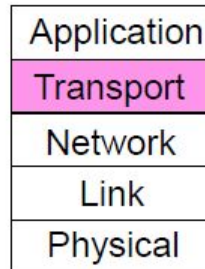


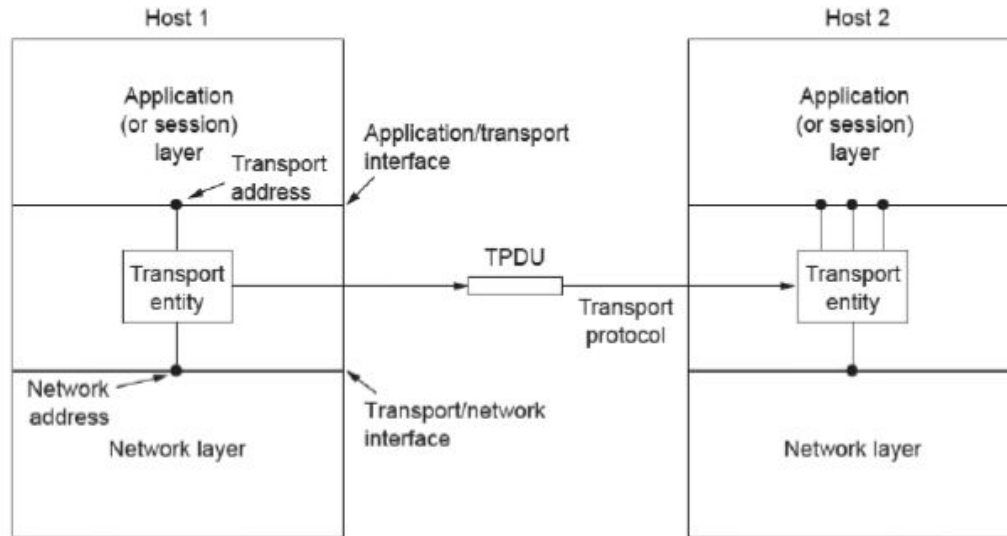
20MCA13-Computer Networks

Unit 5-Transport Layer, Application Layer

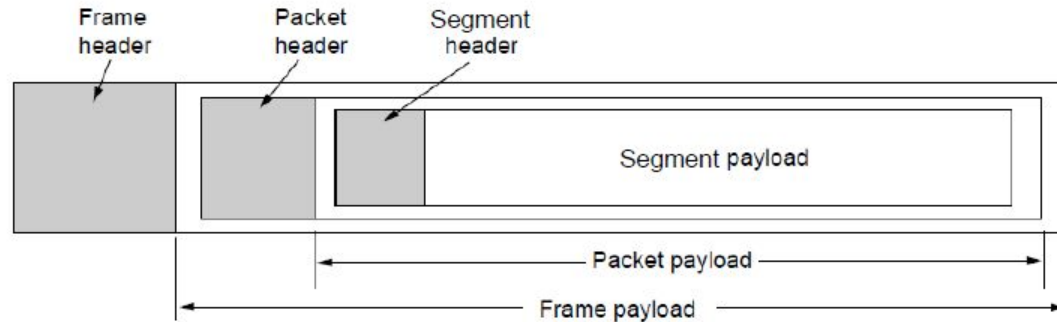
- Responsible for delivering data across networks with the desired reliability or quality
- Transport Service
 - Services Provided to the Upper Layer
 - Transport Service Primitives



- Transport layer adds reliability to the network layer offers connectionless (e.g., UDP) and connection oriented (e.g, TCP) service to applications



- Transport layer sends segments in packets (in frames)



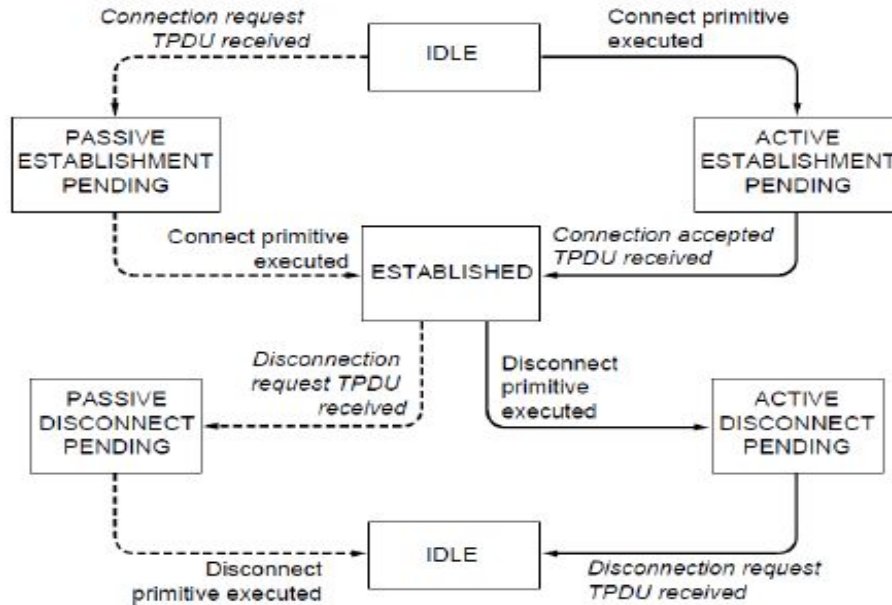
Transport Service Primitives

- Primitives that applications might call to transport data for a simple connection oriented service:
 - Client calls CONNECT, SEND, RECEIVE, DISCONNECT
 - Server calls LISTEN, RECEIVE, SEND, DISCONNECT

Primitive	Segment sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Transport Service Primitives

State diagram for a simple connection-oriented service



Solid lines (right) show client state sequence

Dashed lines (left) show server state sequence

Transitions in italics are due to segment arrivals.

Transport Layer Design Issues

- Transport layer delivers message from one process to another process running on two different hosts.
- Performs number of functions to ensure the accurate delivery of message.
- The various functions of transport layer are:
 - Establishing, Maintaining & Releasing Connection
 - Addressing
 - Data Transfer
 - Flow Control
 - Error Control
 - Multiplexing
 - Congestion Control

● **Establishing, Maintaining & Releasing Connection:**

- Transport layer establishes, maintains & releases end-to-end transport connection on the request of upper layers.
- Establishing a connection involves allocation of buffers for storing user data, synchronizing the sequence numbers of packets etc.
- A connection is released at the request of upper layer.

● Addressing:

- In order to deliver the message from one process to another, an addressing scheme is required.
- Several process may be running on a system at a time.
- To identify the correct process out of the various running processes, transport layer uses an addressing scheme called ***port number***.
- Each process has a specific port number.

- Transport layer breaks user data into smaller units and attaches a transport layer header to each unit forming a TPDU (TransPort Layer Data Unit).
- TPDU is handed over to the network layer for its delivery to destination.
- TPDU header contains port number, sequence number, acknowledgement number, checksum and other fields.

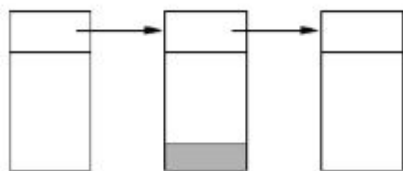
● **Flow Control:**

- Like data link layer, transport layer also performs flow control.
- Flow control at transport layer performed end-to-end rather than node-to-node.
- Uses a sliding window protocol to perform flow control.

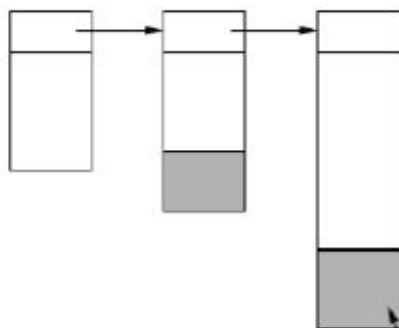
● Error Control:

- Transport layer also provides end-to-end error control facility.
- Deals with several different types of errors:
 - Error due to damaged bits.
 - Error due to non delivery of TPDUs.
 - Error due to duplicate delivery of TPDUs.
 - Error due to delivery of TPDU to a wrong destination.

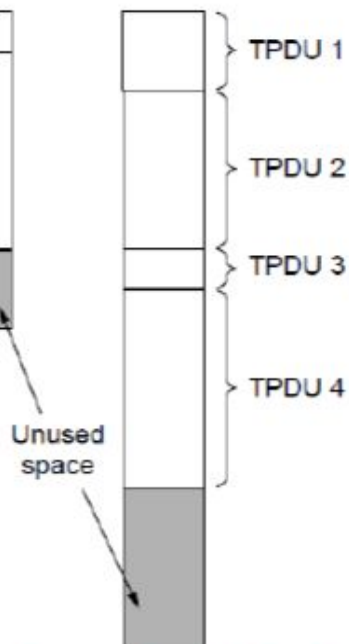
Different buffer strategies trade efficiency / complexity



a) Chained fixed-size buffers



b) Chained variable-size buffers



c) One large circular buffer

Crash recovery

Application needs to help recovering from a crash

- Transport can fail since A(ck) / W(rite) not atomic

Strategy used by sending host	Strategy used by receiving host					
	First ACK, then write			First write, then ACK		
	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

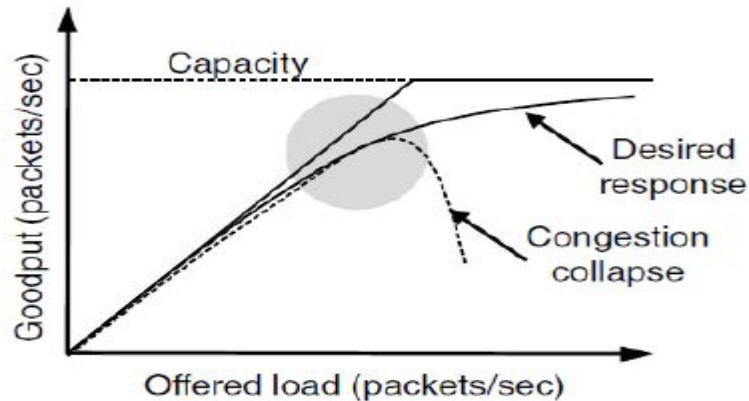
OK = Protocol functions correctly
 DUP = Protocol generates a duplicate message
 LOST = Protocol loses a message

● **Congestion Control:**

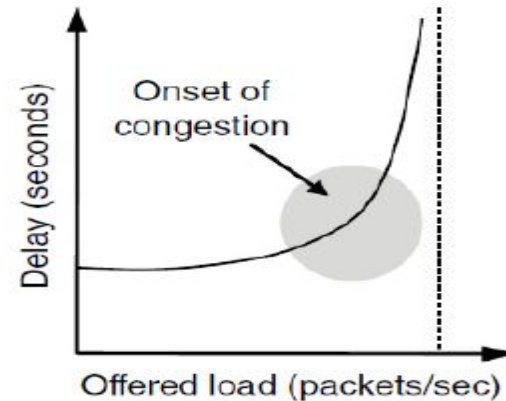
- Transport layer also handles congestion in the networks.
- Several different congestion control algorithms are used to avoid congestion.

