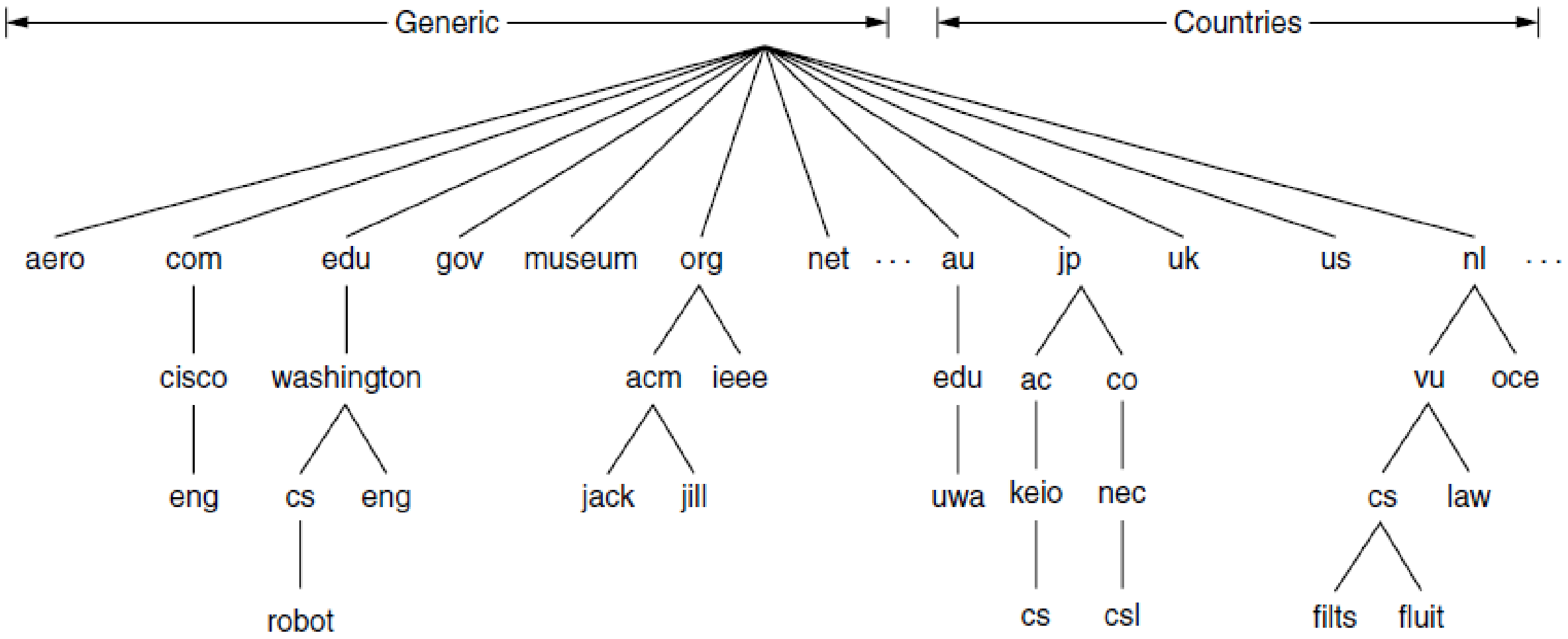
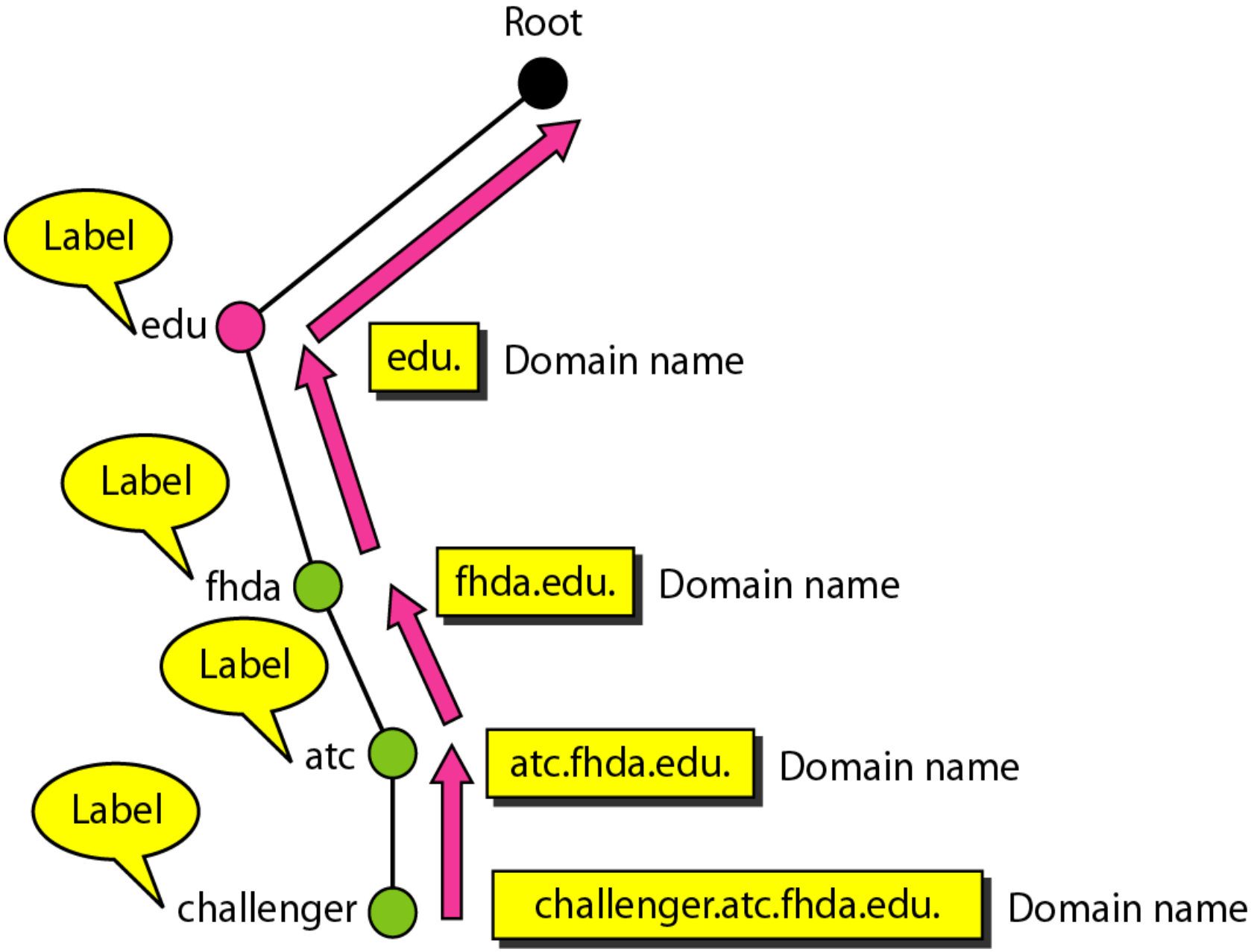


Figure 7-1. A portion of the Internet domain name space.

