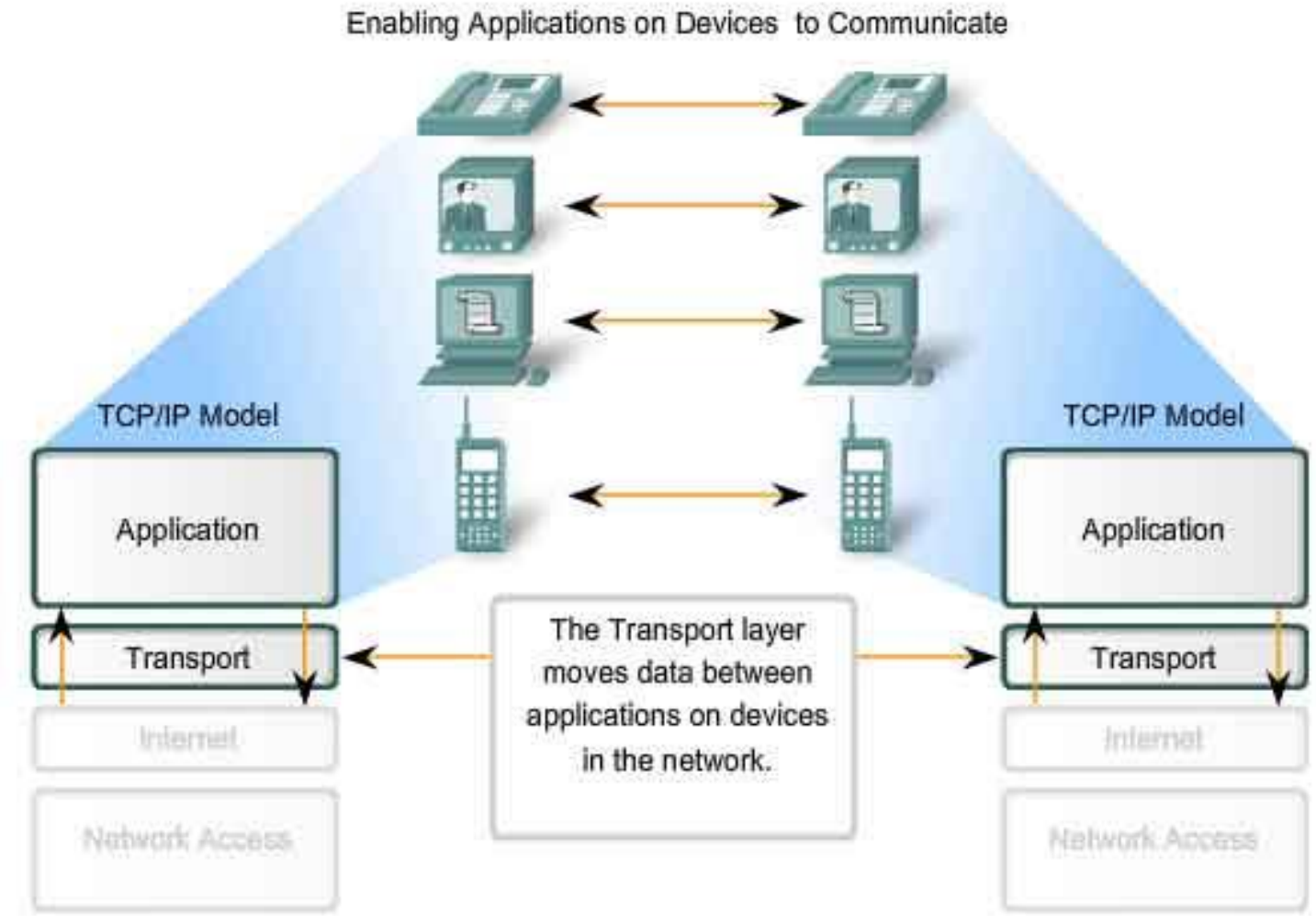
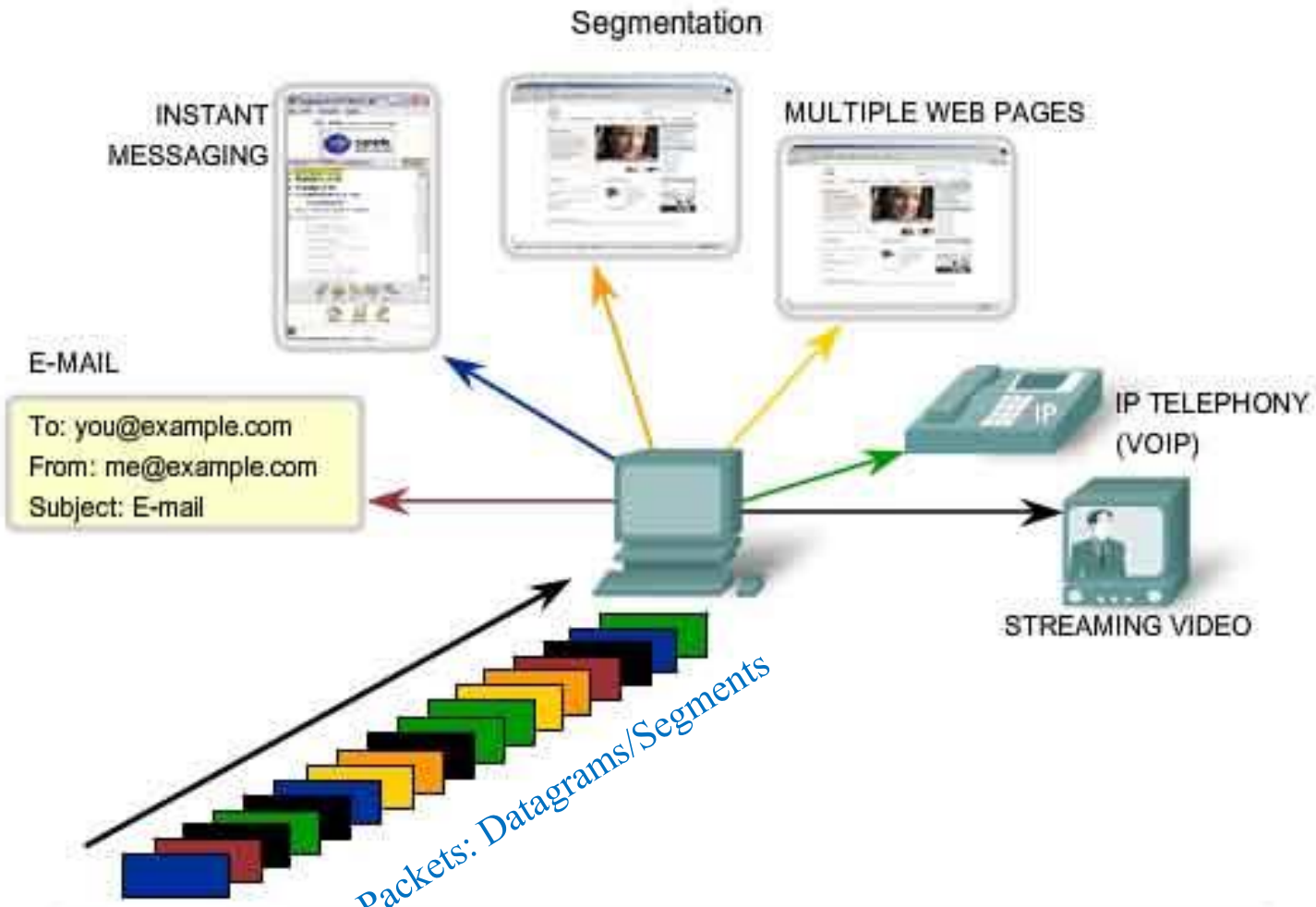
Transportation Protocols: UDP, TCP & RTP