Desirable Bandwidth Allocation

Efficient use of bandwidth gives high goodput, low delay



Goodput rises more slowly than load when congestion sets in

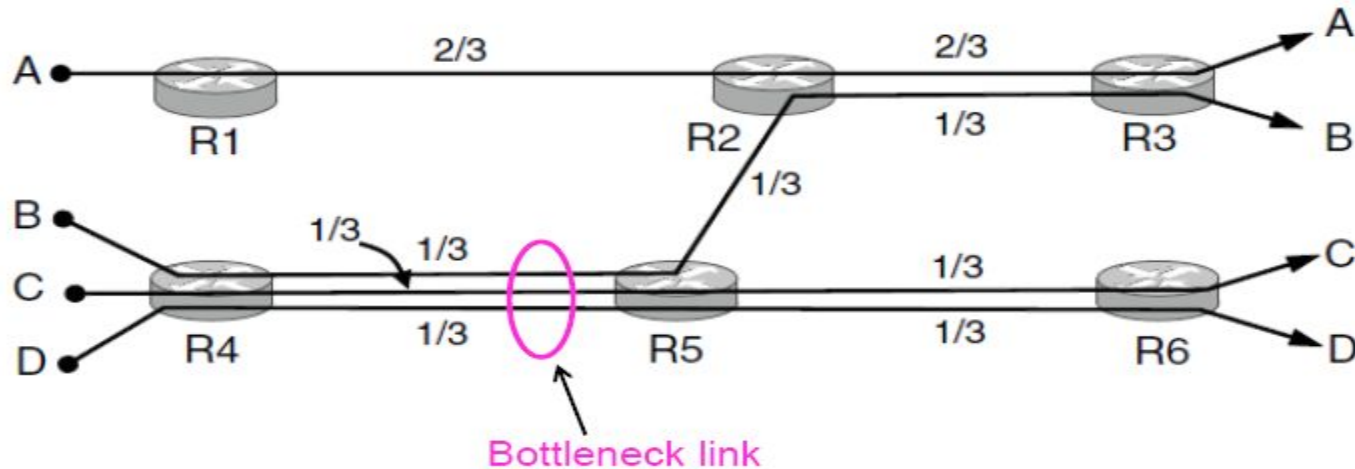


Delay begins to rise sharply when congestion sets in

Desirable Bandwidth Allocation

Fair use gives bandwidth to all flows (no starvation)

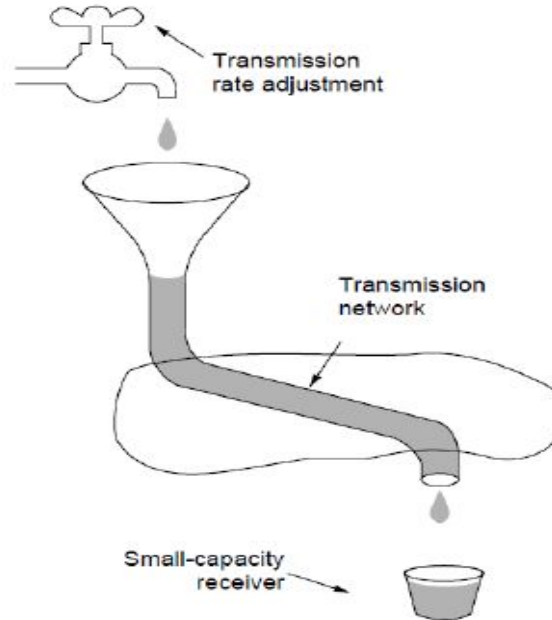
- Max-min fairness gives equal shares of bottleneck



Regulating Sending Rate

Sender may need to slow down for different reasons:

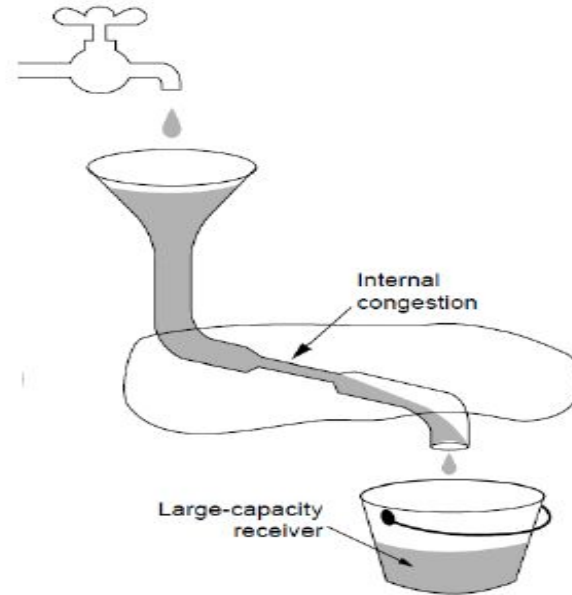
- Flow control, when the receiver is not fast enough [right]
- Congestion, when the network is not fast enough [over]



A fast network feeding a low-capacity receiver
→ flow control is needed

Regulating the sending rate

Our focus is dealing with this problem – congestion



A slow network feeding a high-capacity receiver
→ congestion control is needed

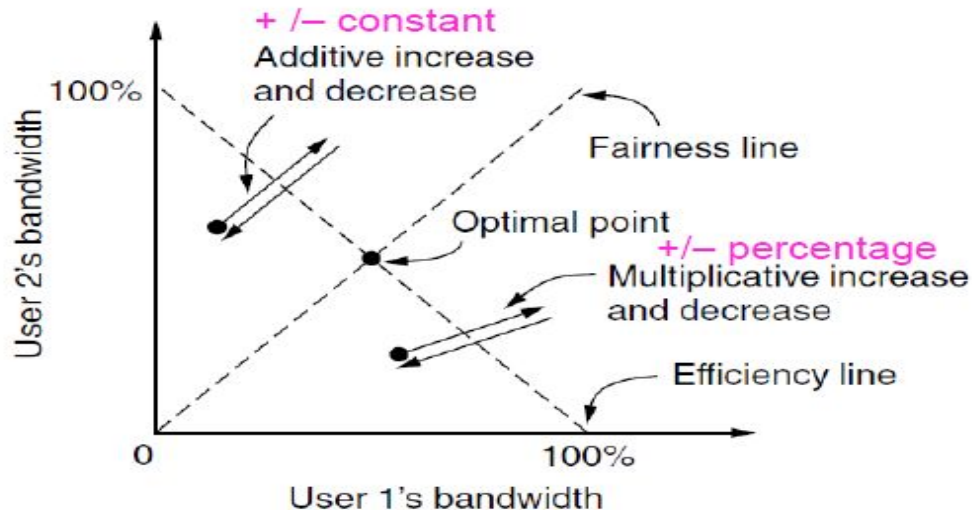
Regulating the sending rate

Different congestion signals the network may use to tell the transport endpoint to slow down (or speed up)

Protocol	Signal	Explicit?	Precise?
XCP	Rate to use	Yes	Yes
TCP with ECN	Congestion warning	Yes	No
FAST TCP	End-to-end delay	No	Yes
CUBIC TCP	Packet loss	No	No
TCP	Packet loss	No	No

Regulating the sending rate

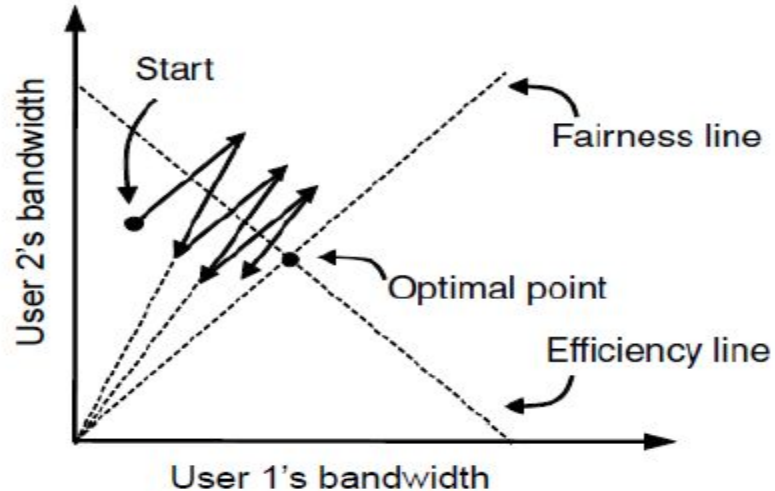
If two flows increase/decrease their bandwidth in the same way when the network signals free/busy they will not converge to a fair allocation



Regulating the sending rate

The AIMD (Additive Increase Multiplicative Decrease) control law does converge to a fair and efficient point!

- TCP uses AIMD for this reason



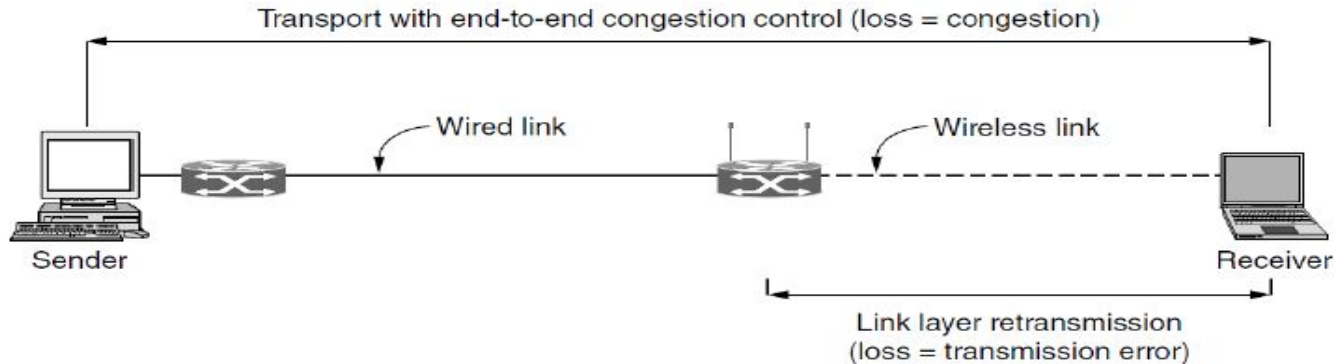
Wireless issues

Wireless links lose packets due to transmission errors

- Do not want to confuse this loss with congestion
- Or connection will run slowly over wireless links!