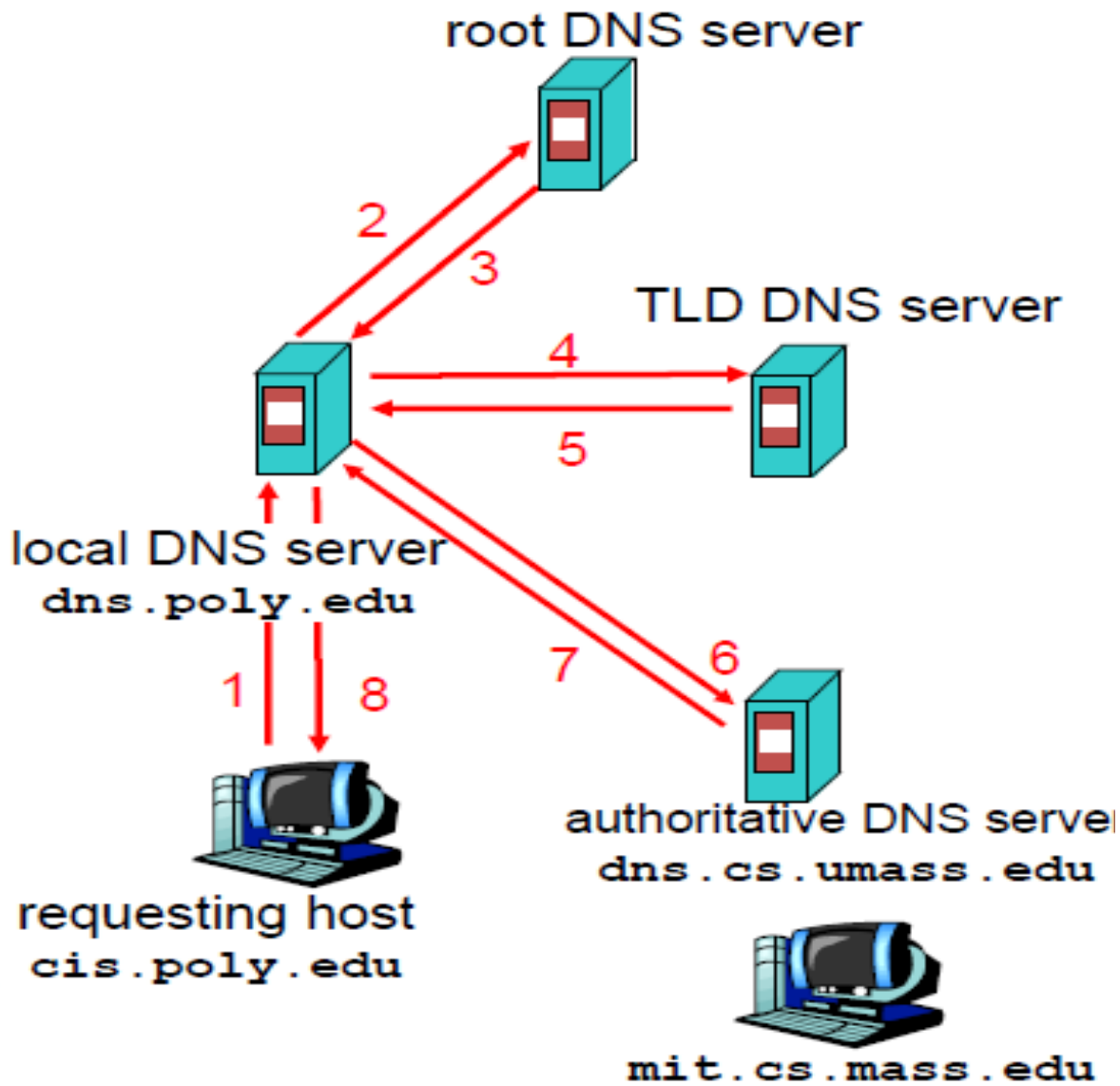
Domain names and labels



Example

- Host at cis.poly.edu wants IP address for mit.cs.mass.edu

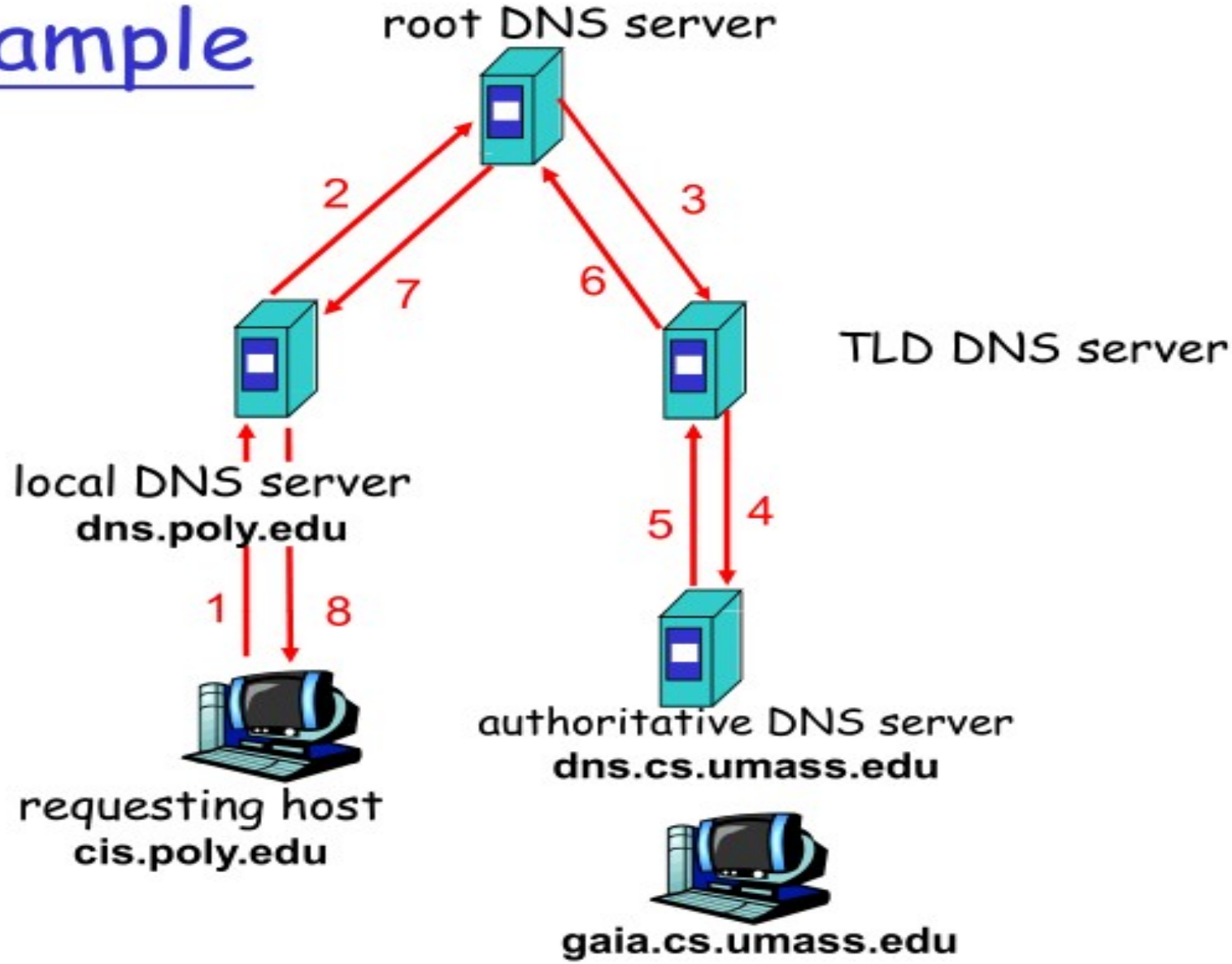
1. shows the query that is sent to the local name server.
2. start at the top of the name hierarchy by asking one of the root name servers.
3. name server for the *edu domain TLD*
4. sends the entire query to the *edu name server (mit.cs.mass.edu)*
5. *Returns the name server for authoritative DNS server*
6,7 . *authoritative DNS server resolves the query to local DNS server*
8. Host at cis.poly.edu gets IP address for mit.cs.mass.edu



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?