- Transportation Functions
- UDP (User Datagram Protocol)
- Port Number to Identify Different Applications
- Server and Client as well as Port
- TCP (Transmission Control Protocol)
  - TCP Segment Format and Delivery
  - TCP Reliability Control
  - TCP Flow Control
  - TCP Congestion Control
  - TCP Connection Control
- Comparison between UDP and TCP
- RTP (Realtime Transport Protocol)

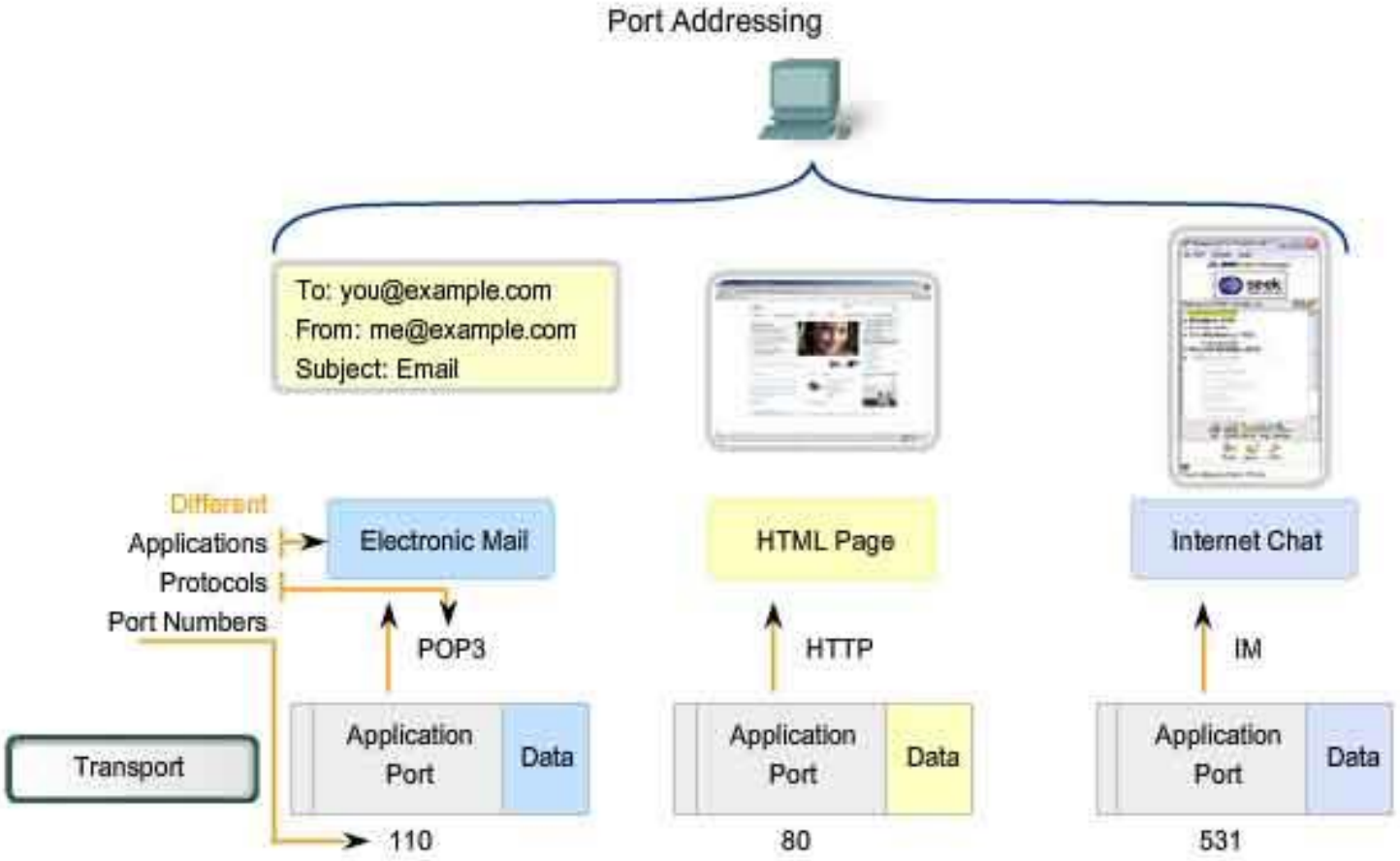
Transport: Application-to-Application



Transport: Handle Different Applications

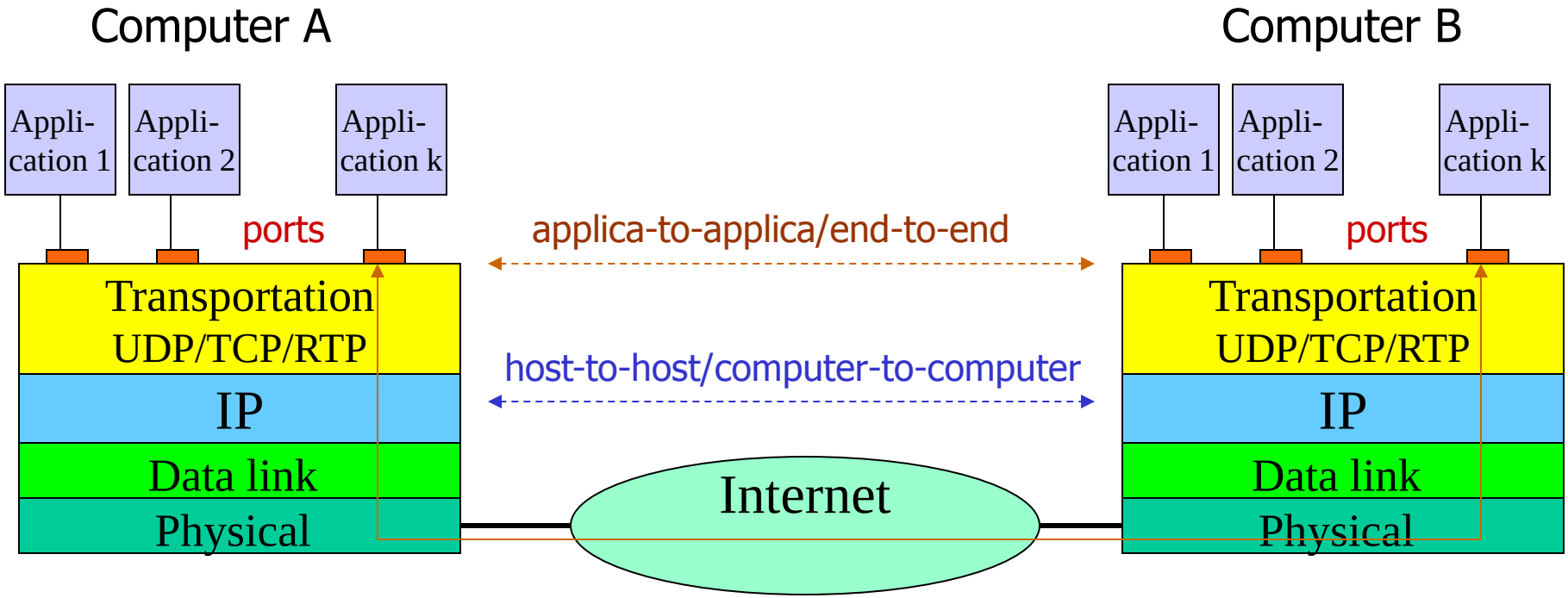


Transport: Applications in Different Ports



Data for different applications is directed to the correct application because each application has a unique port number.