Strategy:

- Wireless links use ARQ, which masks errors



Transport Layer Services

- Transport layer protocols can provide two types of services:
 - Connection Oriented Service
 - Connectionless Service
- **Connection Oriented Service:**
 - A connection is first established between sender and the receiver.
 - Transfer of user data takes place.
 - Connection is released.
 - Generally reliable.
 - Transport layer protocols that provide connection oriented service are **TCP** and **SCTP (Stream Control Transmission Protocol)**.

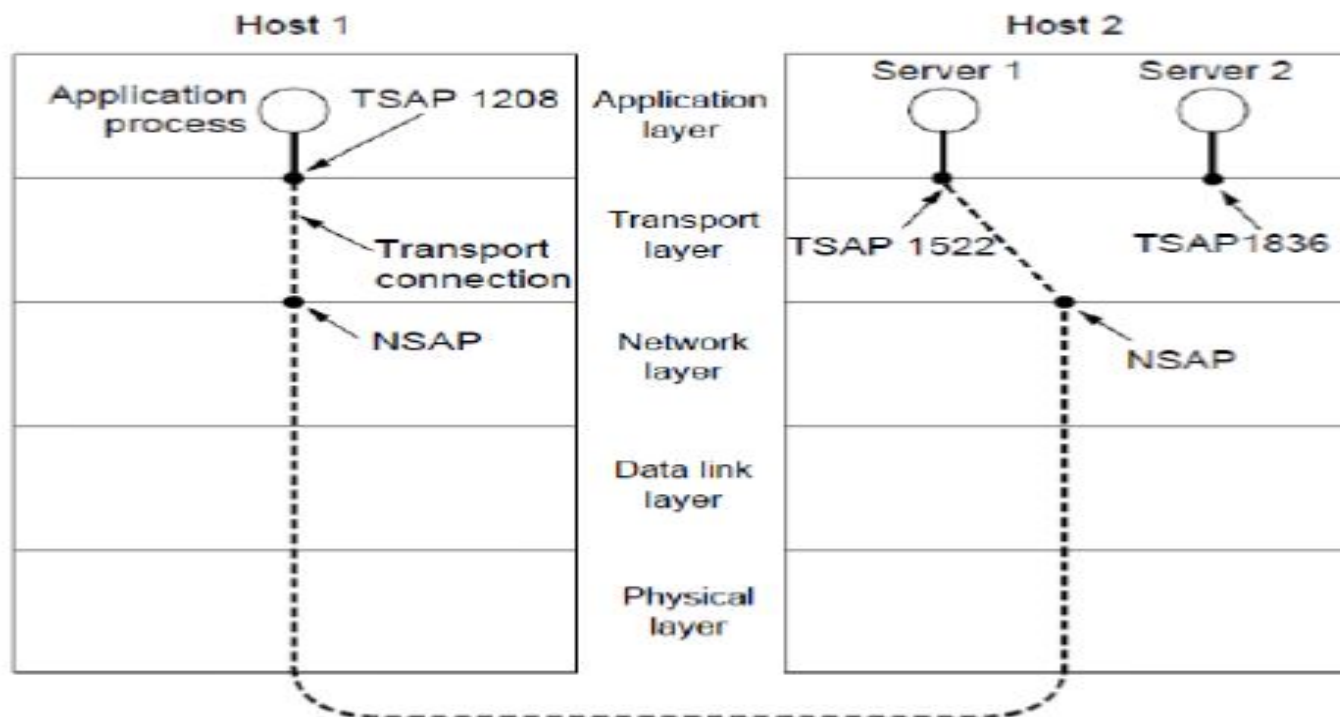
● Connectionless Service:

- Packets are sent from sender to receiver without the establishment of connection.
- Packets are not numbered.
- Packets may be lost, corrupted, delayed or disordered.
- Connectionless service is unreliable.
- Transport layer protocol that provides this service is **UDP**.

Elements of Transport protocols

- **Addressing:**

- In order to deliver data from one process to another, address is required.
- In order to deliver data from one node to another, MAC address is required.
- Such an address is implemented at Data Link Layer and is called **Physical Addressing**.
- In order to deliver data from one network to another, IP address is required.
- Such an address is implemented at Network Layer and is called **Logical Addressing**.
- Similarly, in order to deliver data from a process running on source to process running on destination, transport layer defines the **Service Point Address** or **Port Numbers**.



● Port Numbers:

- Each communicating process is assigned a specific port number.
- In order to select among multiple processes running on a destination host, a port number is required.
- Port numbers are 16-bit integers between 0 and 65,535.
- Port numbers are assigned by Internet Assigned Number Authority (IANA).
- IANA has divided the port numbers in three categories:
 - **Well Known Ports:** Ports ranging from 0 to 1023. For e.g.: HTTP: 80, SMTP: 25, FTP: 21.
 - **Registered Ports:** Ports ranging from 1024 to 49,151 and are not controlled by IANA.
 - **Dynamic Ports:** Ports ranging from 49,152 to 65,535 and can be used by any process.

● **Socket Address:**

- Socket address is a combination of IP address and port number.
- To provide communication between two different processes on different networks, both IP address and port number, i.e. socket address is required.

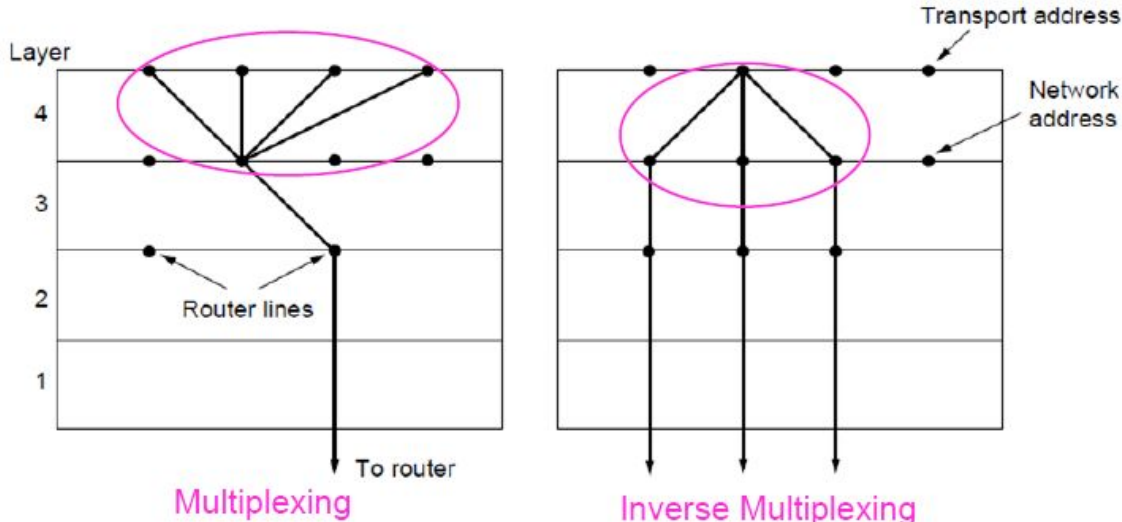
- **Multiplexing & Demultiplexing:**

- A network connection can be shared by various applications running on a system.
- Several running processes want to send data and only one transport layer connection available, where transport layer protocols perform multiplexing.
- Protocol accepts the messages from different processes having their respective port numbers, and add headers to them.
- Transport layer at the receiver end performs demultiplexing to separate the messages for different processes.
- After checking for errors, the headers of messages are dropped and each message is handed over to the respective processes based on their port numbers.

Multiplexing

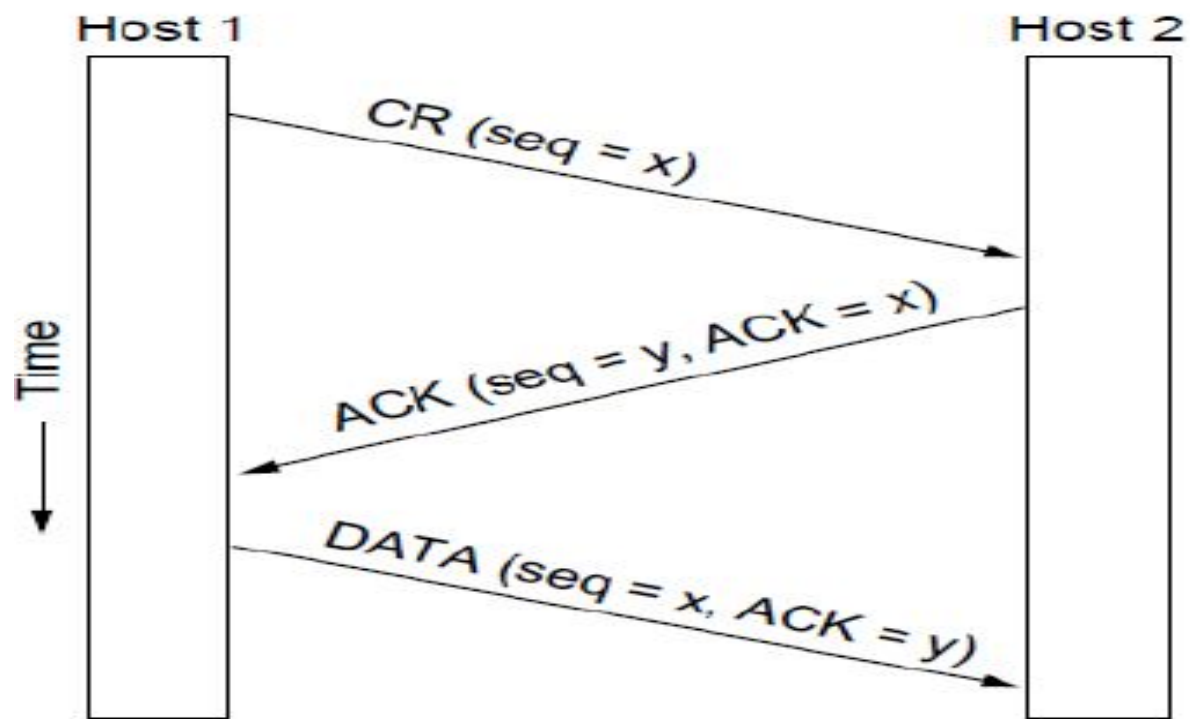
Kinds of transport / network sharing that can occur:

- Multiplexing: connections share a network address
- Inverse multiplexing: addresses share a connection



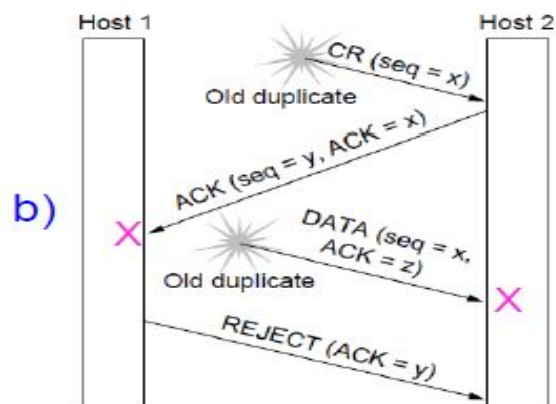
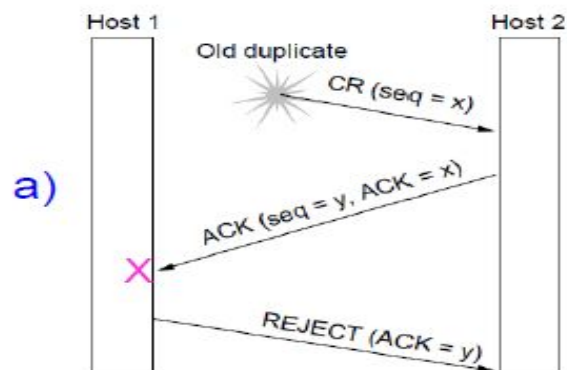
- **Connection Establishment:**

- Before communicating, the source device must first determine the availability of the other to exchange data.
- Path must be found through the network by which the data can be sent called as Connection Establishment.
- Connection establishment involves **Three-Way Handshaking** mechanism:
 - The source sends a **connection request** packet to the destination.
 - The destination returns a confirmation packet back to the source.
 - The source returns a packet acknowledging the confirmation.



