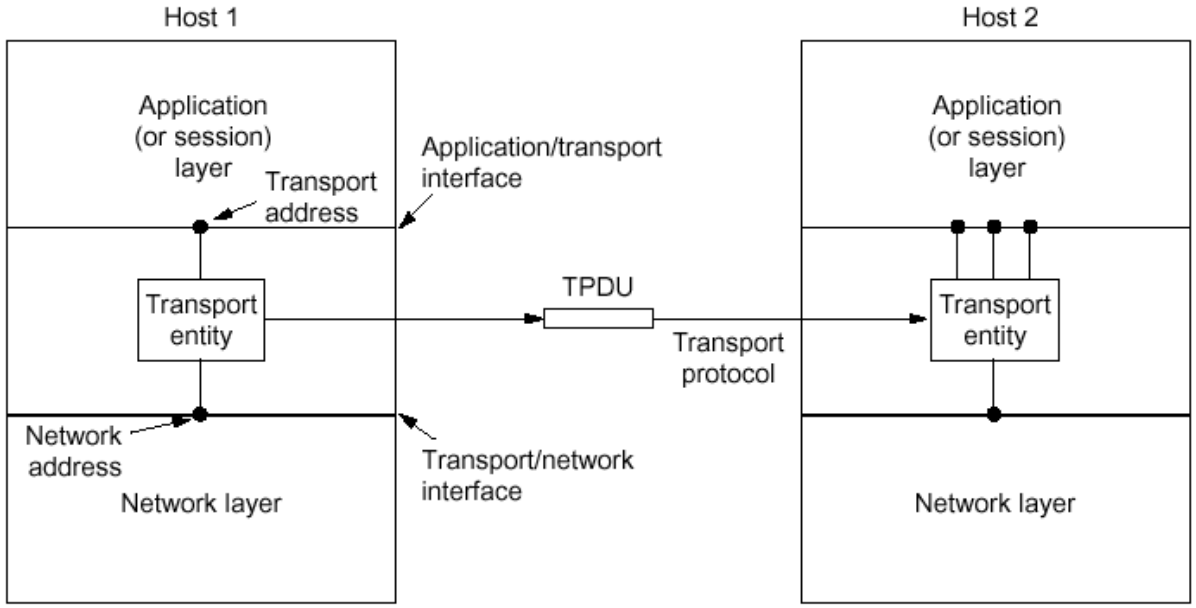
Data Communication Networks Transport Layer

M. R. Pakravan
Department of Electrical Engineering
Sharif University of Technology

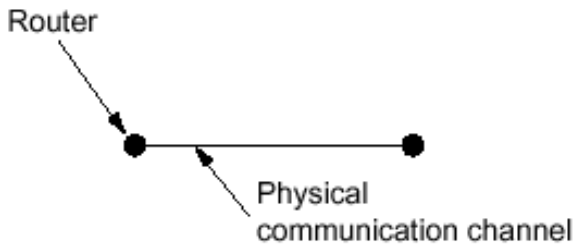
Transport Layer

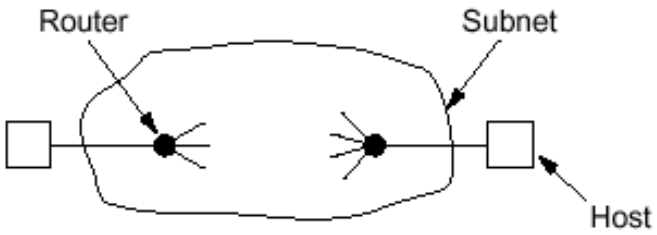
- Tasks:
  - Provide reliable and cost-effective end-to-end communication service to the application layer
  - Independent of used network (shielding)
  - Boundary between network and applications
- TPDU (transport protocol data unit) denotes message sent from entity to entity
- Can be both connection oriented and connectionless service



Transport Service

- Transport Service and Data Link Services:
- Similarities:
  - Both provide point-to-point connection
  - Both must deal with error control, sequencing, flow control, retransmission, etc.
- Differences:
  - Transport connection is indirect
  - Network has memory: packets may be stored and arrive with varying delays and out of order
  - Many connections should be managed (instead of a fixed number of links)