Transportation Functions

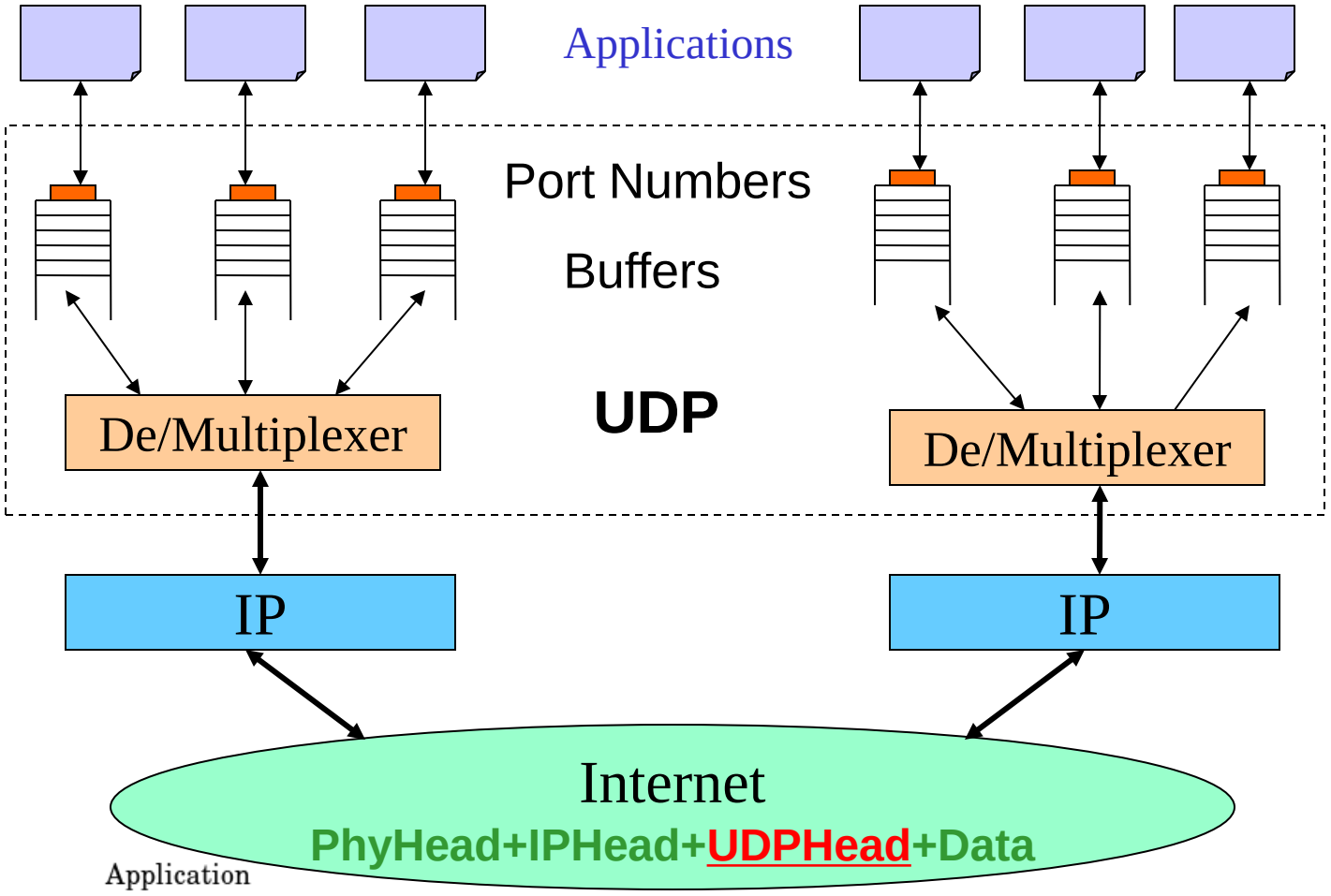


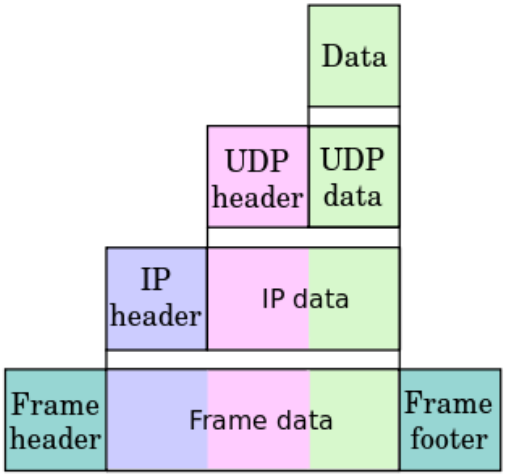
- IP functions
  - Computer-to-computer connectionless communication using IP address
  - Unreliable datagram service: corrupted, lost, duplicated, disordered
- Transportation functions
  - Application-to-application communication using port number, application address
  - Support multiple applications simultaneously using multiplexing
  - Optional functions: reliability, flow control, congestion control, connection service

Lecture 11

UDP

David P. Reed
Designer of UDP

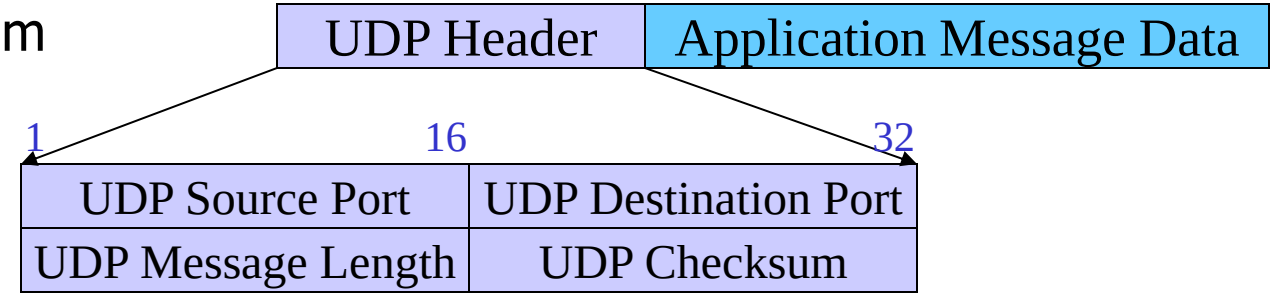




- UDP (User Datagram Protocol)
  - Identifies different applications using port numbers
  - Connectionless, datagrams are delivered independently
  - Unreliable delivery: loss, corruption, duplication, disorder

UDP Header Format and Port Number

UDP datagram



- Source
  - application sending message
- destination
  - application receiving data
- Port number:
  - 2 byte integer from 0~65535
    - 0 to 1023: reserved, well-known
    - 1024 to 49151: registered ports
    - 49152 to 65535: dynamic ports
- Source & destination port numbers may be different

http://en.wikipedia.org/wiki/User_Datagram_Protocol

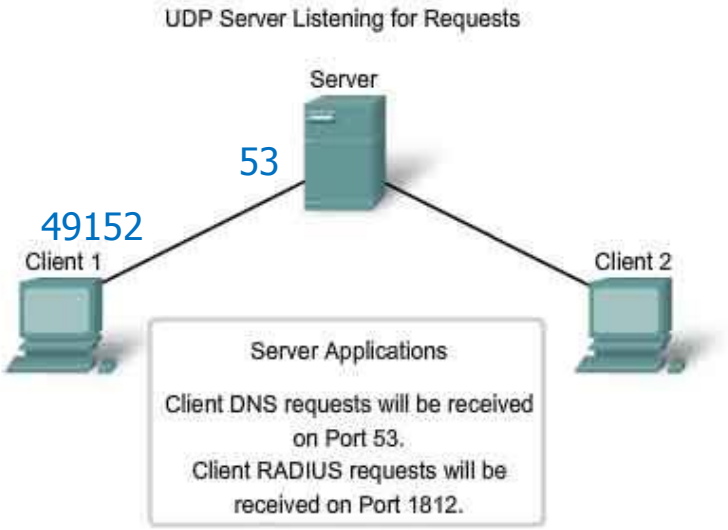
Port	Name	Description
7	echo	Echo input back to sender
9	discard	Discard input
11	systat	System statistics
13	daytime	Time of day (ASCII)
17	quote	Quote of the day
19	chargen	Character generator
37	time	System time (seconds since 1970)
53	domain	Domain Name Server (DNS)
69	tftp	Trivial File Transfer Protocol (TFTP)
123	ntp	Network Time Protocol (NTP)
161	snmp	Simple Network Management Protocol