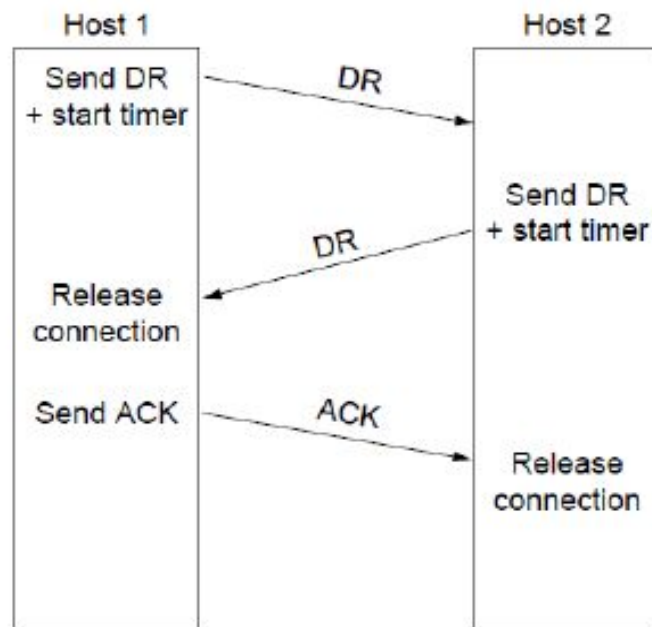
Three-way handshake protects against odd cases:

- a) Duplicate CR. Spurious ACK does not connect
- b) Duplicate CR and DATA. Same plus DATA will be rejected (wrong ACK).



Normal release sequence,
initiated by transport user on
Host 1

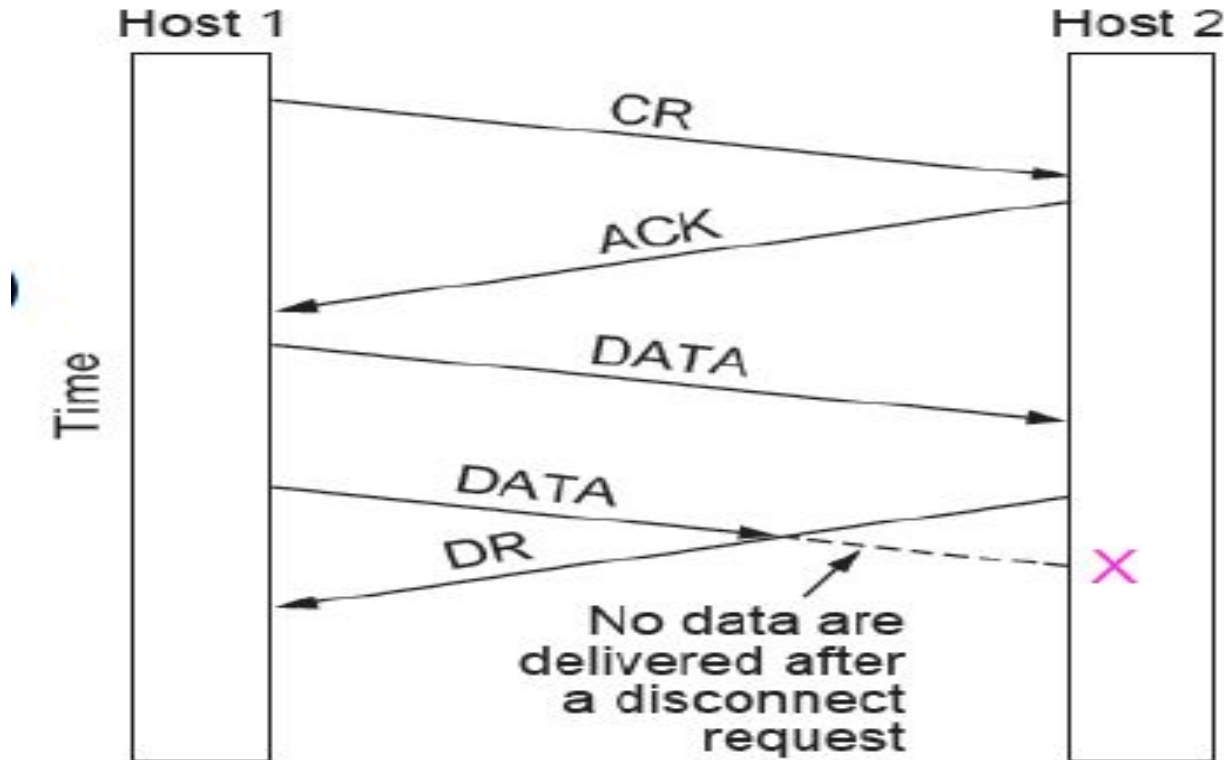
- DR=Disconnect Request
- Both DRs are ACKed by the other side



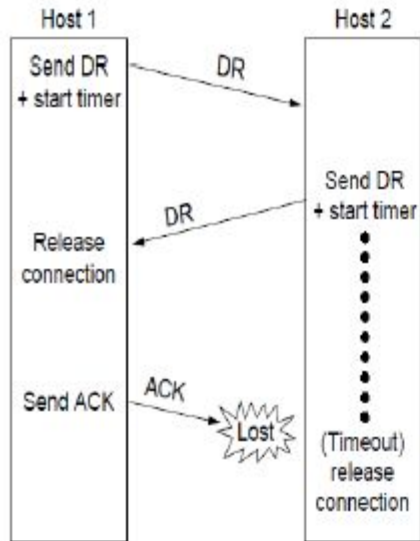
● Connection Release:

- Once all of the data has been transferred, the connection must be released.
- Requires a **Three-Way Handshaking** mechanism:
 - The source sends a **disconnect request packet to the** destination.
 - The destination returns a confirmation packet back to the source.
 - The source returns a packet acknowledging the confirmation.

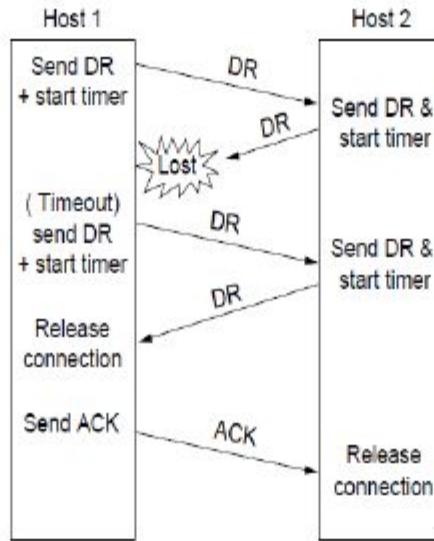
Connection Release



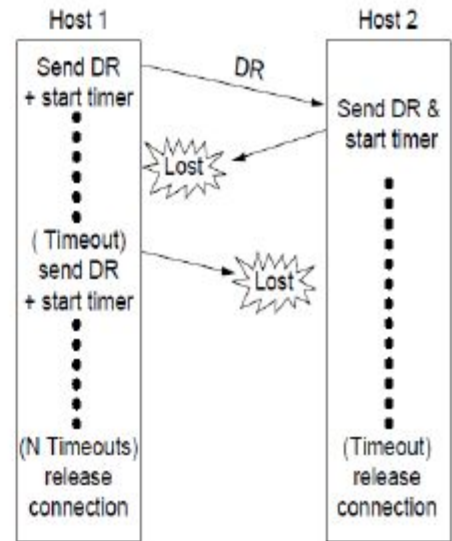
Error cases are handled with timer and retransmission



Final ACK lost,
Host 2 times out



Lost DR causes
retransmissions



Extreme: Many lost
DRs cause both
hosts to timeout

Transport Layer Protocols

- Transport layer provides two types of services:
 - Connection Oriented Service
 - Connectionless Service
- For this, transport layer defines two different protocols:
 - Transmission Control Protocol (TCP)
 - User Datagram Protocol (UDP)

Transmission Control protocol (TCP)

- Connection oriented protocol that provides reliable services between processes on different hosts.
- Uses the services of lower layer which provide connectionless and unreliable service.
- The basic features of TCP are:
 - Provides efficient method for numbering different bytes of data.
 - Provides stream data transfer.
 - Offers reliability.
 - Provides efficient flow control.
 - Provides full duplex operation.
 - Provides multiplexing.
 - Provides connection oriented service.

TCP Segment

- TCP segment is the unit of data transferred between two processes.
 - Each TCP segment consists of two parts:
 - Header Part
 - Data Part

Format of TCP Segment



- **Source Port:**
 - It indicates the port number of a source process. It is of 2 bytes.
- **Destination Port:**
 - It indicates the port number of destination process. It is also 2 bytes.
- **Sequence Number:**
 - It specifies the number assigned to the current message. It is of 4 bytes.
- **Acknowledgement Number:**
 - It indicates the sequence number of the next byte of data. It is of 4 bytes.
- **Header Length:**
 - It indicates number of words in the TCP header. It is a 4 bit field.
- **Reserved:**
 - This 6 bit field is reserved for future use.