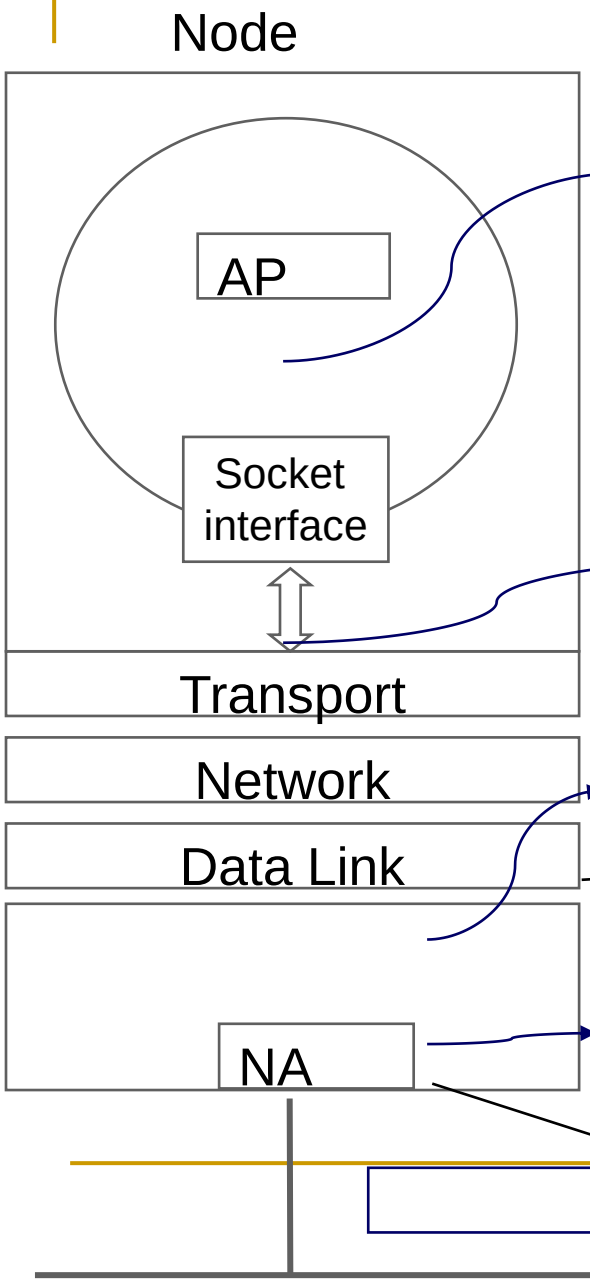


Transport Service

- Primitives for a simple transport service:
  - LISTEN: block until some process connects
  - CONNECT
  - SEND
  - RECEIVE: block until data TPDU arrives
  - DISCONNECT

Primitive	TPDU 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 TPDU arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Protocol Stack



Application

Creates a new end point; allocates table space for it within the transport layer

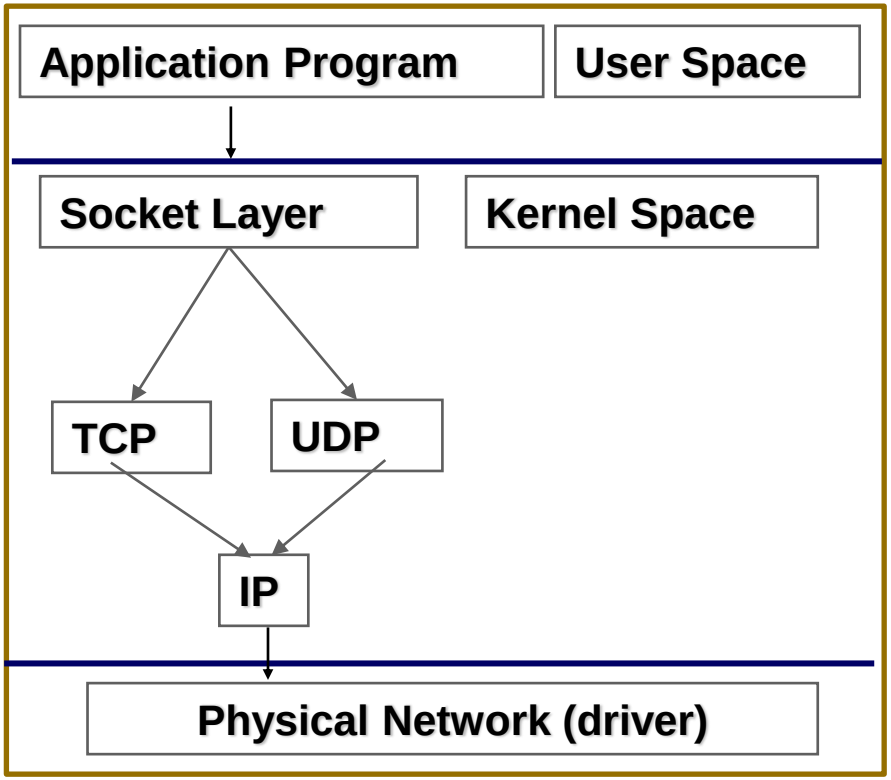
Identification of application (port #)