Server, Client and Port Example

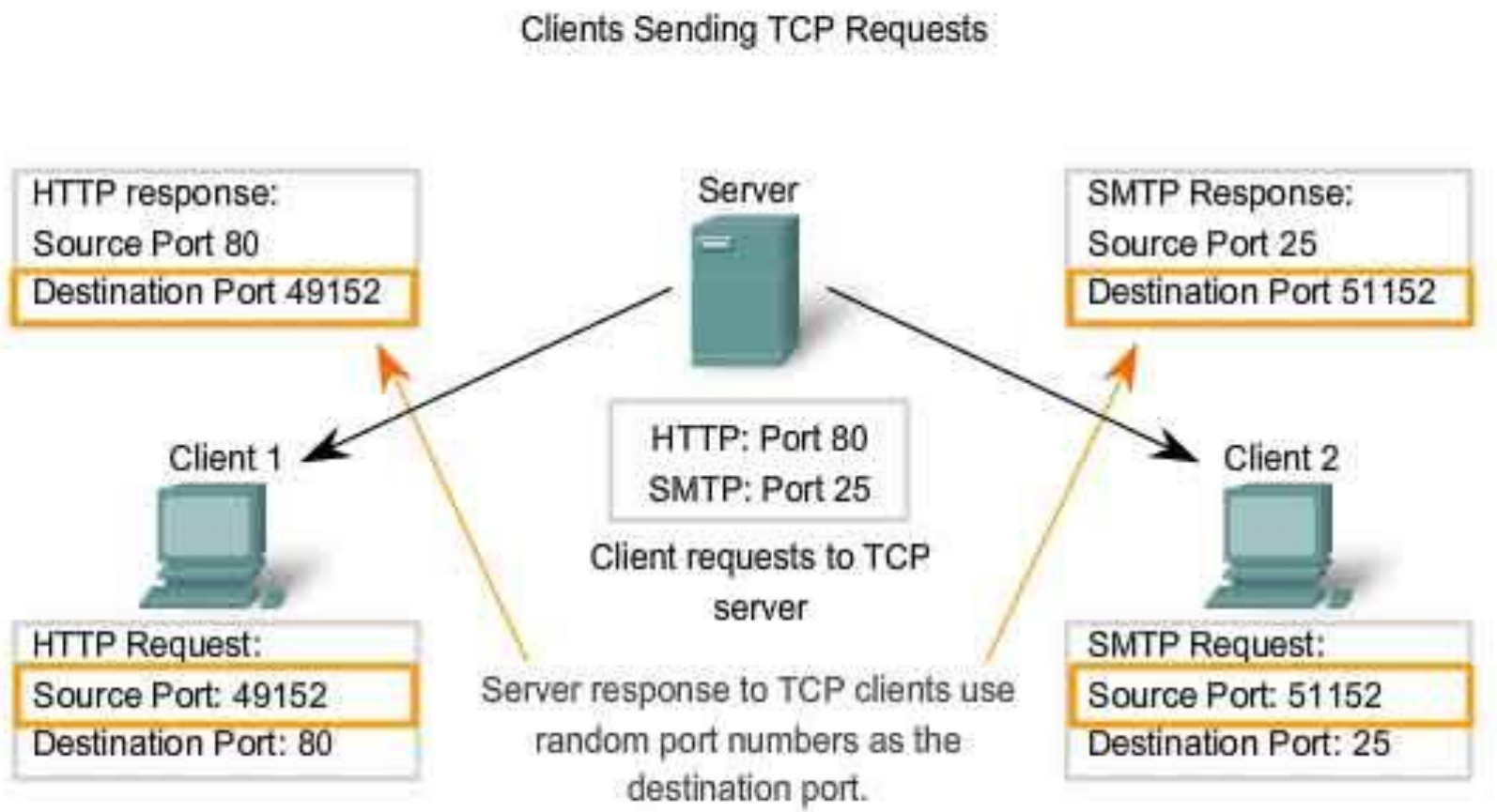
- Server
  - A program in a remote or local machine
  - Executed first and passively waits for connection from clients
  - Accepts request from a client and replies to the client
- Client
  - A program in a local machine
  - executed late and actively initiates connection to a server
  - Sends request to a server and accepts reply from the server

1	16	32
UDP Source Port	UDP Destination Port	
UDP Message Length	UDP Checksum	

- DNS (domain name system) Example:
  - DNS server port number: 53 (fixed)
  - DNS client port number: 49152 (changeable)
  - DNS client connects to DNS server, requests IP address
  - UDP datagram from client to server
    - source port number: 49152
    - destination port number: 53
  - Server replies IP address in a UDP datagram to client
  - UDP datagram from server to client
    - source port number: 53
    - destination port number: 49152