DNS records

DNS: distributed db storing resource records (RR)

RR format: `(name, value, type, ttl)`

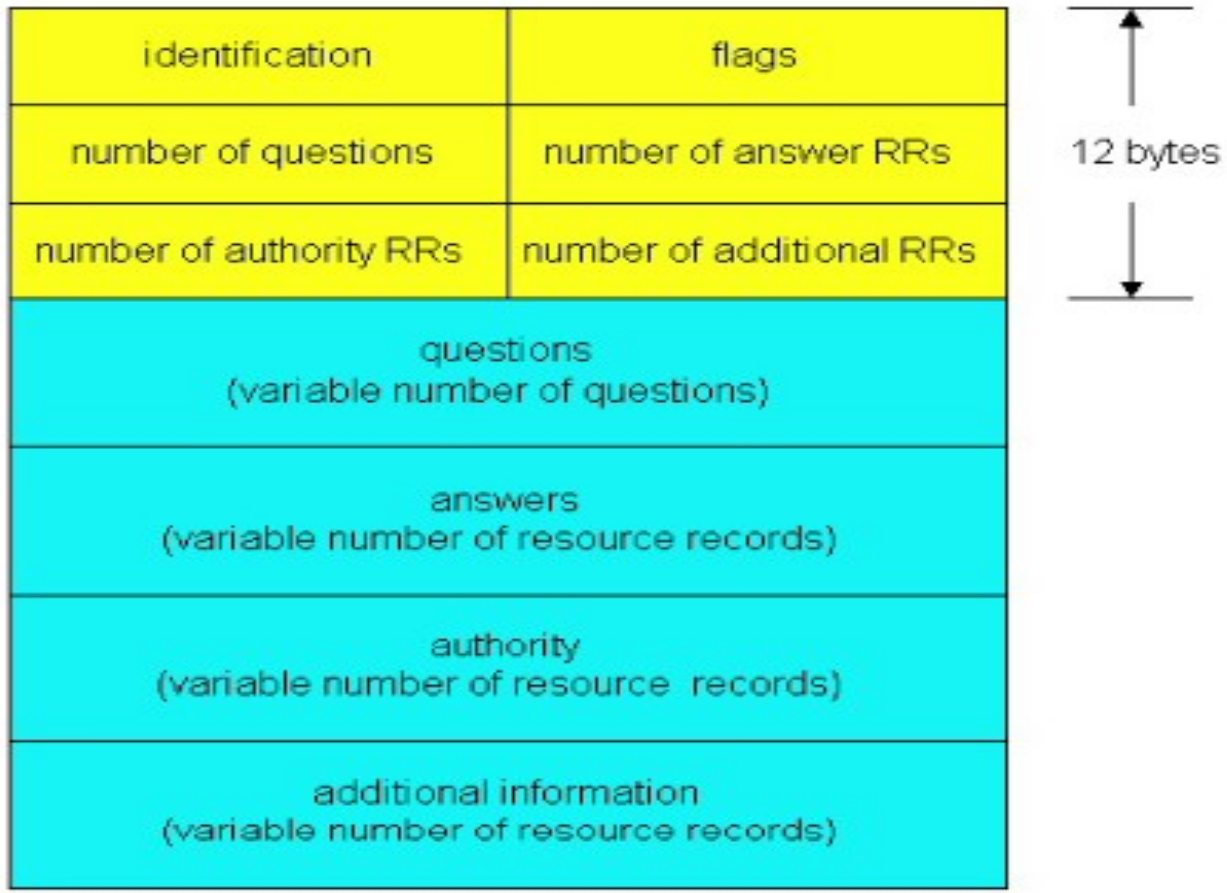
- Type=*A*
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
- Type=CNAME
  - **name** is alias name for some "canonical" (the real) name
    `www.ibm.com` is really `servereast.backup2.ibm.com`
  - **value** is canonical name
- Type=MX
  - **value** is name of mailserver associated with **name**