Identifies the node

Frames

NIC card; identified by NIC card address

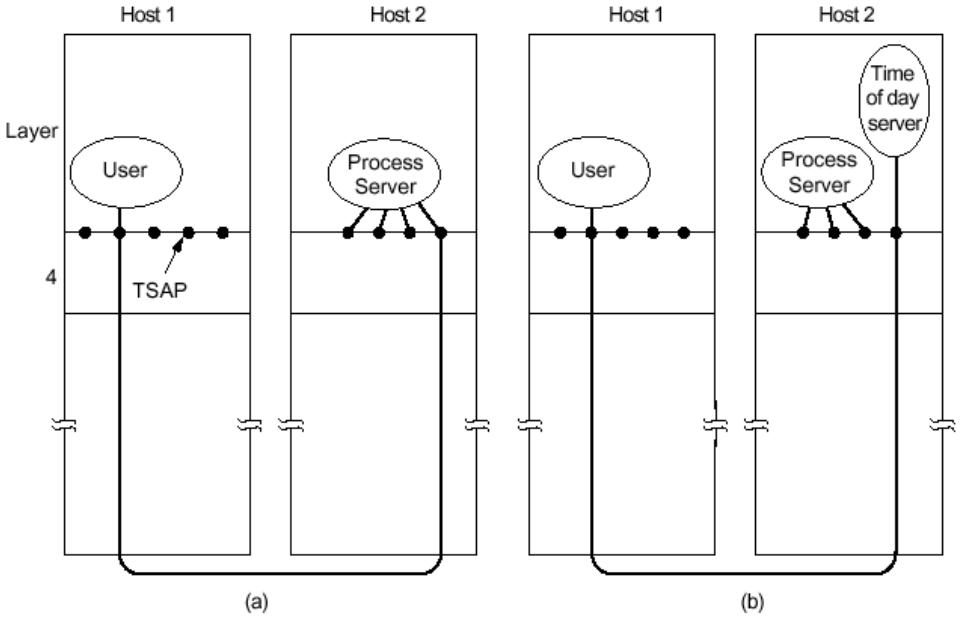
Socket Framework



Elements of transport protocols

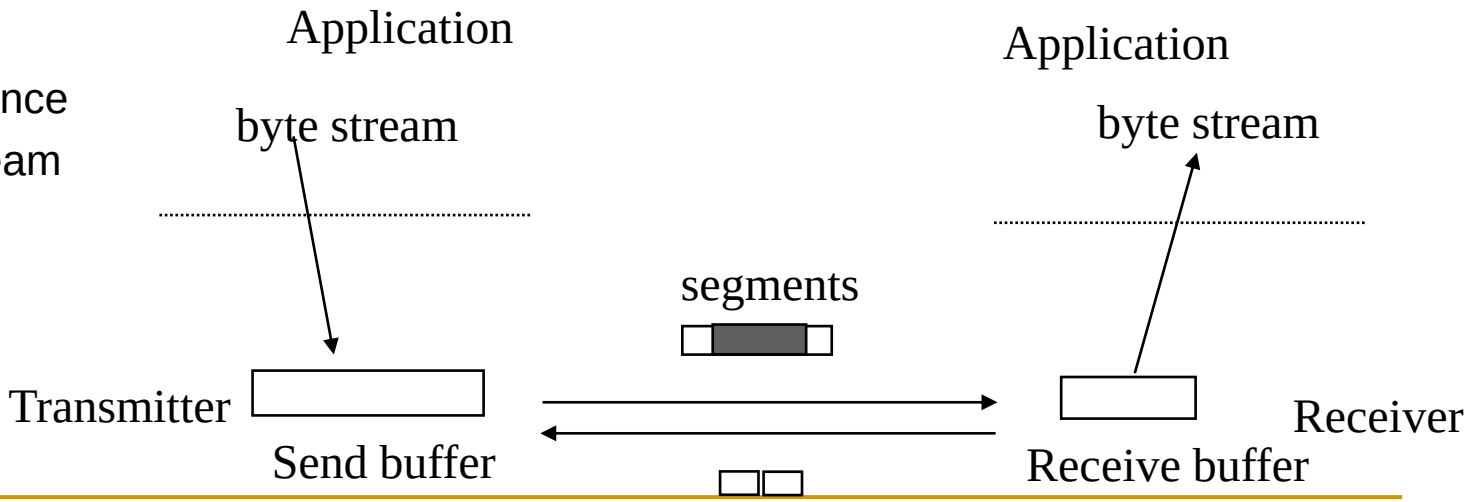
- Addressing: TSAP (transport service access point);
- How do I know the TSAP of the destination?
  - Using well known addresses (works only for (stable) key services)
  - Use name server (or directory server)
    - Connect to this server
    - Send message and ask for TSAP address of needed server
    - Set up a connection with needed server.
    - New services have to register with name server

- TCP addressing:
  - (IP address, port number)
- Example ports of well knows services:
  - port 7: Echo
  - port 23: Telnet
  - port 25: SMTP (email)
  - port 80: HTTP (www)
  - port 110: POP (reading remote email)



Transmission Control Protocol (TCP)

- TCP Provides a logical full duplex connection between two application layer processes across an **unreliable datagram** network (IP network)
- TCP Provides flow control using Selective Repeat
- TCP Can support multiple connections at the same time
- Each direction of the connection can be terminated independently
- TCP Service is:
  - Connection oriented
  - Reliable
  - In-sequence
  - Byte stream