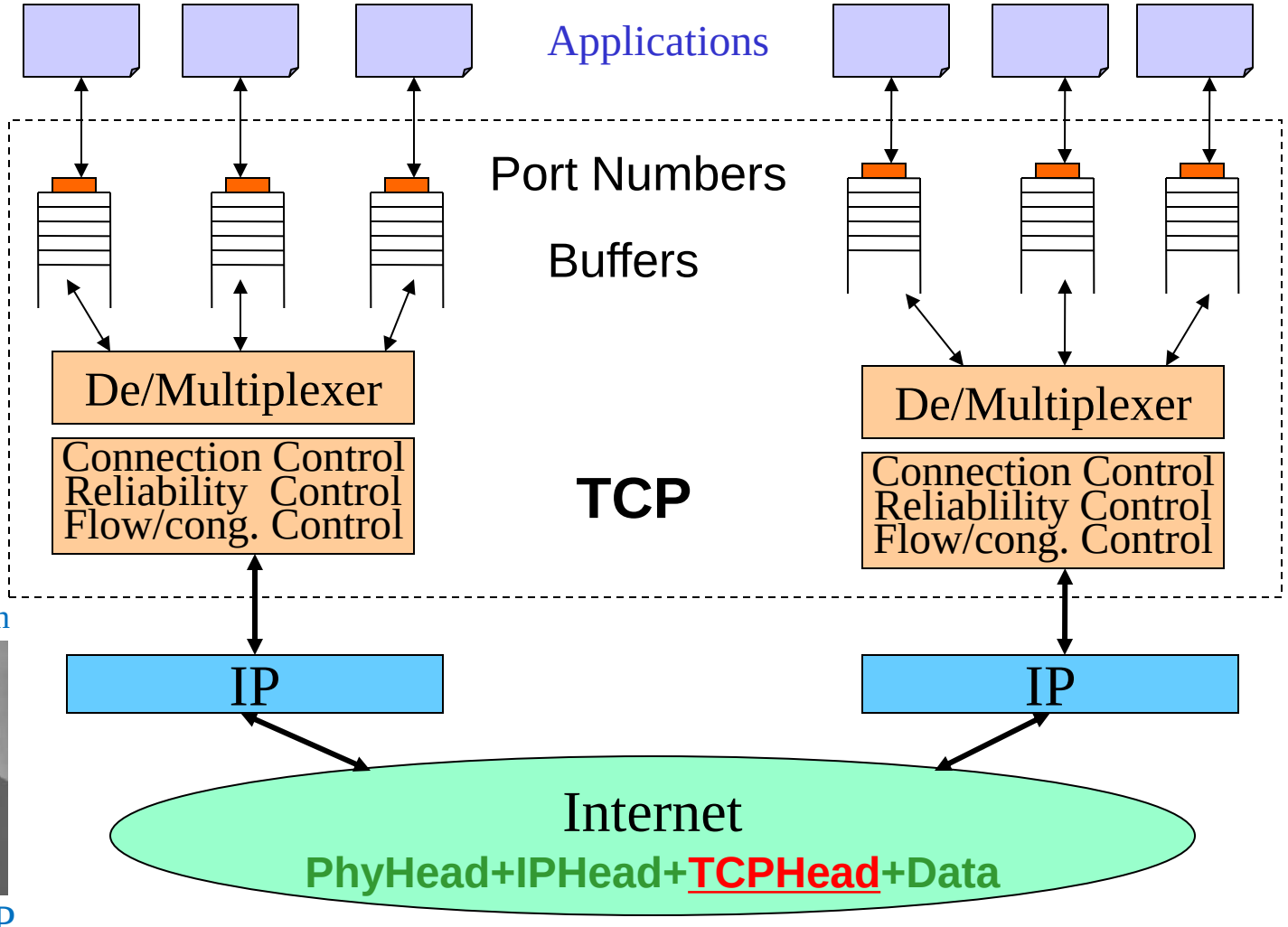
Ports for HTTP/SMTP Communications



Use the "netstat" tool to check port numbers in your computer

Lecture 11

TCP

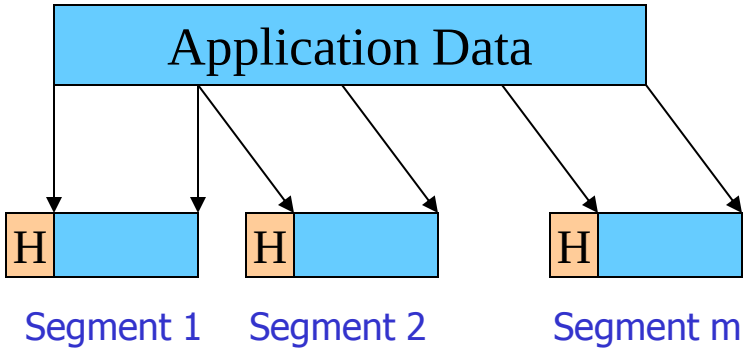


Vinton Cerf Robert Kahn

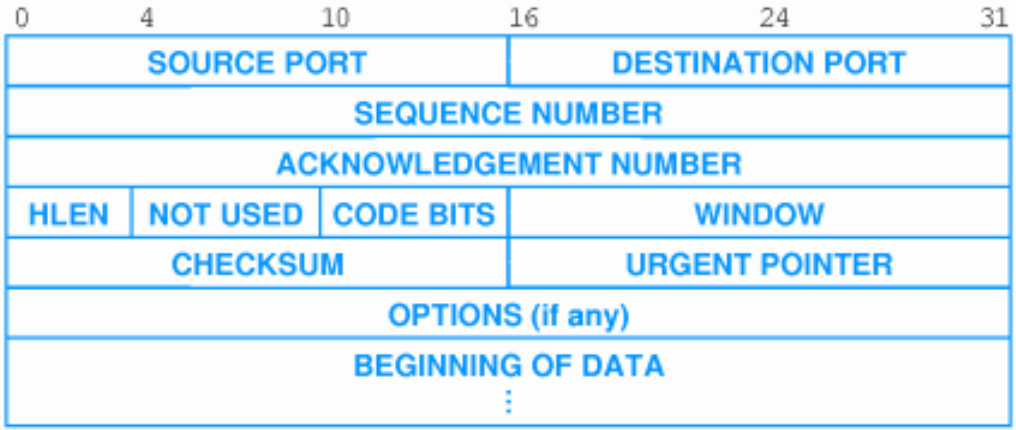
Designers of TCP/IP

- TCP (Transmission Control Protocol)
  - Identifies different applications using port numbers like UDP
  - Connection oriented service: connection establishment and termination
  - Full duplex and stream connection
  - Reliable delivery: no packet loss, error, duplication, disorder