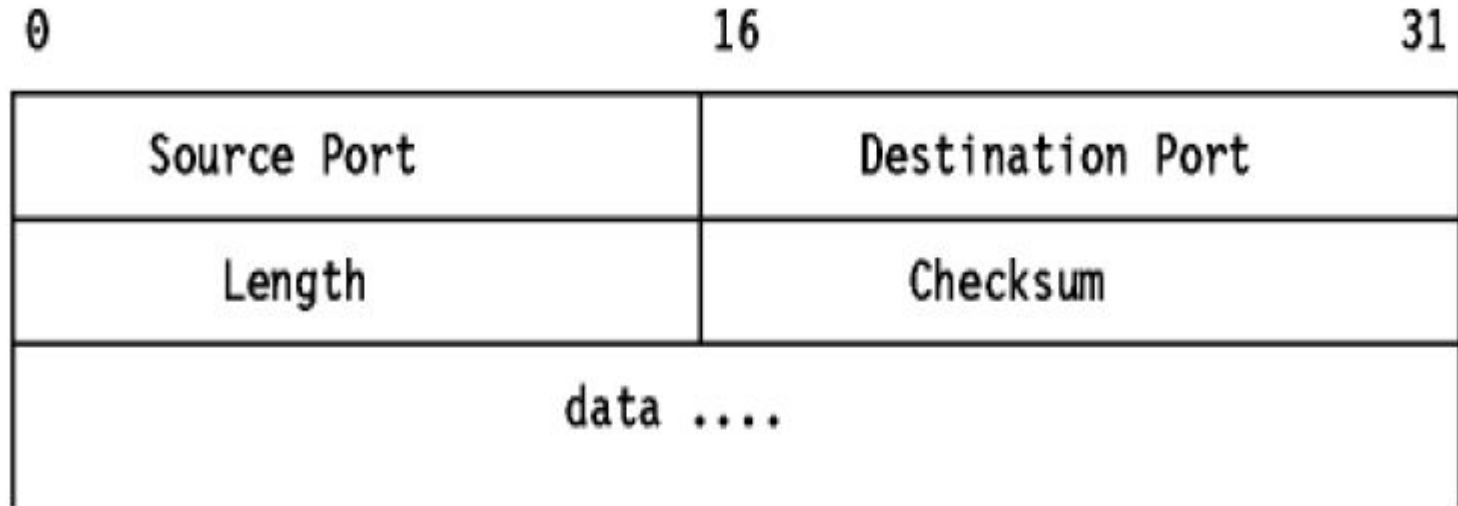
- **Flags:**
 - This 6 bit field consists of 6 different flags:
 - URG (Urgent Pointer)
 - ACK (Acknowledgement)
 - PSH (Request for Push)
 - RST (Reset the Connection)
 - SYN (Synchronize)
 - FIN (Final or Terminate the Connection)
- **Window:**
 - It specifies the size of sender's receiving window, i.e., the buffer space available for incoming data. It is of 2 bytes.
- **Checksum:**
 - This 16-bit field contains the checksum.
- **Urgent Pointer:**
 - This 16-bit field is valid only if urgent pointer in flags is set to 1.
- **Options:**
 - It contains the optional information in the TCP header. It is of 32 bytes.
- **Data:**
 - This field contains the upper layer information. It is of variable size.

User Datagram Protocol (UDP)

- A connectionless, unreliable transport protocol.
- UDP also provides process-to-process communication.
- Does not provide flow control and error control mechanisms unlike TCP.
- Connectionless and so transfers data without establishing a connection.

- The various features of UDP are:
 - Provides connectionless transport service.
 - Unreliable.
 - Does not provide flow control and error control.
 - less complex and simple than TCP, and easy to implement.
 - User datagrams (packets) are not numbered.
- A datagram is the unit of data transferred between two processes.
- Each UDP datagram consists of two parts:
 - Header Part
 - Data Part

Format of UDP Datagram



- **Source Port:**

- It indicates the port number of source process. It is of 16 bits.

- **Destination Port:**

- This 16 bit field specifies the port number of destination process.

- **Length:**

- It specifies the total length of the user datagram (header + data). It is of 16 bits.

- **Checksum:**

- The contains the checksum, and is optional. It is also of 16 bits.

Traditional Applications

- World Wide Web
 - Goal of the Web: To find a way to organize and retrieve information, drawing on ideas about hypertext—interlinked documents
 - core idea of hypertext is that one document can link to another document, and the protocol (HTTP) and document language (HTML) designed to meet that goal
 - Web: set of cooperating clients and servers, all of whom speak the same language: HTTP
 - URL: provide information that allows objects on the Web to be located, and look like following:
 - <https://rvce.edu.in/mca>

Traditional Applications

- World Wide Web

- If URL is given, Web browser open a TCP connection to the Web server at a machine called `https://rvce.edu.in` and immediately retrieve and display the file called `index.html`.
- Most files on the Web contain images and text and many have other objects such as audio and video clips, pieces of code, etc.
- frequently include URLs that point to other files located on other machines, that is the core of the “hypertext” part of HTTP and HTML

Traditional Applications

- World Wide Web

- to view a page, browser (the client) fetches it from the server using HTTP running over TCP
- Like SMTP, HTTP is a text oriented protocol.
- HTTP: request/response protocol, where every message has the general form

START_LINE <CRLF>

MESSAGE_HEADER <CRLF>

<CRLF>

MESSAGE_BODY <CRLF>

- <CRLF>stands for carriage-return-line-feed
- (START_LINE) indicates whether the message is a request message or a response message

Traditional Applications

- World Wide Web

- Request Messages

- first line of an HTTP request message specifies three things
 - operation to be performed
 - Web page the operation should be performed
 - version of HTTP being used.
 - Two most common operations of HTTP are GET (fetch the specified Web page) and HEAD (fetch status information about the specified Web page)

Traditional Applications

- World Wide Web
 - Request Messages

Operation	Description
OPTIONS	Request information about available options
GET	Retrieve document identified in URL
HEAD	Retrieve metainformation about document identified in URL
POST	Give information (e.g., annotation) to server
PUT	Store document under specified URL
DELETE	Delete specified URL
TRACE	Loopback request message
CONNECT	For use by proxies

HTTP request operations

Traditional Applications

- World Wide Web

- Response Messages

- response messages too begin with a single START LINE.
 - line specifies
 - Version of HTTP being used
 - Three-digit code indicating whether or not the request was successful
 - Text string giving the reason for the response

Traditional Applications

- World Wide Web
 - Response Messages

Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error	server failed to fulfill an apparently valid request

Five types of HTTP result codes

Traditional Applications

- World Wide Web

- Uniform Resource Identifiers

- URLs:one type of *Uniform Resource Identifier (URI)*.
 - URI:character string that identifies a resource, and a resource can be anything that has identity, such as a document, an image, or a service.
 - format of URIs allows specialized kinds of resource identifiers to be incorporated into the URI space of identifiers.
 - first part of a URI: *scheme* that names a particular way of identifying a certain kind of resource, such as mailto for email addresses or file for file names.
 - second part of a URI:separated from the first part by a colon, is the *scheme-specific part*.

Traditional Applications

- World Wide Web
 - TCP Connections
 - Original version of HTTP (1.0) established a separate TCP connection for each data item retrieved from the server.
 - Retrieving a page with some text and a dozen icons or other small graphics would result in 13 separate TCP connections being established and closed

Traditional Applications

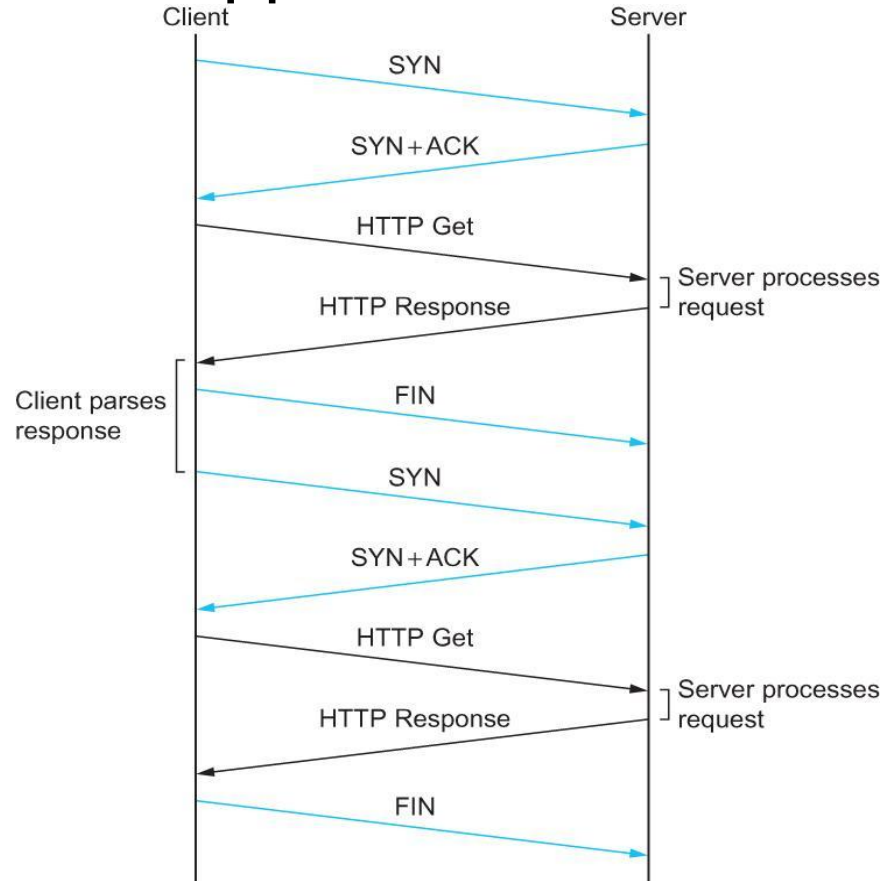
- World Wide Web

- TCP Connections

- HTTP version 1.1 introduced *persistent connections*— the client and server can exchange multiple request/response messages over the same TCP connection.
 - Persistent connections-advantages
 - Eliminate connection setup overhead, thereby reducing the load on the server, load on the network caused by the additional TCP packets, and the delay perceived by the user
 - As a client can send multiple request messages down a single TCP connection, TCP's congestion window mechanism is able to operate more efficiently