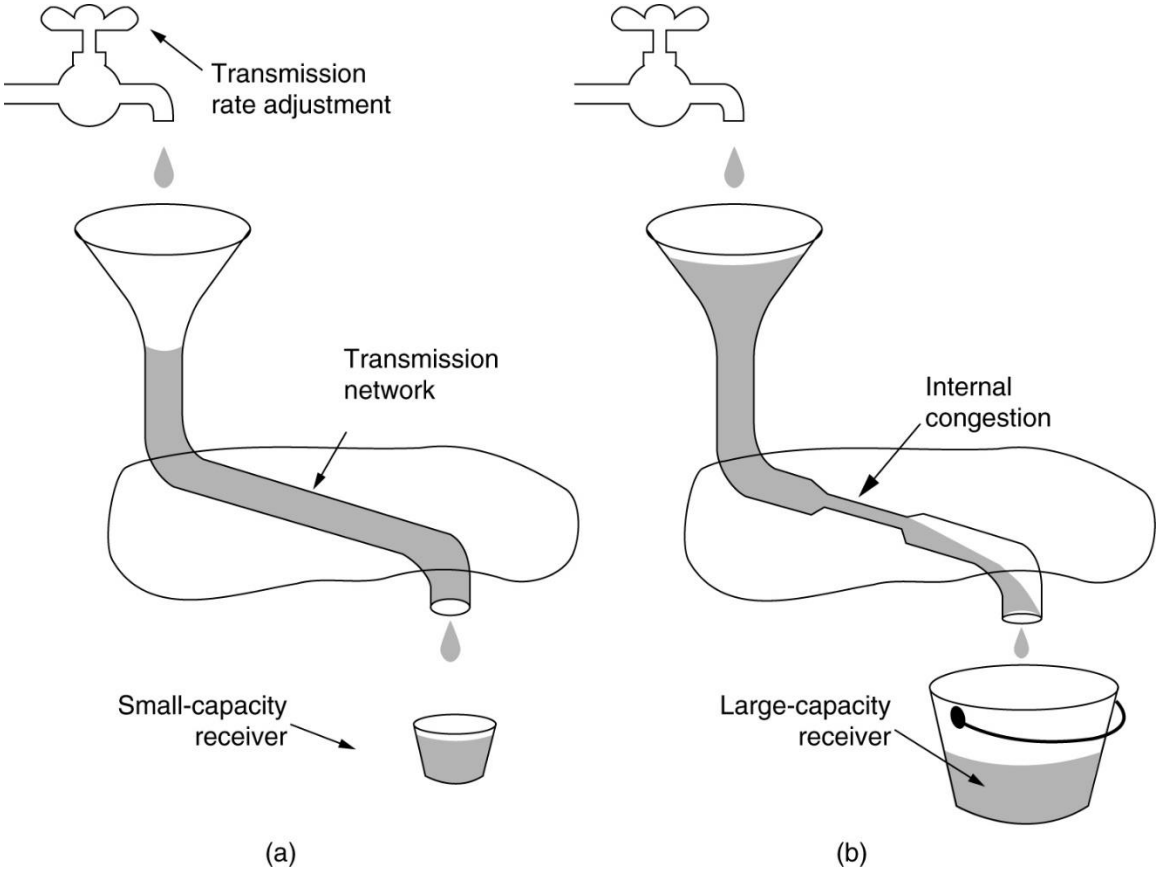
TCP Service Model

- TCP connection is :
  - Full-duplex, point-to-point (No support for multicast or broadcast)
  - Byte stream : message boundaries are not preserved (Application layer should parse and detect messages)
- Both sender and receiver have to create sockets
  - socket # = IP address + Port# (16-bit, = TSAP)
- Connection is identified by: (socket 1, socket 2)
- Port# < 1024 : well-known ports
- FTP: 21, Telnet:23, SMTP: 25, HTTP: 80
- TCP may buffer at both sides; consequently transmission may be delayed
- TCP should handle lost, out of order and delayed segments with non-aligned boundaries.

TCP Service Model

- Every byte has 32-bits sequence number
  - On 10 Mbps it takes about an hour to wrap around
  - Separate 32 bit sequence numbers are used for acknowledgement and for the window mechanism
- Data exchange in segments
  - Segment contains 20 byte header, options, + data
  - Variable payload size (the size is decided by the TCP software and is usually 1500 bytes)
  - Must fit into MTU: maximum transfer unit size of a network
- Sliding window protocol with timeout
  - Receiver sends back ack# equal to next expected segment#
  - Receiver uses piggybacking
  - Sender retransmits if timeout occurs