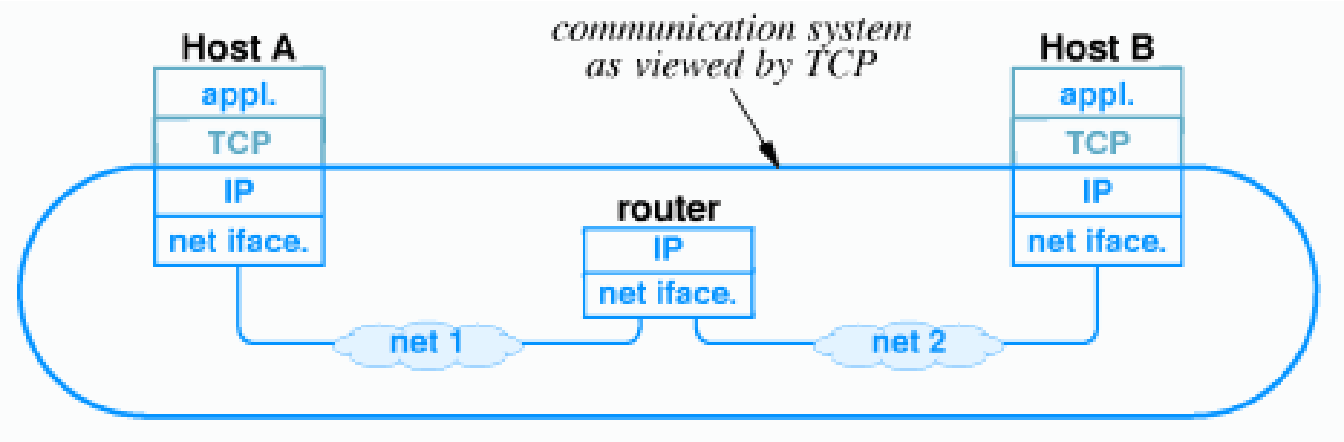
TCP Segment Format and Delivery



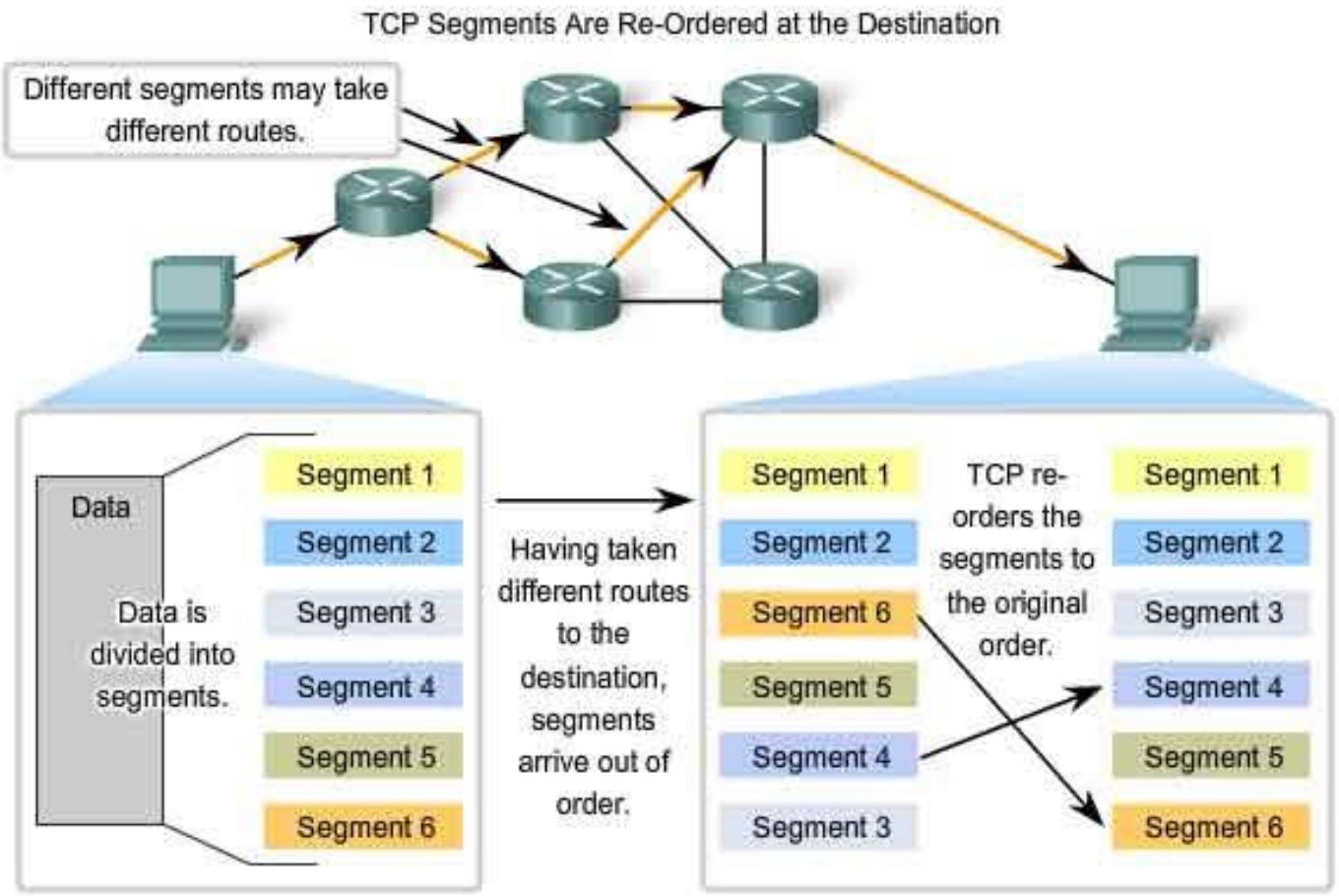
TCP Segments



- Port number: 2 byte (0~65535), FTP: 20, Telnet: 23, HTTP: 80, Servlet: 8080
- Sequence number: unique ID number to identify order/location of a segment
- Codes: signal types of TCP segment: 010000 (ACK), 000010 (SYN), 000001 (FIN)
- TCP uses IP for data delivery like UDP, routers only forward IP datagram



TCP – Using Seq_Number to Handle Disorder



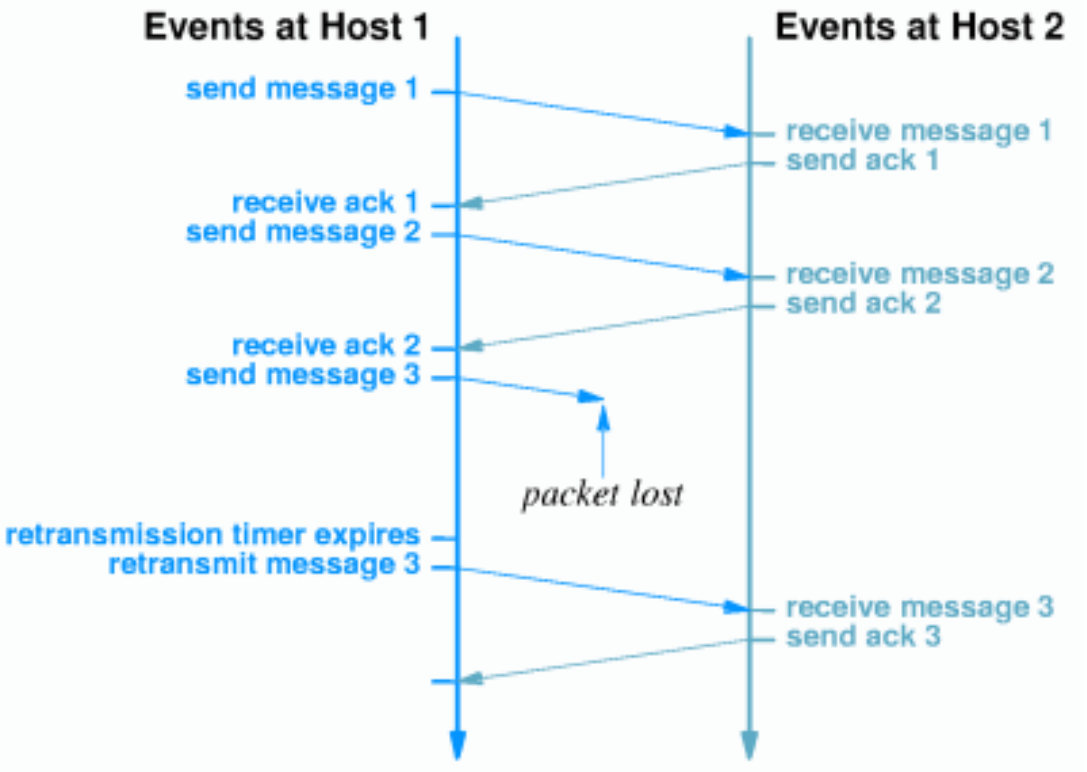
TCP Reliable Data Transmission

- IP datagram can be corrupted, lost, duplicated and disordered
- Duplicated and disordered segments are overcome by unique sequence number

- Corrupted and lost segments are overcome using acknowledge and retransmission technique

- Positive acknowledgement
  - Receiver Crns a short message when data arrives
  - Called acknowledgement
  - May acknowledge multiple segments
- Retransmission
  - Senders starts timer whenever a message is transmitted
  - If timer expires before acknowledge arrives, sender retransmits message

  How to set timeout ??

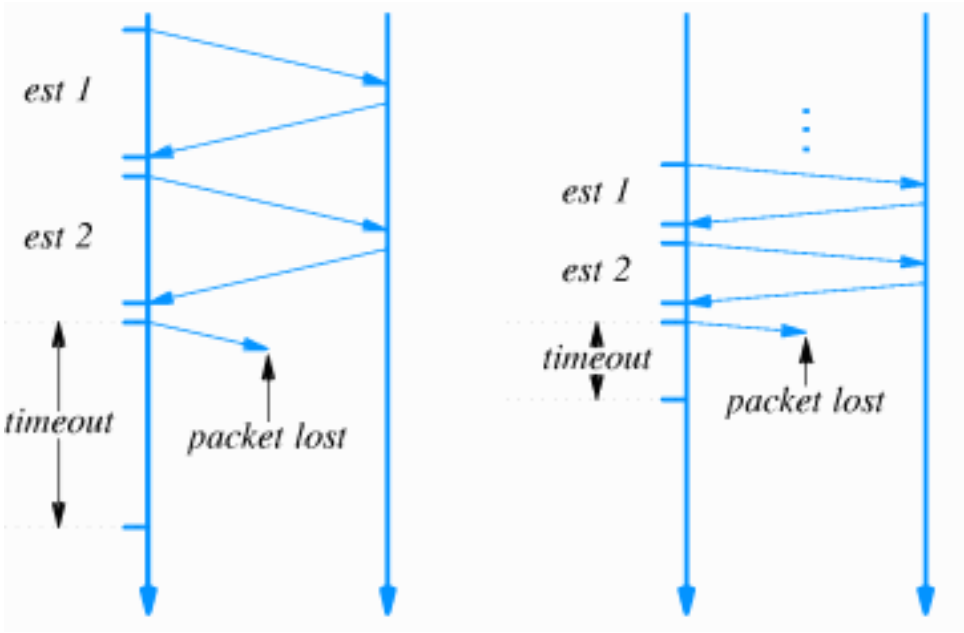


Setting Appropriate Value of Timeout

- Inappropriate timeout can cause poor performance:
  - Too long: sender waits longer than necessary before re-transmitting
  - Too short: sender generates unnecessary traffic
- Timeout must be different for each connection and set dynamically
  - Host on same LAN should have shorter timeout than host 20 hops away
  - Delivery time across internet may change over time; timeout must accommodate changes

- RTO (Retransmission TimeOut)
- RTT (Round Trip Time)
  = sending_time – Ack_receiving_time
- Estimate of RTT on each connection
- Estimate of RTT change
- RTO >~ RTT
- Called adaptive retransmission

- Key to TCP's success !