DNS protocol, messages

DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



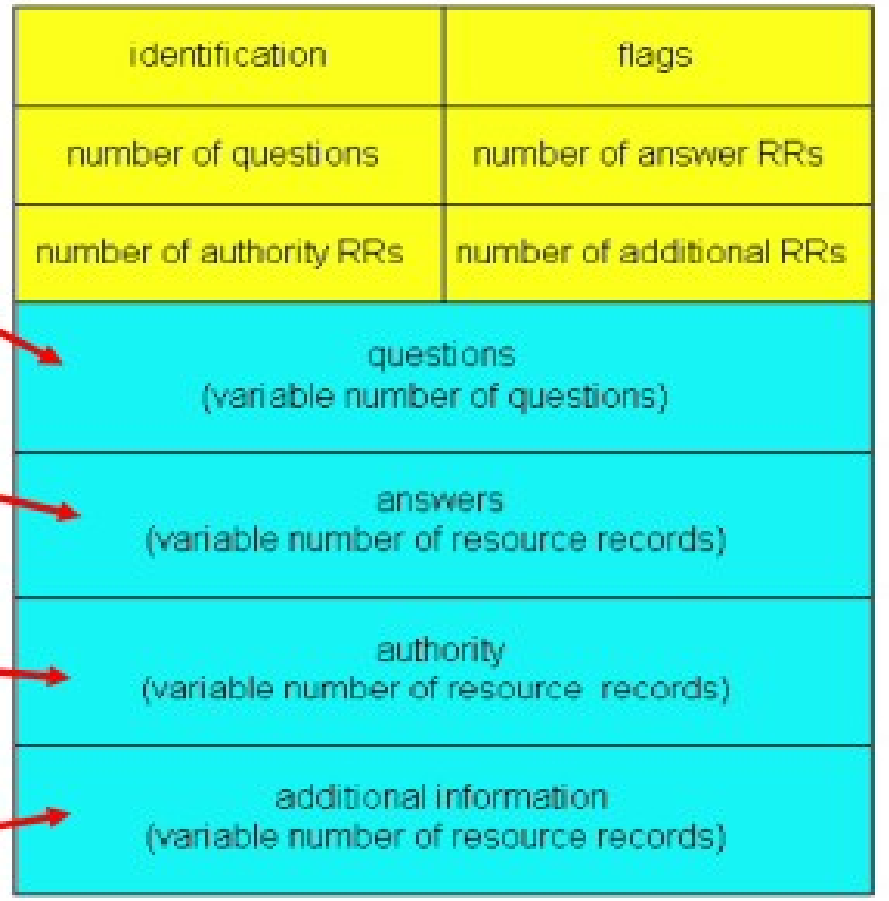
DNS protocol, messages

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used

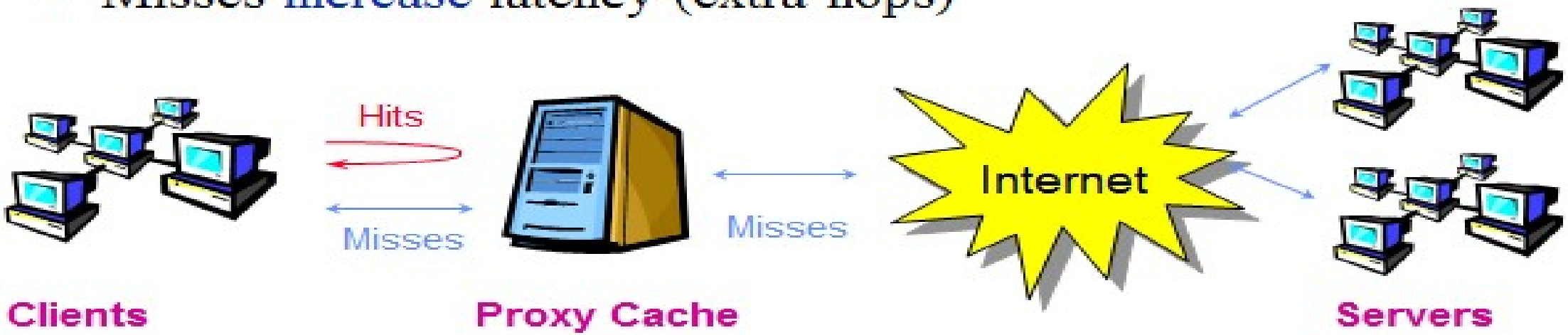


Proxy Caching

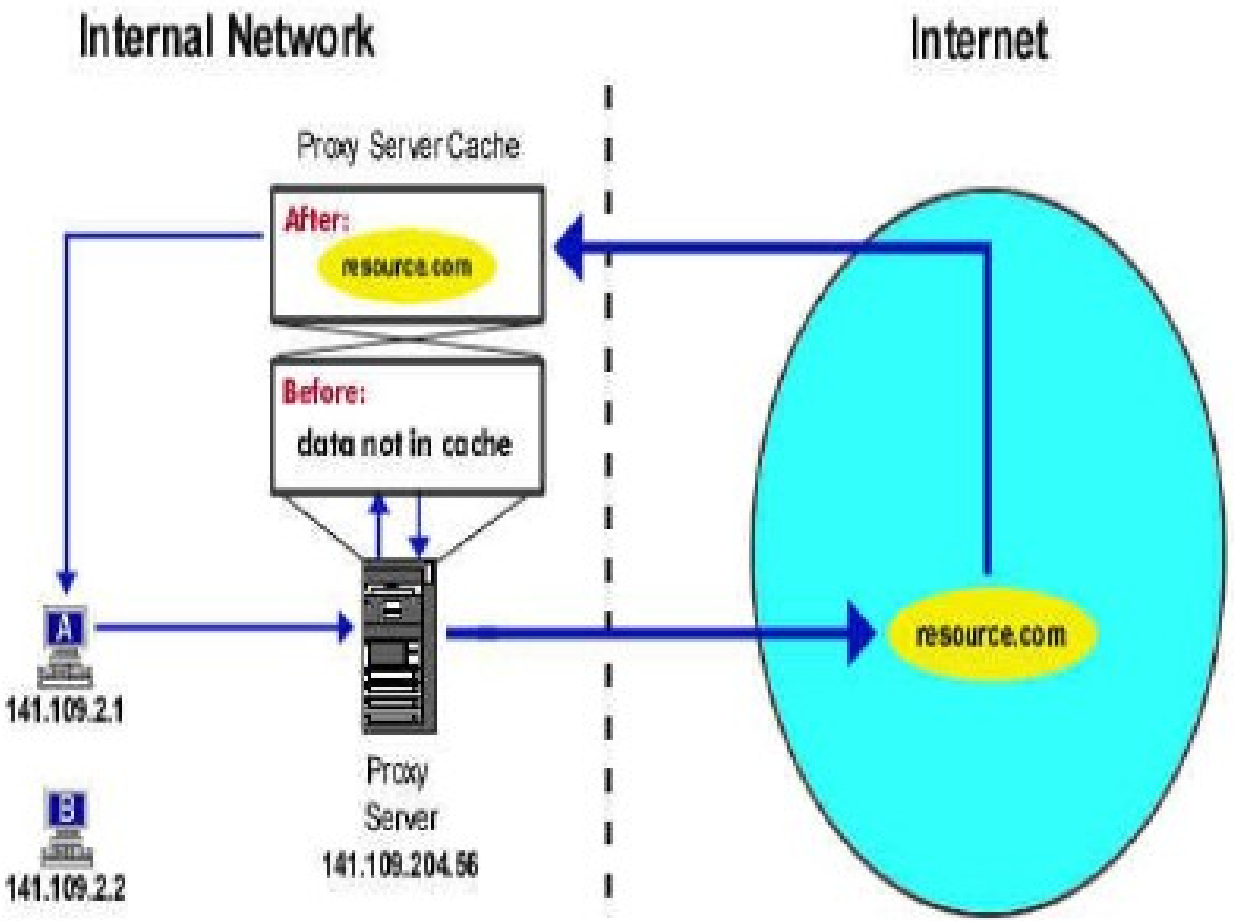
- Proxy server is a server (a computer system or an application) that **acts as an intermediary for requests** from clients seeking resources from other servers.
- Proxy server **stores all the data it receives** as a result of placing requests for information on the internet in it's *cache*.
  - Cache simply means memory
  - The cache is typically hard disk space, but it could be RAM
  - Caching documents means keeping a copy of internet documents so the server doesn't need to request them over again
  - With proxy caching, **clients make requests to servers, but the requests first go through a proxy cache.**

Caching for a Better Web

- Performance is a major concern in the Web
- Proxy caching is the most widely used method to improve Web performance
  - Duplicate requests to the same document served from cache
  - Hits reduce latency, bandwidth demand, server load
  - Misses increase latency (extra hops)



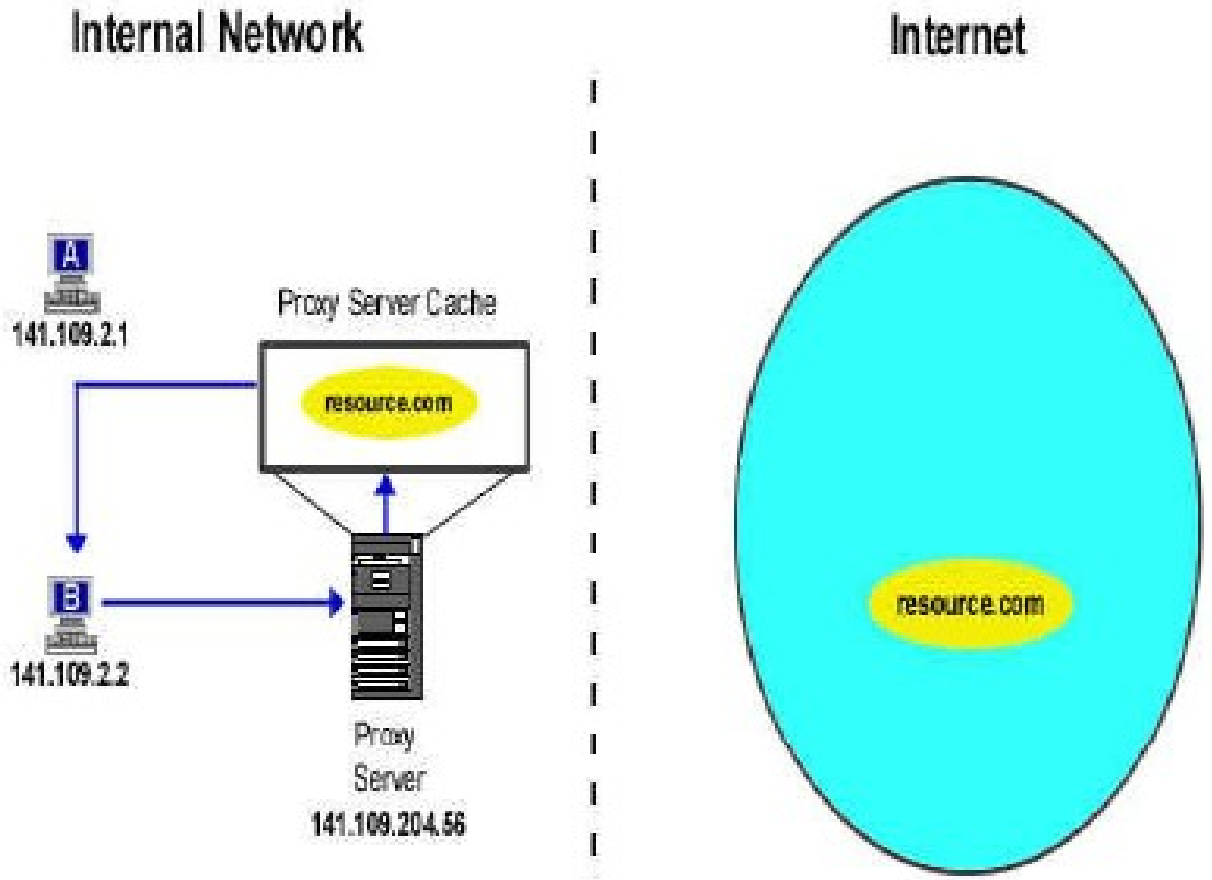
Caching a Document on a Proxy Server



Scenario 1:

- user **A** request a web page
- the request goes to the proxy server
- the proxy server checks to see if the document is stored in cache
- the document is not in cache so the request is sent to the Internet
- the proxy server receives the request, stores (or caches) the page
- the page is sent to user **A** where is viewed