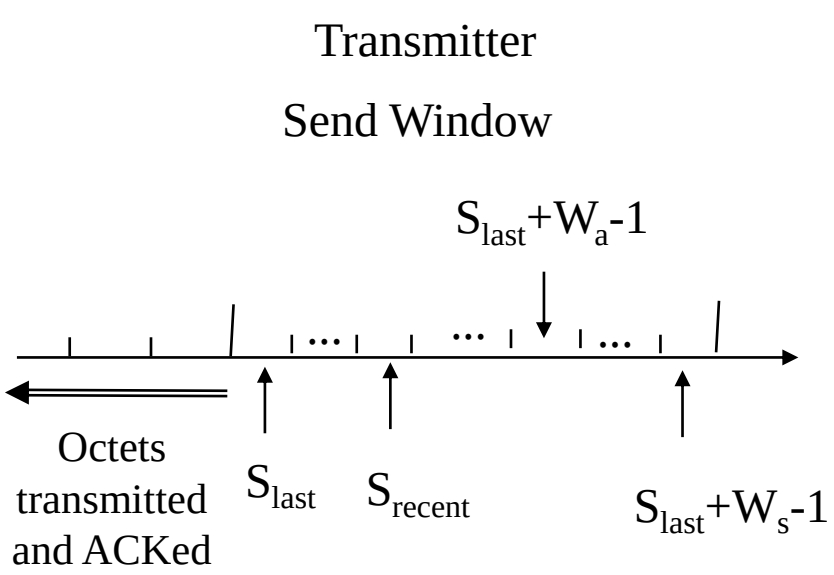
TCP Flow Control vs. Congestion Control

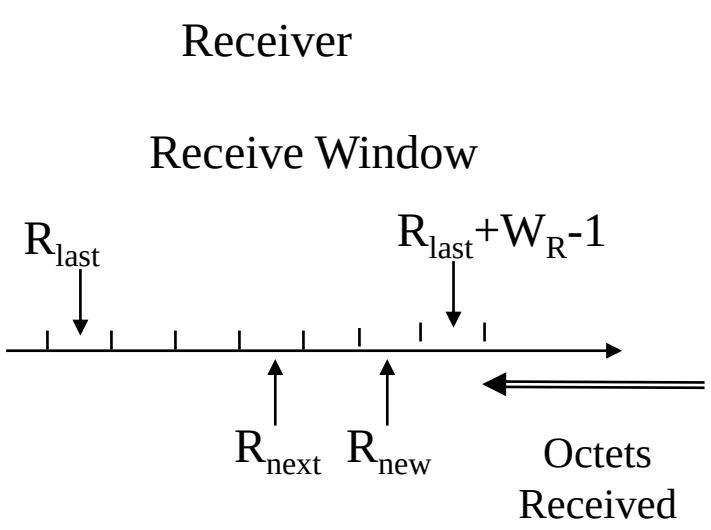


TCP Congestion Control

- TCP controls the data flow between two endpoints of a connection
- Selective repeat sliding window is used for flow control
- Receiver specifies its reception capacity called “Advertised Window” (W_A) in the header of the messages that it sends back to transmitter. This handles the end-to-end flow control
- Network congestion can also be controlled by slowing down the flow of data that TCP generates into the network. This requires feedback from the network behavior.
- The maximum amount of bytes that a TCP sender can transmit without congesting the network is called “Congestion Window” (W_C)
- TCP works by adapting these two windows to control the traffic generation process.

TCP Operation





S_{last} oldest unacknowledged octet
S_{recent} highest-numbered transmitted octet
$S_{last}+W_a-1$ highest-numbered octet that can be transmitted
$S_{last}+W_s-1$ highest-numbered octet that can be accepted from the application

R_{last} highest-numbered octet not yet read by the application
R_{next} next expected octet
R_{new} highest numbered octet received correctly
$R_{last}+W_R-1$ highest-numbered octet that can be accommodated in receive buffer

TCP Operation

- R_{new} can be greater than R_{next} because the receiver accepts out of sequence bytes
- Receiver buffer has a size of W_R
- $R_{last}+W_R-1$ is the maximum numbered byte the receiver can accept
- Advertised window: capacity of buffer available at the receiver: $W_A=W_R-(R_{new}-R_{last})$
- Transmitter should make sure that the number of outstanding bytes is less than the advertised window size: $S_{recent}-S_{last}\leq W_A$
- TCP separates acknowledgment from flow control.
- Advertised window is used to effectively control the traffic flow. It is contained in the header of the frames going back to the sender.

TCP Operation

- String of bytes from application layer are grouped together to form a "segment"
- A 32 bit sequence number is assigned to each byte of the segment.
  - Initial seq# in each direction is chosen randomly and negotiated.
- Timer management:
  - Round trip delay of each segment (Transmission to ack) is measured.
  - Appropriate time is chosen based on history and variation of this delay
  - Typical moving average parameters: α=7/8 and β=1/4

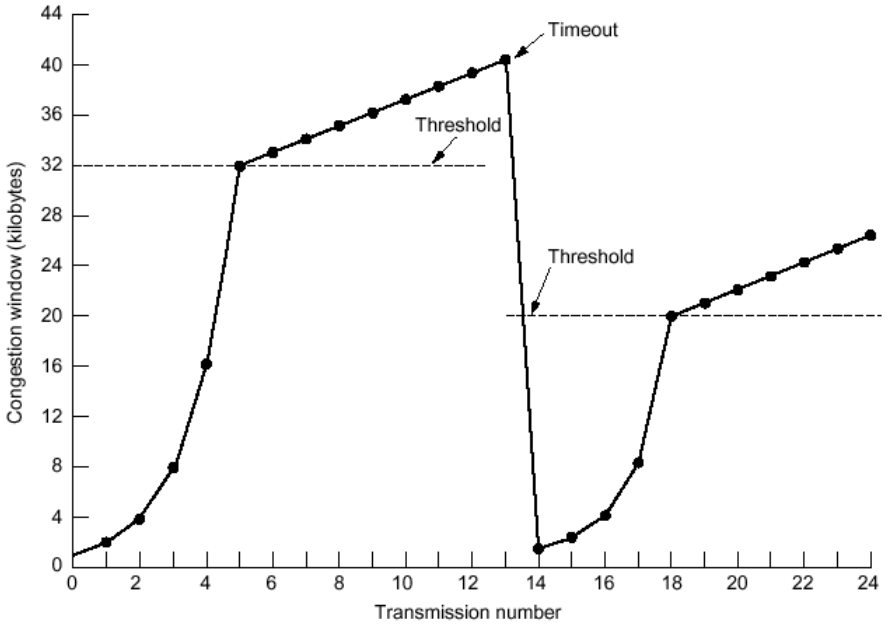
$$t_{RTT}(new) = \alpha t_{RTT}(old) + (1-\alpha)\tau_n$$