Traditional Applications

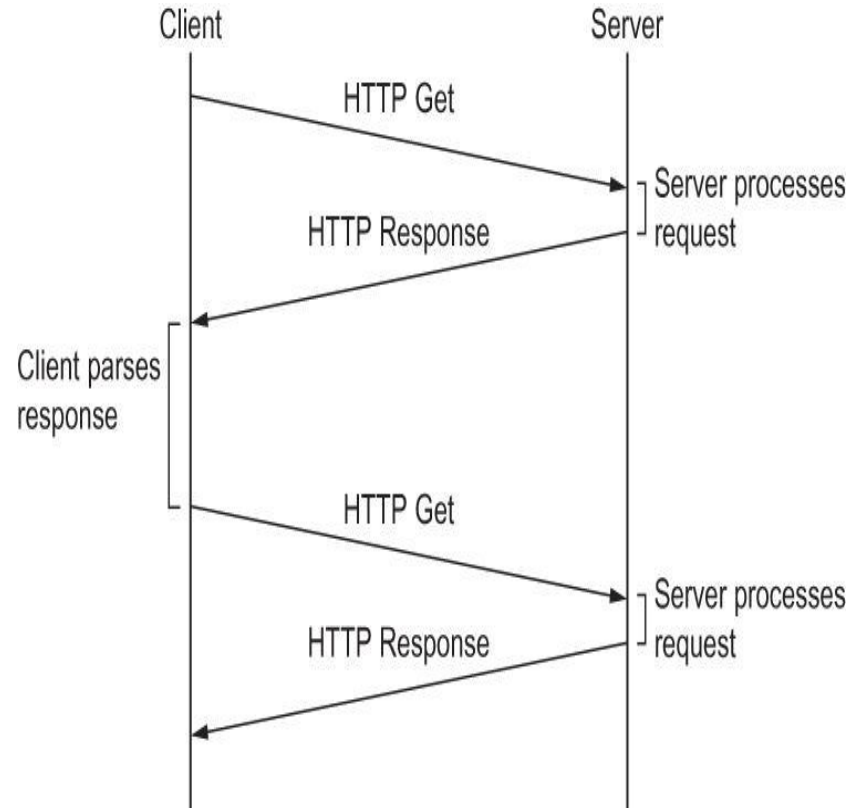
- World Wide Web
 - TCP Connections



HTTP 1.0 behavior

Traditional Applications

- World Wide Web
 - TCP Connections



HTTP 1.1 behavior with persistent connections

Traditional Applications

- World Wide Web

- Caching

- Benefits

- From the client's perspective, a page can be retrieved from a nearby cache and displayed much more quickly than if it has to be fetched from across the world.
 - From the server's perspective, having a cache intercept and satisfy a request reduces the load on the server

Traditional Applications

- World Wide Web

- Caching

- Can be implemented in many different places Eg. a user's browser can cache recently accessed pages, and display the cached copy if the user visits the same page again.
 - A site can support a single site-wide cache that allow users to take advantage of pages previously downloaded by other users
 - Closer to the middle of the Internet, ISPs can cache pages
 - Users within the site most likely know what machine is caching pages on behalf of the site, and configure their browsers to connect directly to the caching host and the node is called a *proxy*

Electronic Mail

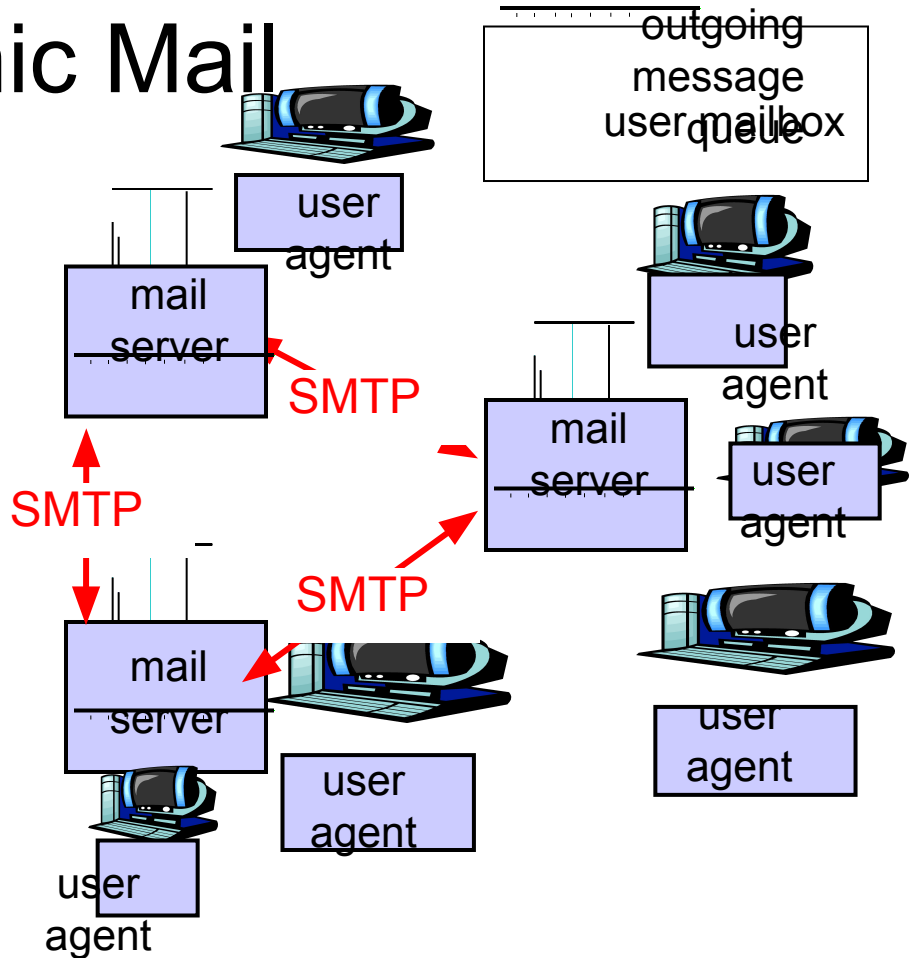


Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

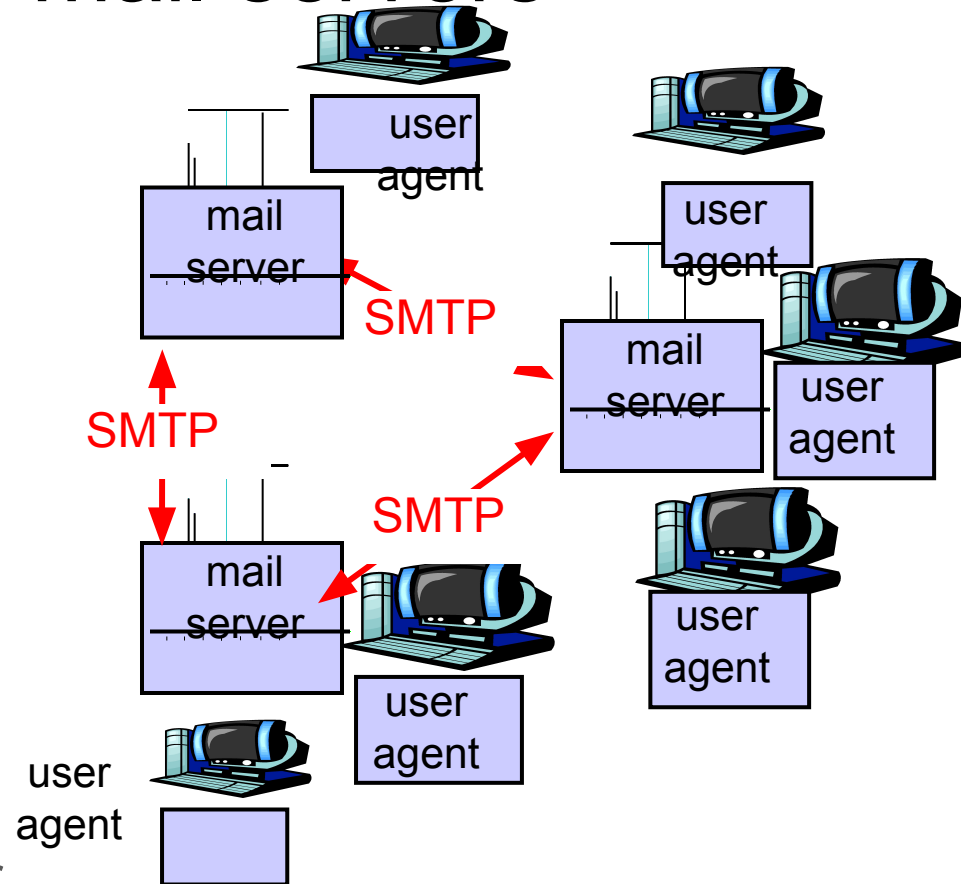
- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, pine, mutt, Thunderbird
- outgoing, incoming messages stored on server



Electronic Mail: mail servers

Mail Servers

- **mailbox** contains incoming messages for user
- **message queue** of outgoing (to be sent) mail messages
- **SMTP protocol** between mail servers to send email messages
 - client: sending mail server
 - server: receiving mail server

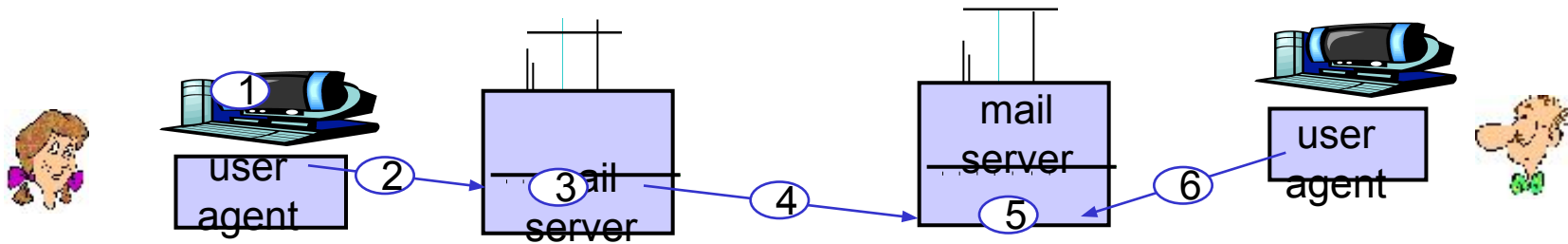


Electronic Mail: SMTP [RFC 2821]

- Uses TCP to reliably transfer email message from client to server, port 25
- Direct transfer: sending server to receiving server
- 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

Scenario: Alice sends message to Bob

- 1) Alice uses UA to compose message and "to" `bob@some school.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