Retrieving Cached Documents



Scenario 2:

- user **B** request the same page as user **A** (ie. resource.com)
- the request goes to the proxy server
- the proxy server checks its cache for the page
- the page is stored in cache
- the proxy server sends the page to user **B** where it is viewed
- no connection to the Internet is required

Internet Cookies

- Actual Definition
  - Short pieces of data used by web servers to help identify web users
  - Placed on your hard drive by server
  - Identifies your IP address during your visit
  - Helpful on shopping sites
- If no cookie, server issues one
- If already existent, cookie is sent to server
- Server reads cookie and sends it back with update
- Large provider with ad banners on many sites
- Cookies are visible on multiple sites for tracking
- Forms rich profiles of users and threatened to personalize

Internet Cookies

- ***What info can cookies return?***
  - *IP address*, ***your type of browser***, *operating system*, ***number of visits to web site***, *volunteered information*
- ***Who can see the cookies?***
  - Only the server that put the cookie on your hard drive.
  - Larger sites can set a domain attribute for other servers in the same sub domain
- ***Why a bad reputation for cookies?***
  - when used as a tracking device
  - can see what web pages you visit, how often, and how long
  - clicking on ad banners
  - used to infer your interests

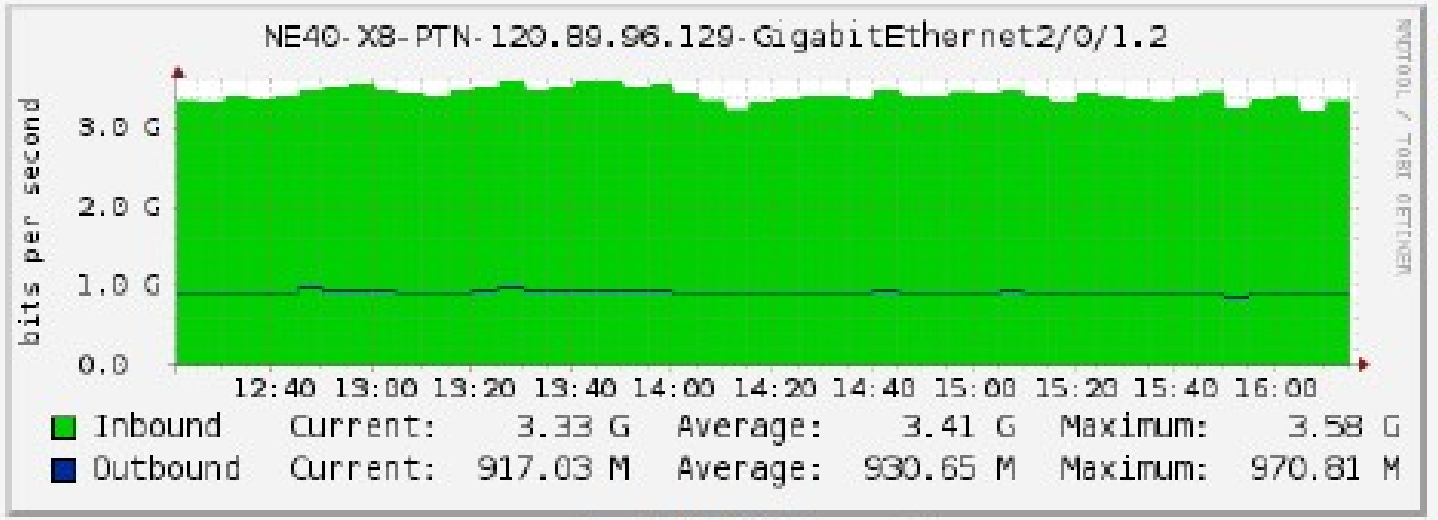
MRTG (Multi Router Traffic Grapher)

- Most common network traffic measurement tool
- It is free software for monitoring and measuring the traffic load on network links. It allows the user to see traffic load on a network over time in graphical form.
- It was originally developed by “Tobias Oetiker and Dave Rand” to monitor router traffic, but has developed into a tool that can create graphs and statistics for almost anything.
- Monitors bits in and out of a network device (e.g.. Switch port, router port, NIC card)
- MRTG uses simple SNMP queries on a regular interval to generate graphs.
- MRTG generates HTML pages containing PNG images which provide an almost live visual representation of this traffic.

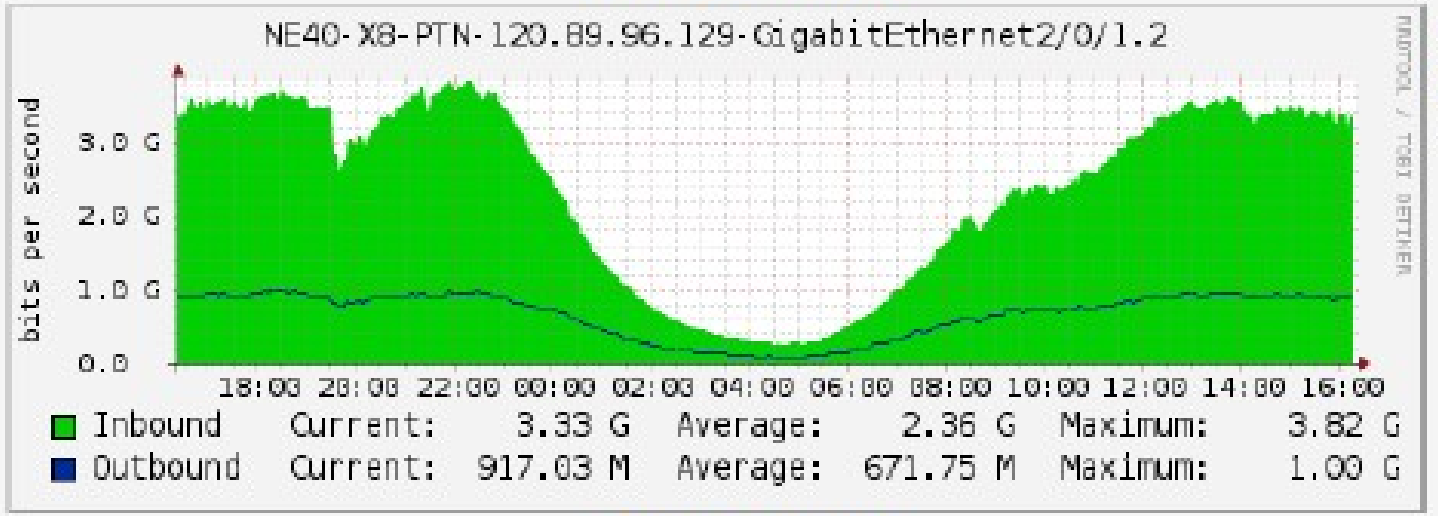
MRTG (Multi Router Traffic Grapher)

- MRTG measures bandwidth
- MRTG software can be used not only to measure network traffic on interfaces, but also build graphs of anything that has an equivalent SNMP MIB - like CPU load, Disk availability, temperature, etc...
- **MRTG: Issues**
  - MRTG generates each graph (what if you have hundreds of graphs!) every 5 minutes, creating a lot of overhead.
  - It also has very few customizable graphing options.
  - Disk space is always an issue.
  - MRTG management itself can be tedious work.