$$d_{RTT}(new) = \beta d_{RTT}(new) + (1-\beta)\,|\,\tau_n - t_{RTT}\,|$$

$$t_{out} = t_{RTT} + 4d_{RTT}$$

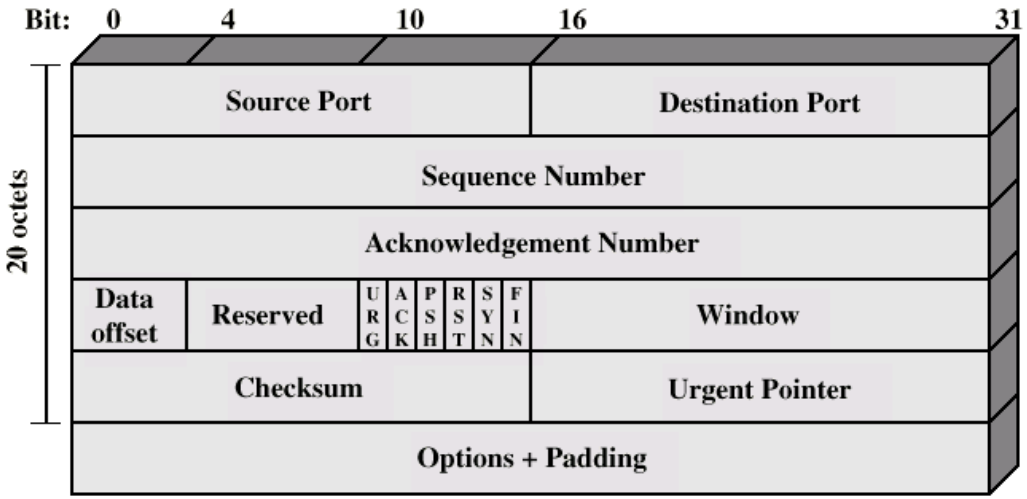
Congestion Control in TCP

- Sender can transmit up to the minimum of "Advertised Window" (W_A) and "Congestion Window" (W_C). This is called Current Window
- TCP transmitter controls W_C dynamically. Steps are:
  - Slow Start: At the start, W_C =1
  - After each successful ACK from receiver, $W_C = W_C * 2$
  - Congestion Avoidance: If W_C =congestion threshold (66,535 bytes), increase W_C linearly with each successful ACK.
  - Congestion Reaction: When there is congestion (Time out on ACK) set congestion threshold to the be $W_{CU}/2$ and set W_C =1
  - Assumption is that ACK loss is due to congestion.
  - This is valid for wired networks, but not for wireless networks where PHY is not as reliable.



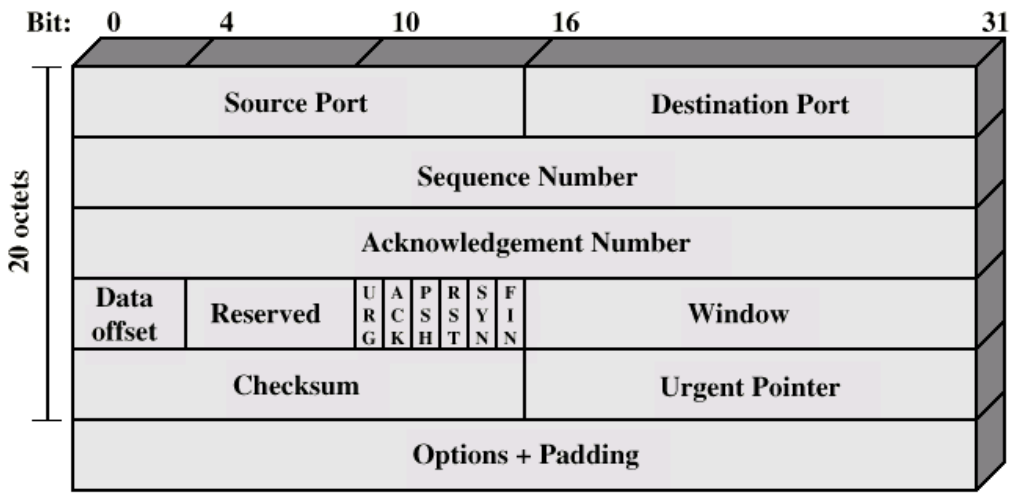
TCP Frame

- Source Port, Destination Port:
  <IP+TCP Port> uniquely identifies a process on each end.
- Seq_no : Position of the first byte of this segment in the sender's byte stream (Can be different in each direction)
- ACK_no : Sequence number of the next data byte that the sender expects to receive if the ACK bit is set
- Window Size: The number of bytes the receiver is willing to accept
- Urgent Pointer: The value of this field added to that in the Seq No. Field points to the last byte of URGENT DATA that needs immediate delivery
- Data Offset:
  Length of TCP header