Sample SMTP interaction

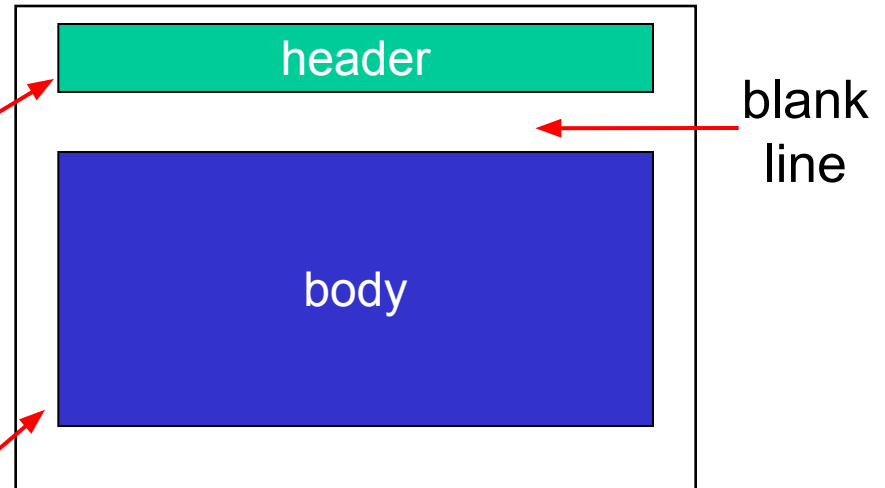
```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: From: Alice <alice@crepes.fr>
C: To: Bob <bob@hamburger.edu>
C: Subject: Toppings
C:
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Mail message format

SMTP: protocol for exchanging email
msgs

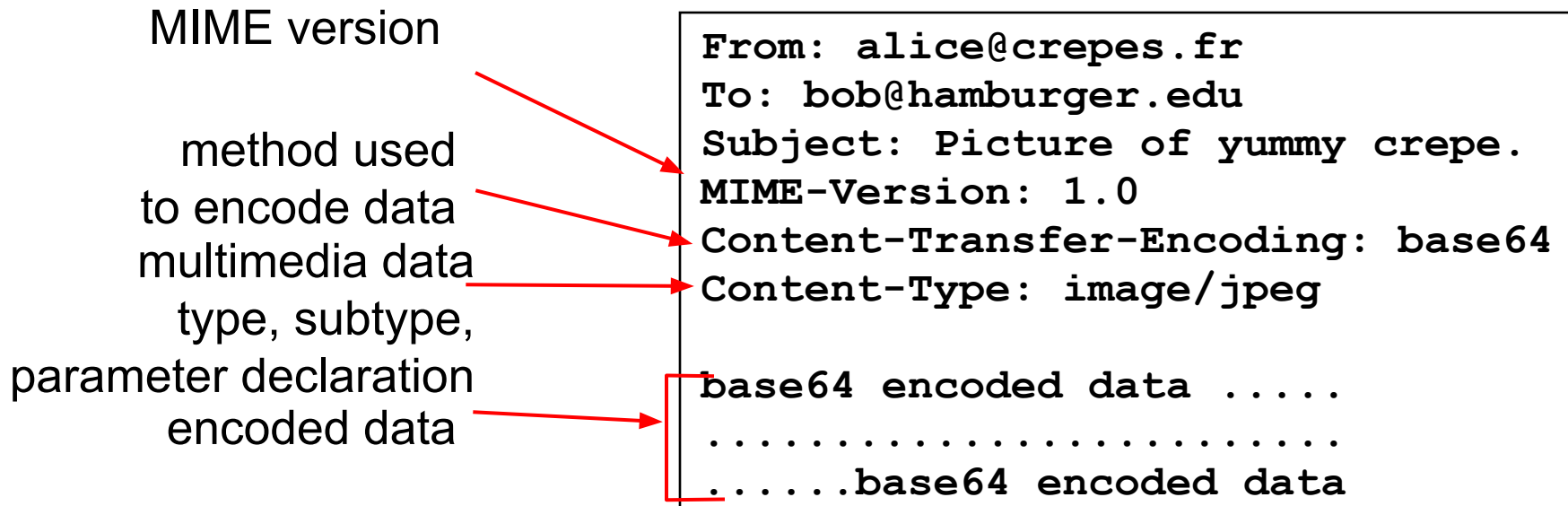
RFC 822: standard for text message
format:

- header lines, e.g.,
 - To:
 - From:
 - Subject:*different from SMTP commands!*
- body
 - the “message”, ASCII characters only

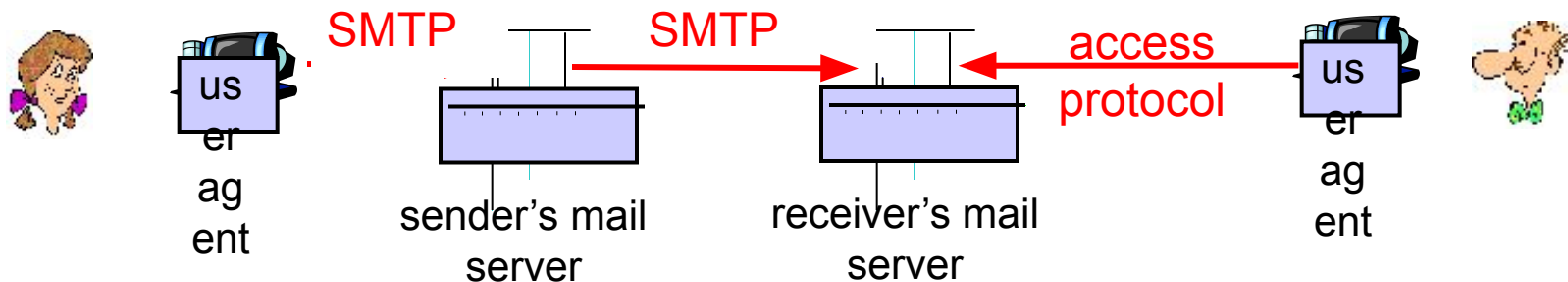


Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type



Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
 - POP: Post Office Protocol [RFC 1939]
 - authorization (agent <-->server) and download
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - more features (more complex)
 - manipulation of stored msgs on server
 - HTTP: Gmail , Yahoo! Mail, etc.

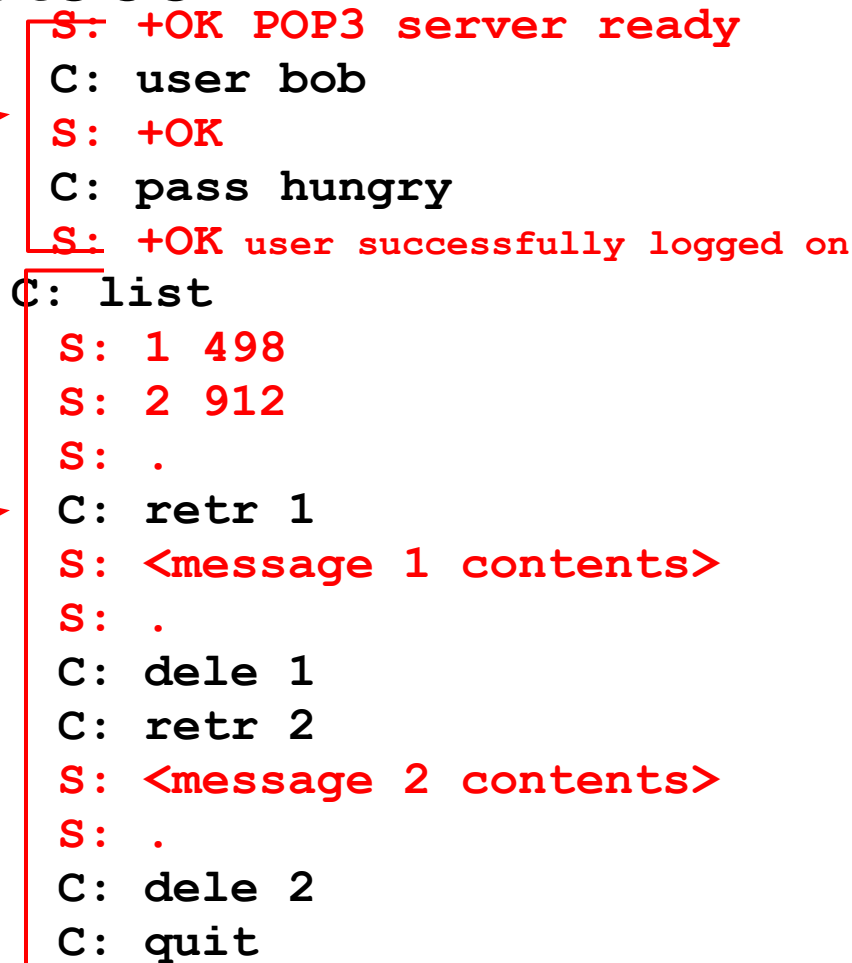
POP3 protocol

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**



A diagram illustrating a POP3 protocol session. A vertical red line separates the client commands from the server responses. Red arrows point from the 'authorization phase' and 'transaction phase' headers to their respective parts of the session. The session starts with the server sending '+OK POP3 server ready'. The client then sends 'user bob', and the server responds with '+OK'. The client sends 'pass hungry', and the server responds with '+OK user successfully logged on'. The client then sends 'list', and the server responds with '1 498', '2 912', and a period. The client sends 'retr 1', and the server responds with '<message 1 contents>' and a period. The client sends 'dele 1', and the server responds with a period. The client sends 'retr 2', and the server responds with '<message 2 contents>' and a period. The client sends 'dele 2', and the server responds with a period. Finally, the client sends 'quit', and the server responds with a period.