Viewing Graph 'NE40-X8-PTN-120.89.96.129-GigabitEthernet2/0/1.2'



Hourly (1 Minute Average)



Daily (5 Minute Average)

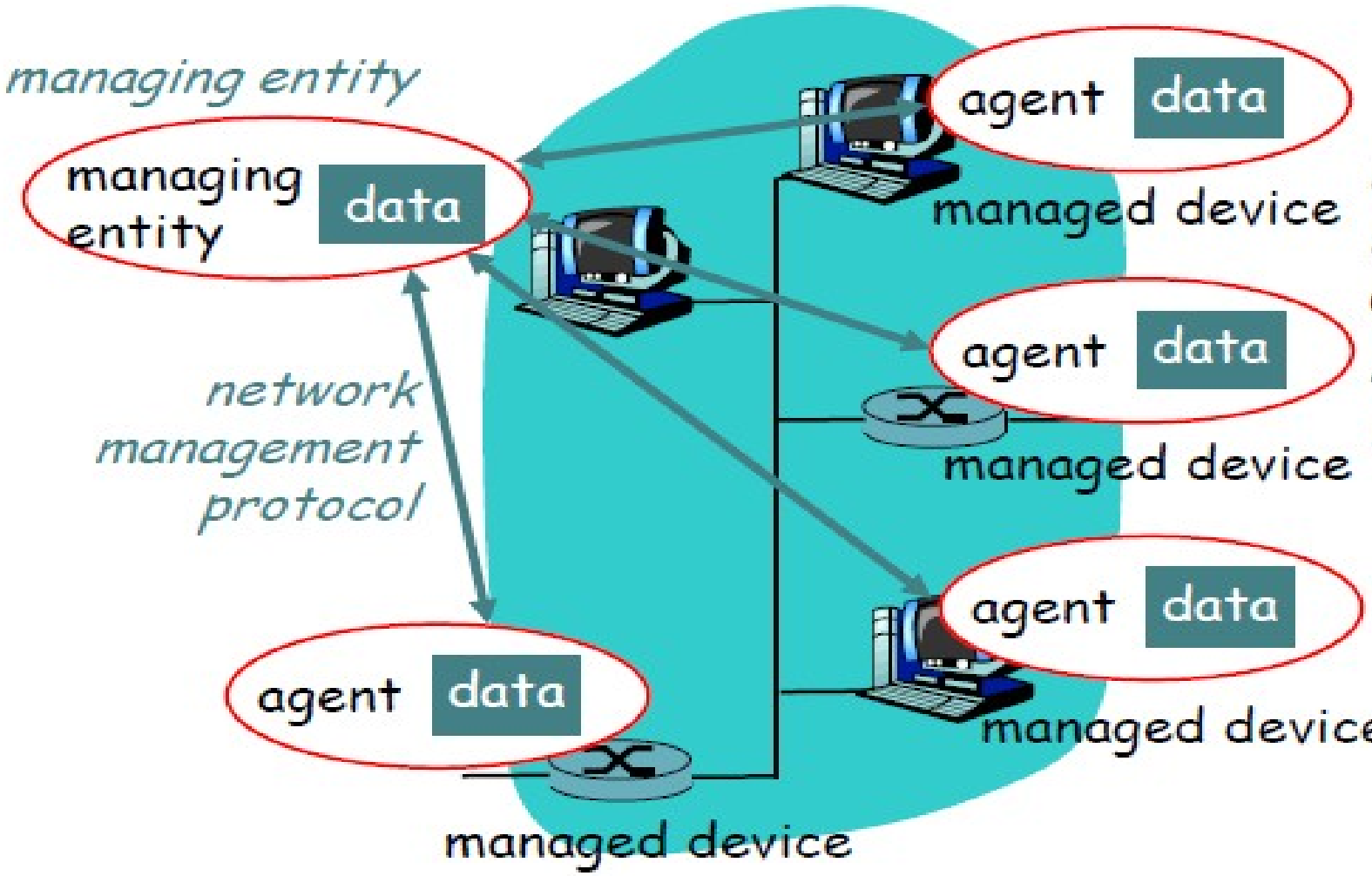
SNMP

- Simple Network Management Protocol (SNMP) is an "Internet-standard protocol **for managing devices on IP networks**."
- Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks, and more.
- **It allows Administrators to**
  - manage network performance,
  - find and solve network problems, and
  - plan for network growth
- **SNMP works as the manager/agent model.**
  - Manager (the monitoring ”client”)
  - Agent (running on the equipment server)

SNMP

- The manager and agent use a **Management Information Base (MIB)** and a relatively small **set of commands to exchange information**.
- MIBs are files **defining the objects** that can be **queried, including**:
  - Object name
  - Object description
  - Data type (integer, text, list)
- The MIB is organized **in a tree structure** with individual variables, represented as leaves on the branches.
- A long numeric tag or **object identifier (OID)** is used to **distinguish each variable uniquely in the MIB and in SNMP messages**.

SNMP-Infrastructure for network management



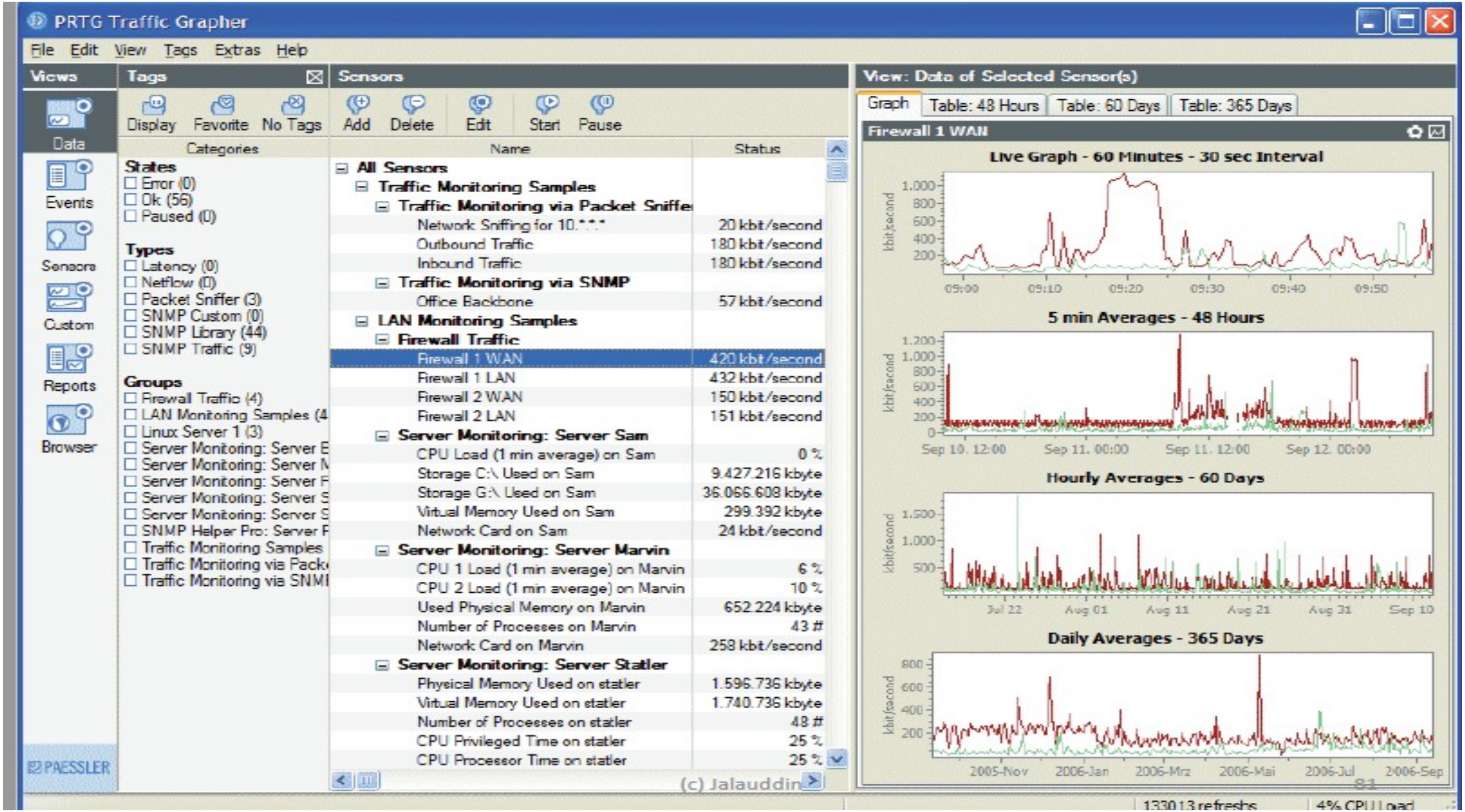
managed devices contain *managed objects* whose data is gathered into a *Management Information Base (MIB)*