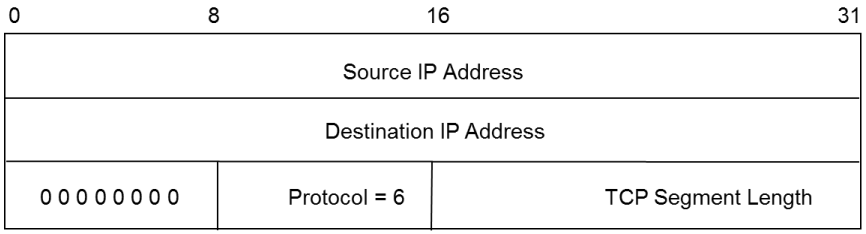
TCP Frame

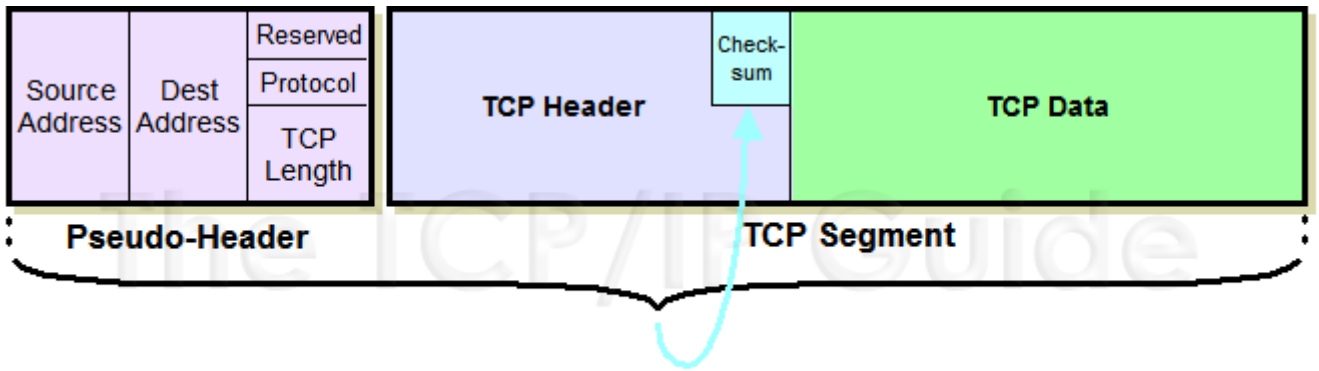
- PSH: if set to 1, receiver should pass the data to application immediately without buffering
- ACK: ACK # is valid if ACK=1
- URG: Urgent # is valid if URG=1
- RST: if set to 1, reset connection
- FIN: If set to one, the sender has no more bytes to send (Can still receive). Connection is terminated if the other end sends a FIN
- SYN: Indicate a connection request or connection acknowledge during the startup procedure
  - Connection Request: [SYN,ACK]=[1,0]
  - Connection Accepted: [SYN,ACK]=[1,1]



TCP Checksum

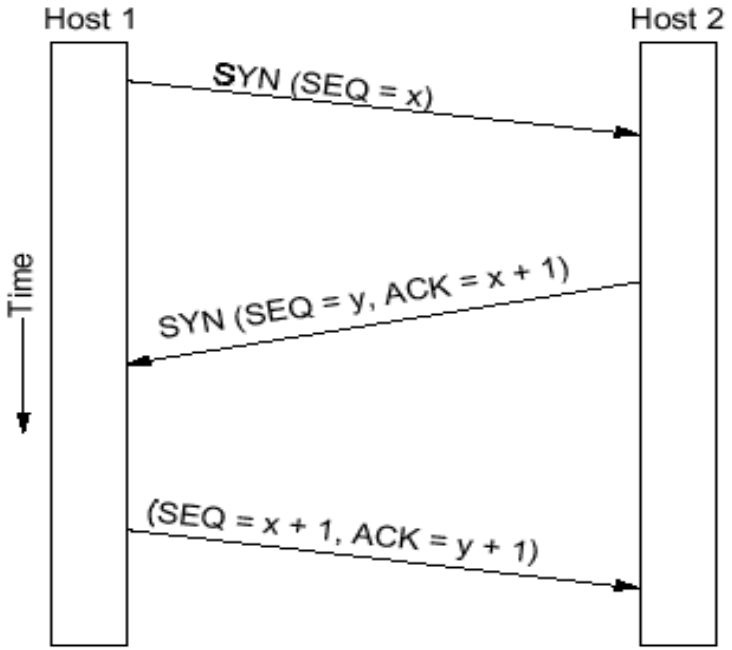
- TCP checksum calculation notes:
  - If segment length is not a multiple of 16, it is padded with zero to make it a multiple of 16 bits.
  - A pseudo-header is created by using the following format and is included in the CRC calculation to make sure that the segment is delivered to the right IP address





TCP Connection Setup

- Host 2 is waiting for incoming connection
- Host 1: Sends a request, SYN=1, Seq_no=x
- Host 2: Replies by setting ACK=1, ACK_no=1+x, SYN=1, Seq_no=y
- Host 1: Acknowledges by setting the ACK=1, ACK_no=1+y, Seq_no=1+x
- If during a connection establishment phase, one of the hosts decides to refuse a connection, it will send a RESET segment by setting the RST bit.

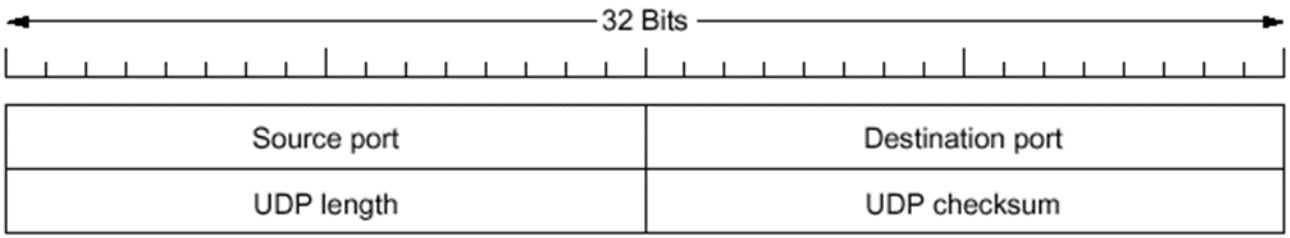


User Datagram Protocol (UDP)

- UDP in an unreliable, connectionless transport protocol
- UDP does not provide guarantee of delivery, ordering, or duplicate protection.
- UDP is suitable for purposes where error checking and correction is either not necessary or is performed in the application, avoiding the overhead of such processing at the network interface level.
- Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.