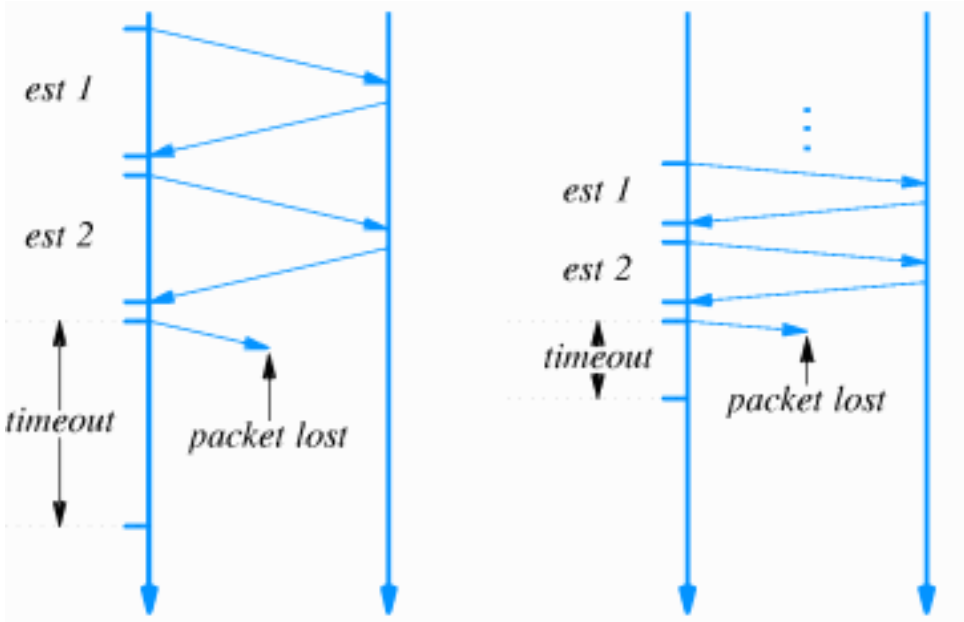


Adaptive Retransmission Algorithm

- Timeout should be based on *round trip time* (RTT)
- Sender can't know RTT of any packet before transmission
- Sender picks *retransmission timeout* (RTO) based on *previous* RTTs
- Specific method is call *adaptive retransmission algorithm*

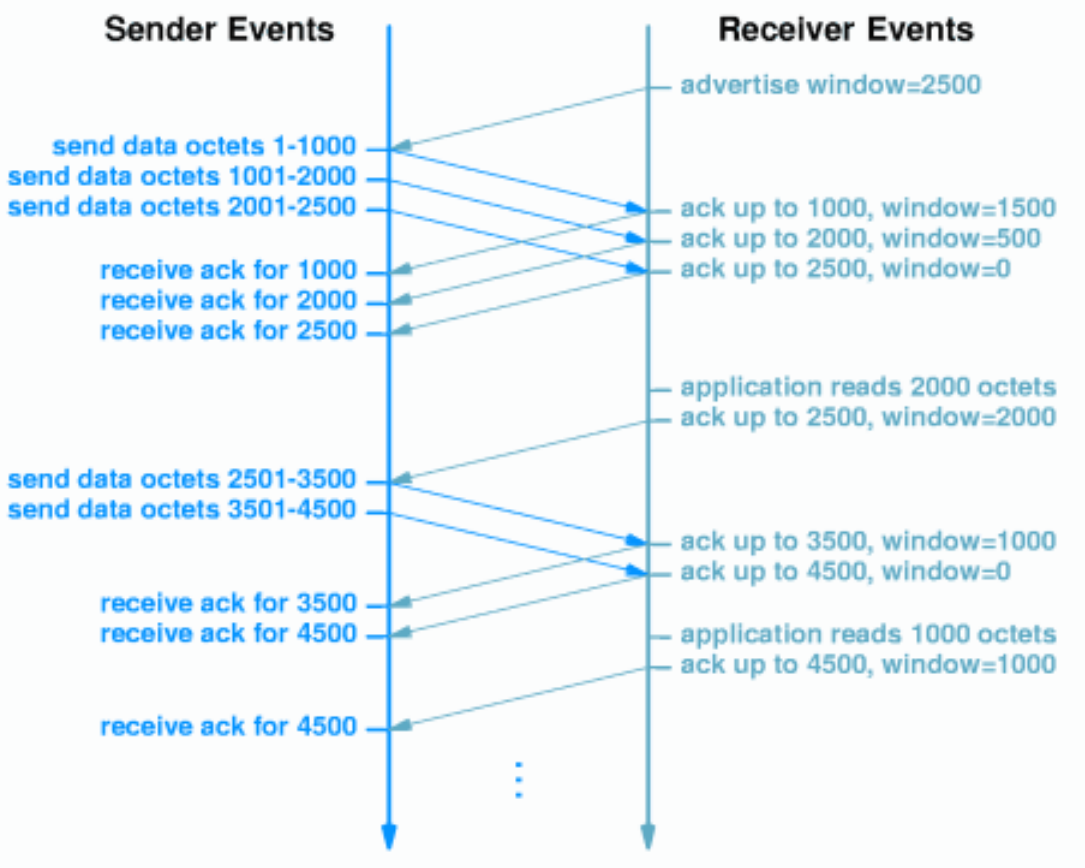
- Weighted average for RTT:
  RTT(n+1)= α RTT(n) + (1-α)RTTn
  where RTTn is RTT of the nth segment
- Computation of RTO
  RTO(n+1)= β RTT(n+1)
  where 0< α <1 and 1.3< β <2.0
  RTO(n+1) is RTO for (n+1)th segment

- Initial value RTT(0)
  decided in connection establishment
- When packet lost → no RTTn
  RTO(n+1) = 2 RTO(n)
  RTT(n+1)=RTT(n)
  - Proposed by Karn
  - called exponential backoff



TCP Flow Control

- Flow control
  - overcome data overrun when sending data fast than receiver can processed
- TCP uses slide window control
- Receiver
  - informs available buffer space – window
  - Each acknowledgement carries new window information called window advertisement
  - Maximum Window: buffer size
  - Minimum window: zero (closed window)
- Sender
  - Can send data up to entire window before ack arrives
- Interpretation
  I have received up through X, and can take Y more octets/bytes



TCP Slide Widow Animation
http://histrory.visualland.net/tcp_swnd.html

TCP Congestion Control

- Excessive traffic can cause packet loss
  - Transport protocols respond with retransmission
  - Excessive retransmission can cause *congestion collapse*
- TCP interprets packet loss as an indicator of congestion
- Sender uses TCP *congestion control* and slows transmission of packets
  - Sends single packet
  - If acknowledgment returns without loss, sends two packets
  - When TCP sends one-half window size, rate of increase slows

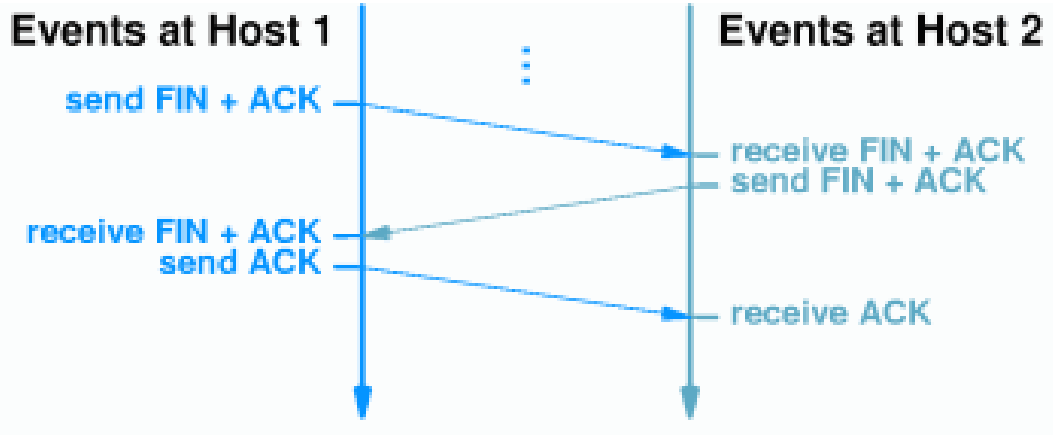
http://en.wikipedia.org/wiki/Transmission_Control_Protocol