SNMP Version 3

- SNMP versions **1 and 2 are insecure**
- SNMP version **3 has been created to fix this**
- The most common module is based in user, or a “User-based Security Model”
  - **Authenticity and integrity**: Keys are used for users and messages have digital signatures generated with a hash function (MD5 or SHA)
  - **Privacy:** Messages can be encrypted with secret-key (private) algorithms (DES)
  - **Temporary validity:** Utilizes a synchronized clock with a 150 second window with sequence checking

Paessler Router Traffic Grapher(PRTG)

- An easy to use **Windows-based software** for monitoring **network & bandwidth usage** as well as various **other network** parameters **like memory and CPU utilization**
- Provides system administrators with **live readings and periodical usage** trends of leased lines, routers, firewalls, servers, and many other network devices
- **Features**
  - Supports **data acquisition** via SNMP, packet sniffing, or Netflow
  - **Classifies** network traffic **by IP address, protocol, and other parameters**
  - **Easy installation** & use on Windows 2000/XP/2003
  - **Capable of** monitoring up to **several thousand sensors**
- **SNMP** to access traffic counters or other readings from **SNMP enabled devices**
- **Packet Sniffing** to look **at incoming/outgoing network packets** that pass through a **network card of a computer**
- **NetFlow** for analyzing **Cisco NetFlow packets sent by Cisco routers**


PRTG Traffic Grapher
File Edit View Tags Extras Help
Views
Data
Events
Sensors
Custom
Reports
Browser
PAESSLER
Tags
Display Favorite No Tags
Categories
States
Error (0)
Ok (56)
Paused (0)
Types
Latency (0)
Netflow (0)
Packet Sniffer (3)
SNMP Custom (0)
SNMP Library (44)
SNMP Traffic (9)
Groups
Firewall Traffic (4)
LAN Monitoring Samples (4
Linux Server 1 (3)
Server Monitoring: Server E
Server Monitoring: Server M
Server Monitoring: Server F
Server Monitoring: Server S
Server Monitoring: Server S
SNMP Helper Pro: Server F
Traffic Monitoring Samples
Traffic Monitoring via Packe
Traffic Monitoring via SNMI
Sensors
Add Delete Edit Start Pause
Name
Status
All Sensors
Traffic Monitoring Samples
Traffic Monitoring via Packet Sniffe
Network Sniffing for 10.*.*.* 20 kbit/second
Outbound Traffic 180 kbit/second
Inbound Traffic 180 kbit/second
Traffic Monitoring via SNMP
Office Backbone 57 kbit/second
LAN Monitoring Samples
Firewall Traffic
Firewall 1 WAN 420 kbit/second
Firewall 1 LAN 432 kbit/second
Firewall 2 WAN 150 kbit/second
Firewall 2 LAN 151 kbit/second
Server Monitoring: Server Sam
CPU Load (1 min average) on Sam 0 %
Storage C:\ Used on Sam 9.427.216 kbyte
Storage G:\ Used on Sam 36.066.608 kbyte
Virtual Memory Used on Sam 299.392 kbyte
Network Card on Sam 24 kbit/second
Server Monitoring: Server Marvin
CPU 1 Load (1 min average) on Marvin 6 %
CPU 2 Load (1 min average) on Marvin 10 %
Used Physical Memory on Marvin 652.224 kbyte
Number of Processes on Marvin 43 #
Network Card on Marvin 258 kbit/second
Server Monitoring: Server Statler
Physical Memory Used on statler 1.596.736 kbyte
Virtual Memory Used on statler 1.740.736 kbyte
Number of Processes on statler 48 #
CPU Privileged Time on statler 25 %
CPU Processor Time on statler 25 %
(c) Jalauddin
View: Data of Selected Sensor(s)
Graph
Table: 48 Hours
Table: 60 Days
Table: 365 Days
Firewall 1 WAN
Live Graph - 60 Minutes - 30 sec Interval
kbit/second
1.000
800
600
400
200
09:00
09:10
09:20
09:30
09:40
09:50
5 min Averages - 48 Hours
1.200
1.000
800
600
400
200
0
Sep 10, 12:00
Sep 11, 00:00
Sep 11, 12:00
Sep 12, 00:00
Hourly Averages - 60 Days
1.500
1.000
500
Jul 22
Aug 01
Aug 11
Aug 21
Aug 31
Sep 10
Daily Averages - 365 Days
800
600
400
200
2005-Nov
2006-Jan
2006-Mrz
2006-Mai
2006-Jul
2006-Sep
133013 refreshs
4% CPU Load

Socket programming

Goal: learn how to build client/server application that communicate using sockets

Socket API

- explicitly created, used, released by apps
- client/server paradigm
- two types of transport service via socket API:
  - unreliable datagram
  - reliable, byte stream-oriented

socket

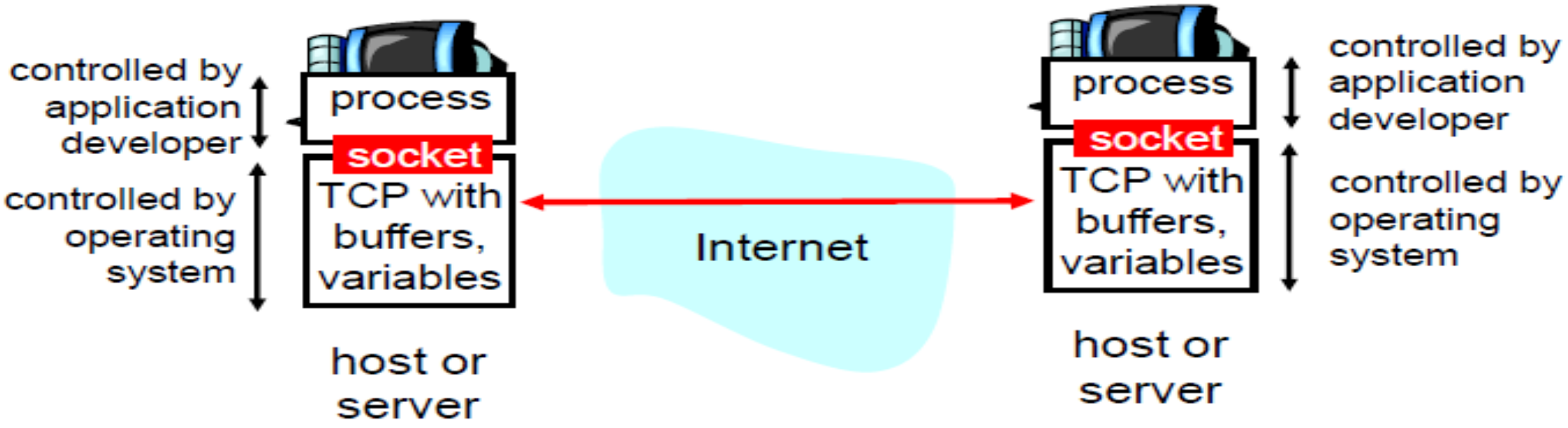
a *host-local*, *application created*, *OS-controlled* interface (a "door") into which application process can both send and receive messages to/from another application process