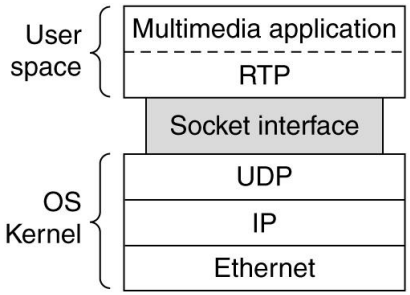
UDP

- UDP Services:
  - Routes the received packet to the desired application on the host (Destination Port)
  - Checks the integrity of the datagram. This is optional! (UDP Checksum)
    - If a host does not wish to calculate the checksum, it sets it to all 0's.
- Applications that use UDP: Domain Name Services (DNS), Simple Network Management Protocol (SNMP), Real Time Protocol (RTP)
- UDP checksum calculation: Similar to TCP (pad to 16, psuedo-header for IP verification)

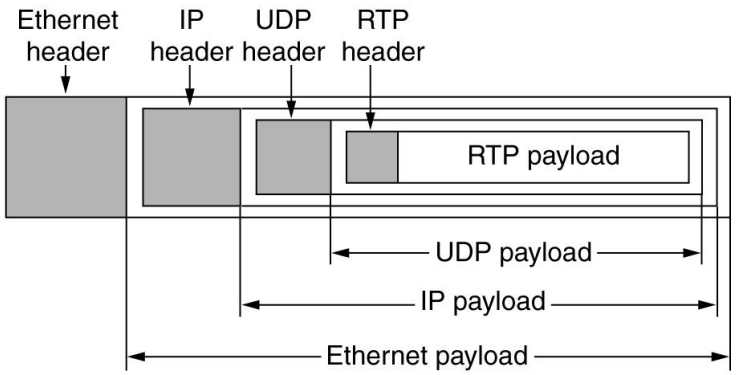


Real Time Transport Protocol (RTP)

- A generic protocol for real time applications such as voice and video
- Uses UDP and acts as an interface between user application and transport protocol (Mostly UDP)
- Header specifies the profile and encoding format of the payload (for example: single audio stream, mp3)
- Packets are numbered to allow detection of missing packets
- Time stamping is used to allow synchronization and jitter compensation
- No flow control, no error control, no acknowledgement and no repeat-requests
- Services:
  - Payload type identification
  - Sequence numbering
  - Time stamping
  - Delivery monitoring



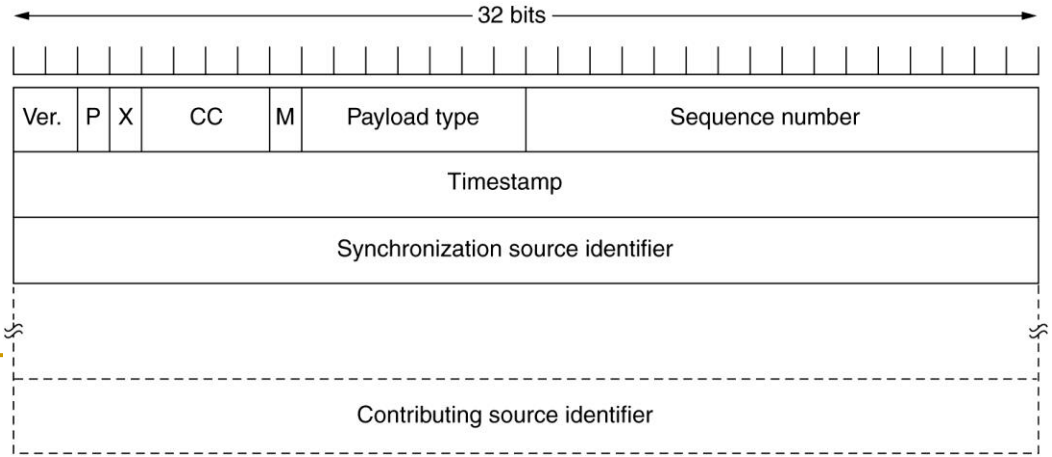
(a)



(b)

Real Time Transport Protocol (RTP)

- P bit: the packet has been padded to a multiple of 4-bytes. The last padding byte tells how many bytes were padded
- X bit: an extension header is present
- CC field: How many contributing sources of data are present
- M bit: Marker bit used by the application
- Time stamp of the first sample of the payload content is noted in the header:
  - Appropriate time alignment of the samples in the receiver for smooth playback and jitter reduction.
  - Synchronization between multiple streams such as video and audio in a video-conference.
- Synch Source ID: Tells which stream this packet belongs to.



Real Time Transport Control Protocol

- RTP is used in conjunction with the RTP Control Protocol (RTCP).
- RTCP provides out-of-band statistics and control information for an RTP session.
- RTCP partners with RTP in the delivery and packaging of multimedia data but does not transport any media data itself.
- The primary function of RTCP is to provide feedback on the quality of service (QoS) in media distribution by periodically sending statistics information to participants in a streaming multimedia session.
- When both protocols are used in conjunction, RTP is originated and received on even port numbers and the associated RTCP communication uses the next higher odd port number.
- RTCP feedback can let the source know about delay, jitter, bandwidth, congestion, etc.
- This data can be used by the source to adjust encoding or transport properties of the stream
- Different streams can use different clocks with different granularities and different drift rates.