```
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 2 contents>
S: .
C: dele 2
C: quit
```

POP3 (more) and IMAP

More about POP3

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

IMAP

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

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- “name”, e.g., www.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
 - note: core Internet function, implemented as application-layer protocol
 - complexity at network’s “edge”

DNS

DNS services

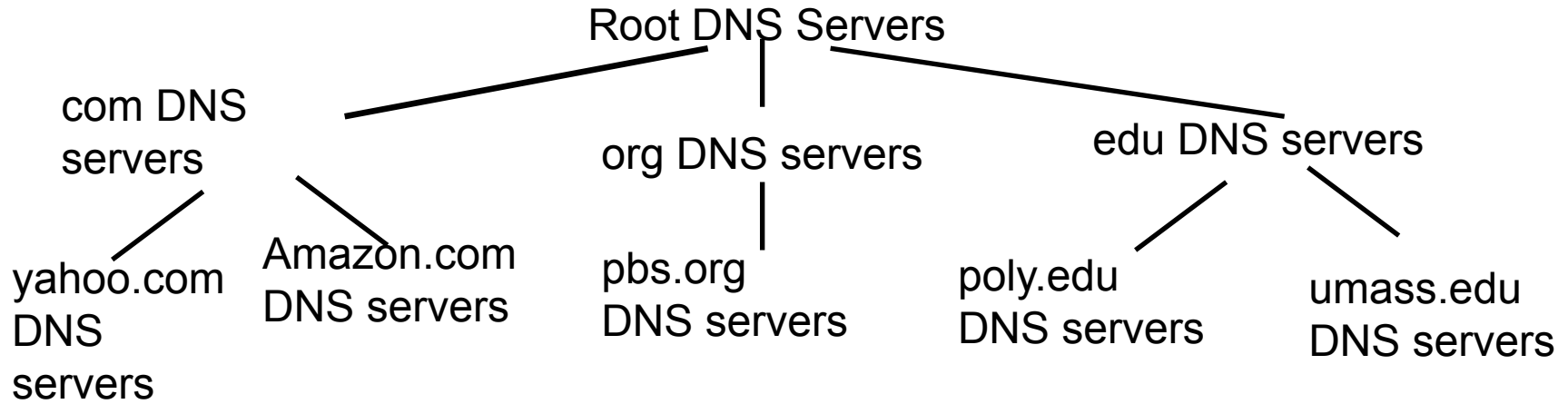
- Hostname to IP address translation
- Host aliasing
 - Canonical and alias names
- Mail server aliasing
- Load distribution
 - Replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Maintenance

Doesn't scale!

Distributed, Hierarchical Database



Client wants IP for www.amazon.com; 1st approx:

- 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 servers

- Contacted by local name server that can not resolve name
- Root name server:
 - contacts authoritative name server if name mapping not known
 - gets mapping
 - returns mapping to local name server



13 root name servers
worldwide

TLD and Authoritative Servers

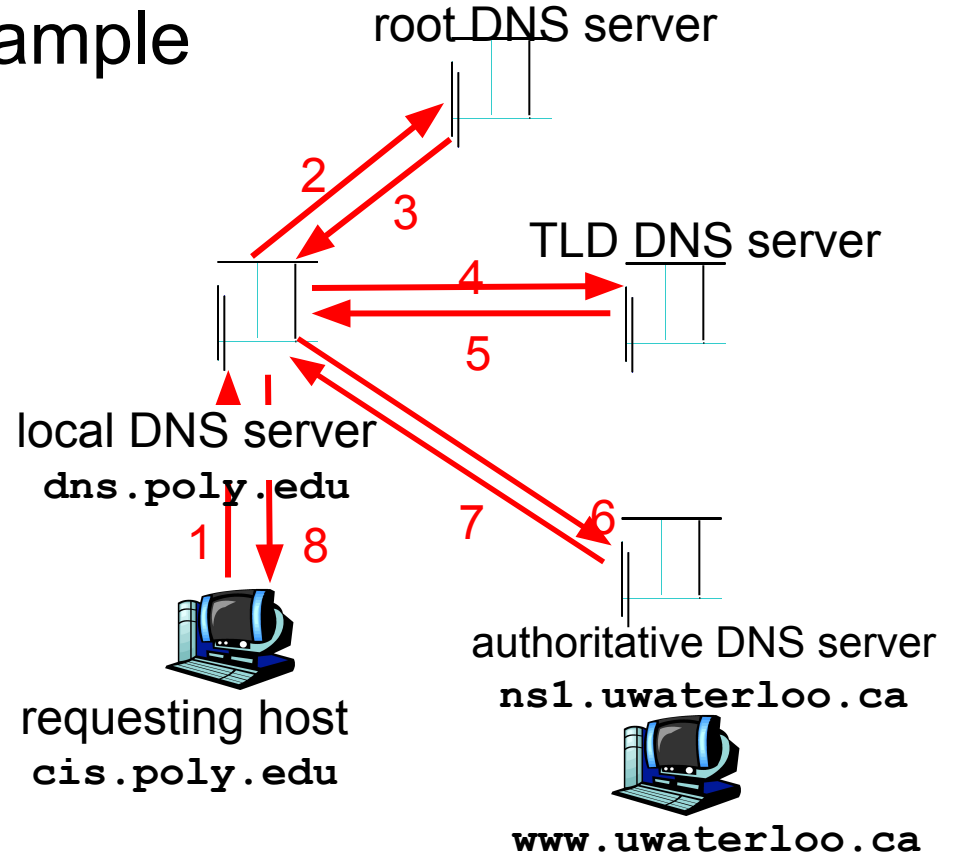
- **Top-level domain (TLD) servers:** responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
 - Network solutions maintains servers for com TLD
 - Educause for edu TLD
 - CIRA for ca TLD
- **Authoritative DNS servers:** organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web and mail).
 - Can be maintained by organization or service provider

Local Name Server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one.
 - Also called “default name server”
- When a host makes a DNS query, query is sent to its local DNS server
 - Acts as a proxy, forwards query into hierarchy.

Example

- Host at cis.poly.edu wants IP address for www.uwaterloo.ca



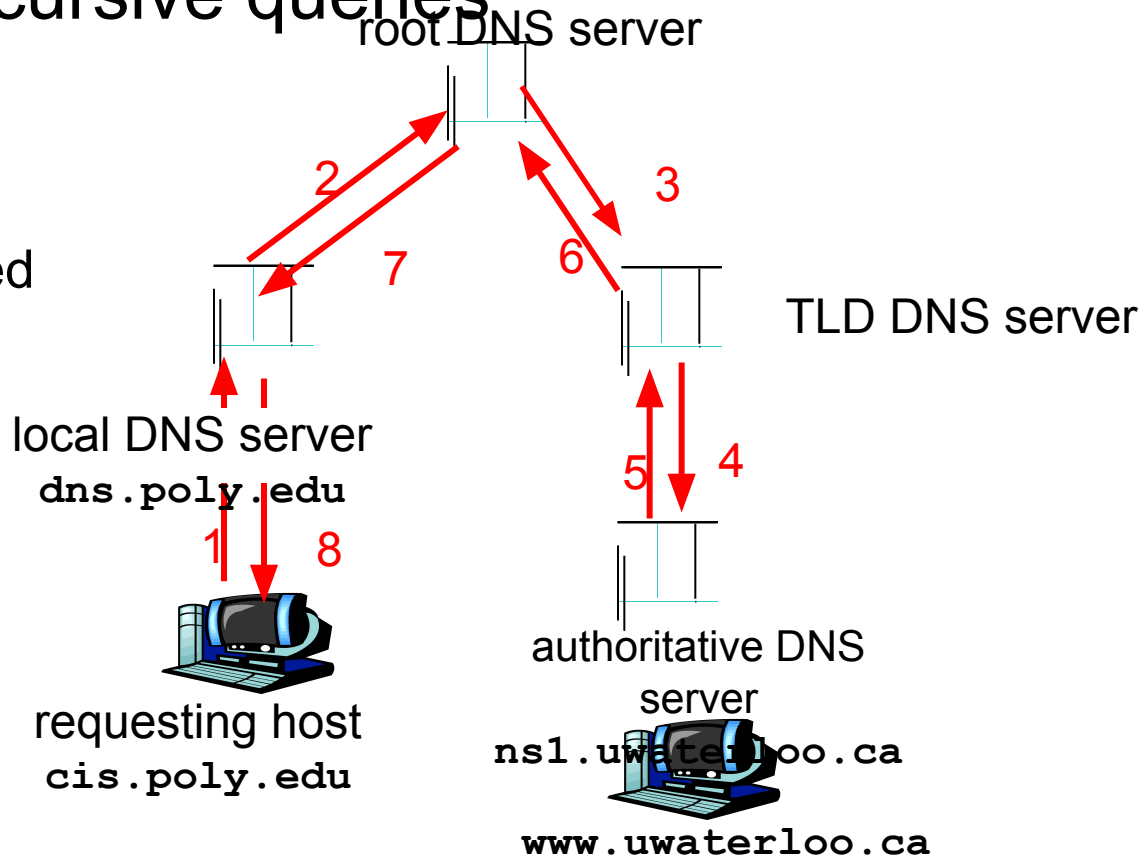
Recursive queries

recursive query:

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

iterated query:

- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”



DNS: caching and updating records

- Once (any) name server learns mapping, it *caches* mapping
 - cache entries timeout (disappear) after some time
 - TLD servers typically cached in local name servers
 - Thus root name servers not often visited
- Update/notify mechanisms under design by IETF
 - RFC 2136
 - <http://www.ietf.org/html.charters/dnsind-charter.html>

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

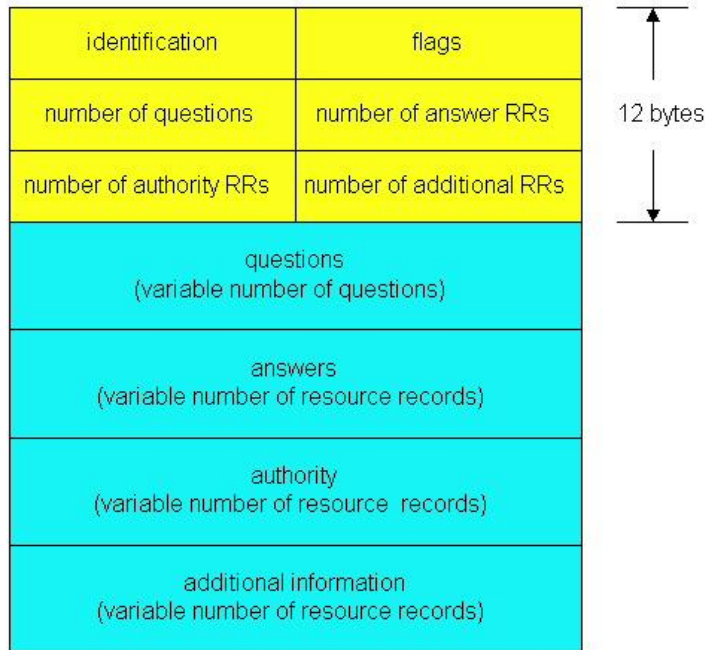
- ◆ **value** is name of mailserver associated with **name**

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

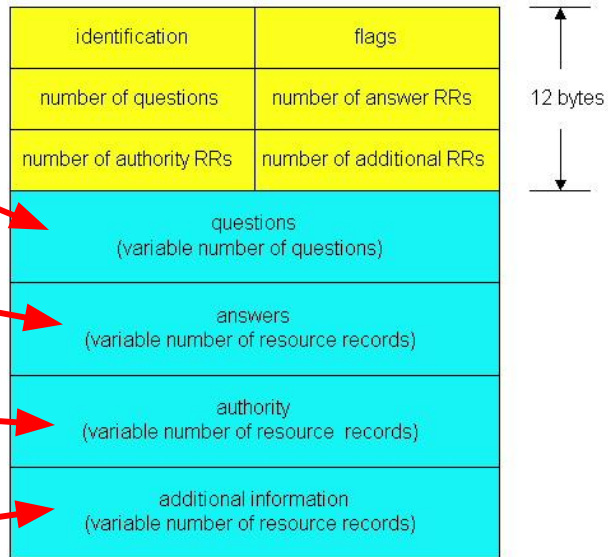


DNS protocol, messages

Name, type fields
for a query

RRs in response
to query
records for
authoritative servers

additional “helpful”
info that may be used



Inserting records into DNS

- Example: just created startup “Network Utopia”
- Register name networkutopia.com at a **registrar** (e.g., Network Solutions)
 - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
 - Registrar inserts two RRs into the com TLD server:

```
(networkutopia.com, dns1.networkutopia.com, NS)  
(dns1.networkutopia.com, 212.212.212.1, A)
```

- Put in authoritative server Type A record for `www.networkutopia.com` and Type MX record for `networkutopia.com`
- **How do people get the IP address of your Web site?**