Socket programming with TCP

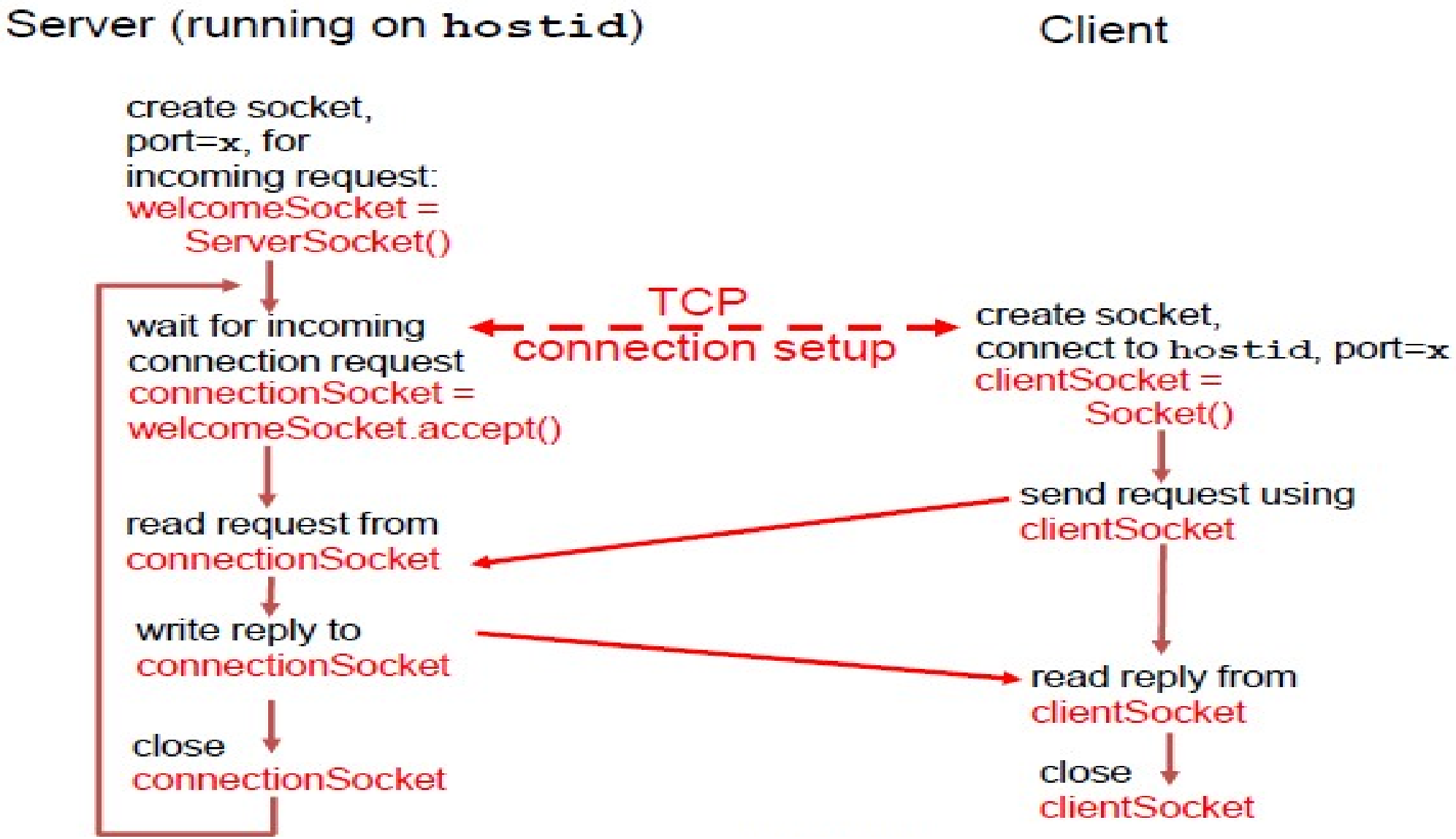
Socket: a door between application process and end-end-transport protocol (UDP or TCP)

- An abstract interface provided to the application programmer

TCP service: reliable transfer of bytes from one process to another



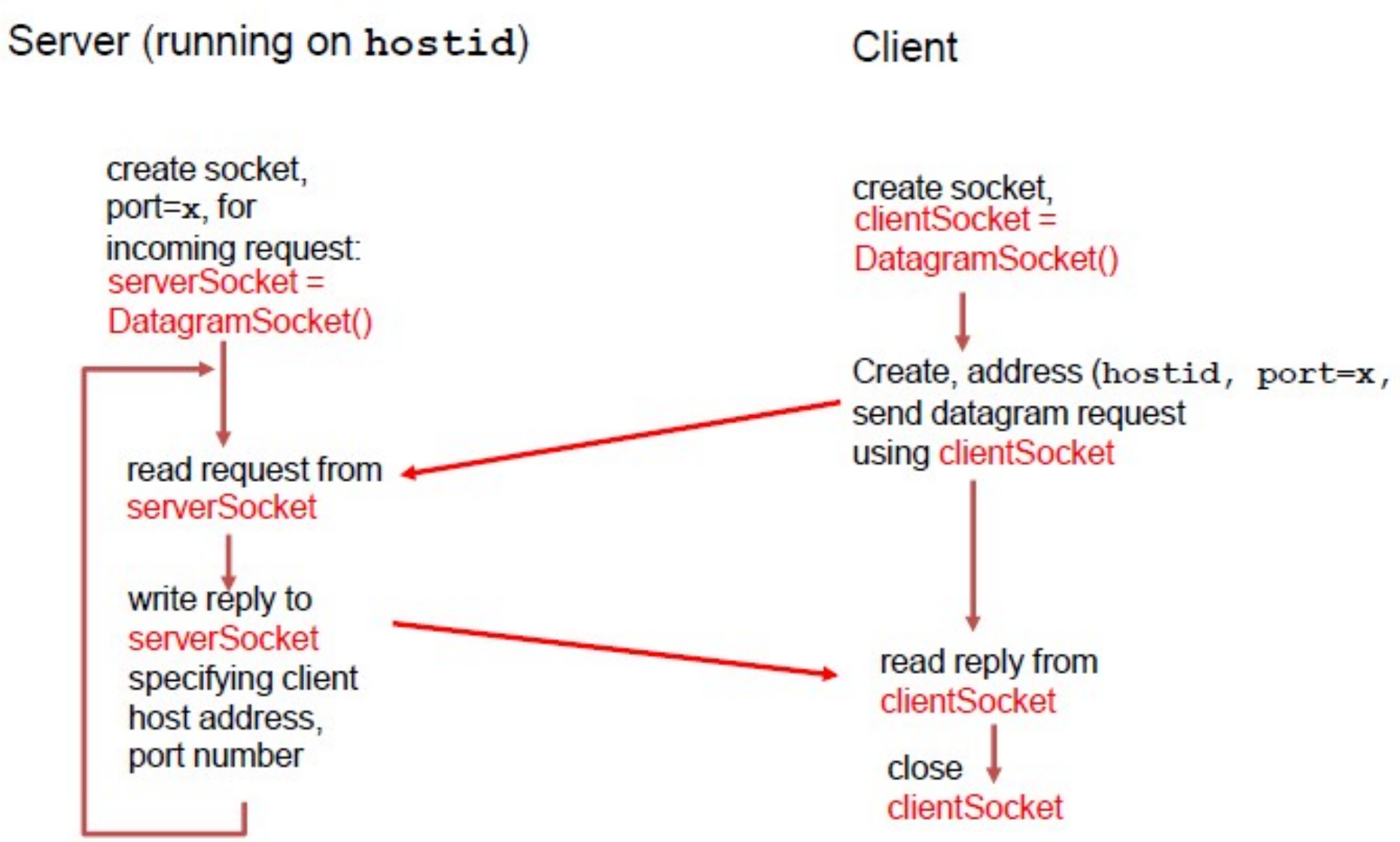
Client-server socket interaction: TCP



Socket programming *with UDP*

- **UDP:** no "**connection**" between client and server
- **No handshaking**
- **Sender** explicitly attaches **IP address and port of destination** to each packet. Server must extract IP address, port of sender from received packet to return uppercase sentence to sender
- **UDP:** transmitted data may **be received out of order, or lost**
- application viewpoint *UDP provides unreliable transfer of groups of bytes ("datagrams") between client and server*

Client-server socket interaction: UDP



Thank You
???

References:

- Data Communications and Networking “ Behrouz A. Forouzan”
- Computer Networks “A. S. Tanenbaum” Fifth Edition
- Data and Computer Communications “William Stallings” Tenth Edition.