TCP Connection Control: Startup & Shutdown

- TCP is connection-oriented.
  - A connection must be established, called startup, before data communications
  - A connection must be terminated, called shutdown, if no data to be sent.
- Connect start must be reliable and shutdown must graceful
- Difficult because segments can be
  - lost
  - duplicated
  - delayed
  - delivered out of order
  - Either side can crash
  - Either side can reboot
- Needed to avoid duplicated "shutdown" message from affecting later connection

TCP Three-Way Handshake

- TCP uses three-message exchanges called 3-way handshake to startup/shutdown
- Synchronization segment (SYN) to describe messages in startup connection
- Finish segment (FIN) to describe message in shutdown connection
- Host 1 sends segment with SYN/FIN bit set and a random sequence number
- Host 2 responds with segment with SYN/FIN bit set, acknowledgment to Host 1, and the random sequence number
- Host 1 responds with acknowledgment
- Remember TCP segment will retransmit lost, duplicated, delayed
- 3-way handshake ensures unambiguous, reliable, graceful startup/shutdown despite packet loss, duplication and delay



Animation

Animation

Comparison between UDP and TCP

- **UDP**
  - **Connectionless** service, UDP datagrams are delivered independently
  - **Unreliable** delivery: packet loss, corruption, duplication, disorder
  - Relatively **fast** as compared with TCP
  - Simple request-response communication without internal flow and error control
  - Some data (e.g., audio/video) with a certain toleration of errors
  - Multicasting and broadcasting

- **TCP**
  - **Connection-oriented**: connection establishment & termination
  - Full duplex and stream connection
  - **Reliable** delivery: no loss, error, duplication, disorder
  - Connection overhead
  - Relatively **slow**: retransmission flow control
  - Heavily used like ftp, email, Web service, …

TCP	UDP
Reliable	Unreliable
Connection-oriented	Connectionless
Segment retransmission and flow control through windowing	No windowing or retransmission
Segment sequencing	No sequencing
Acknowledge segments	No acknowledgement

UPD Video http://www.youtube.com/watch?v=NigkgDu452k

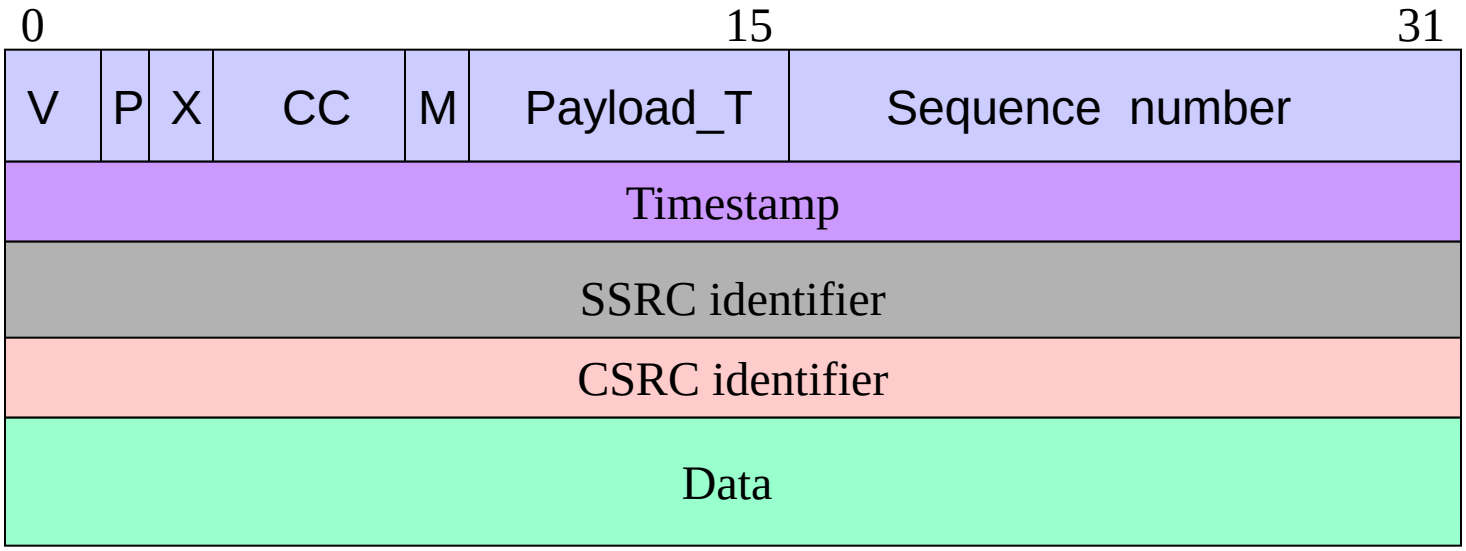
TCP Animation http://cyberdig.blogspot.jp/2012/05/animation-tcp-vis-vas-udp.html

RTP (Realtime Transport Protocol)

- RTP (RFC 1889) provides end-to-end transport functions for applications that require real time transmissions, such as audio and video over unicast or multicast packet network services
- RTP normally runs on top of UDP but not limits to this
- RTP does not provide QoS guarantees
- RTP deals with jitter, loss, timing recovery and inter-media synchronization
- RTP is often used together with RTP control protocol (RTCP) which monitors the transmission quality and conveys information about participants
- RTP is not implemented as a separated layer, but can be incorporated into the application processing
  → JMF (Java Media Framework) API

- Real time play music or watch movie (unicast)
- Internet telephony (unicast)
- Audio conference (multicast)
- Audiovisual conference (multicast)
- Note 1: a pair of consecutive port numbers for one medium
  - medium (audio/video) stream
  - RTCP stream
- Note 2: audio and video use different port pairs because of their different features

RTP Message Format



Version (V, 2B): RTP version (2 in current)
Padding (P, 1B): optional bytes followed the payload
Extension (X, 1B): optional header field
CSRC count (CC, 4B): Numbers of CSRC identifiers
Marker (M, 1B): mark significant event such as video frame
Payload type (P, 7B): format of RTP data
Sequence number (16B): detect packet order and loss
Timestamp (16B): sampling instant for synchronization
SSRC (16B): distinguish synchronization sources in a session
CSRC list (16B): specify contributing sources for the payload

http://en.wikipedia.org/wiki/Real-time_Transport_Protocol

Exercise 11

1. Explain similarities and differences between UDP and TCP protocols

2. The following is a TCP header in hexadecimal format.
   05320017 00000001 00000000 501007FF 00000000
   a. What are the source and destination port numbers?
   b. What is the sequence number?
   c. What is the acknowledgement number?
   d. What is the length of the header?
   e. What is the type of the segment?
   f. What is the window size?
3. A computer uses TCP to send a data to the other computer. The data is 100 bytes. Calculate the efficiency (ratio of useful bytes to total bytes) first at the TCP level (no optional field), then at the IP level (no option field), and finally at Ethernet link layer (no option field), respectively.

4. Corrupted and lost segments are overcome using acknowledge and retransmission technique in TCP. In the adaptive retransmission technique, timeout setting is very important and use the algorithm $RTO(n+1)= \beta\ RTT(n+1)$ where $RTT(n+1)= \alpha\ RTT(n) + (1-\alpha)RTTn$. Suppose $\alpha = 0.8$, $\beta = 1.5$ and $RTT(0) = 500ms$, calculate $RTO(1) \sim RTO(5)$ for $RTT1=600ms$, $RTT2=400ms$, $RTT3$ (lost), and $RTT4=500ms$.

5. Suppose two programs use TCP to establish a connection, communicate, terminate the connection, and then open a new connection. Further suppose a *FIN* message sent to shut down the first connection is duplicated and delayed until the second connection has nee established. If a copy of the old *FIN* is delivered, will TCP terminate the new connection? Why?