5 : Transport Layer

IT4506 – Computer Networks

Level II - Semester 4

Overview

- In this topic we discuss the Transport layer services and protocols in the transport layer.

Intended Learning Outcomes

- At the end of this lesson, you will be able to;
  - Discuss the protocols in the transport layer.
  - Describe the Internet Transport Protocols.
  - Explain the User Datagram Protocol including the Remote Procedure Calls and Realtime Transport Protocols.
  - Summarise the Transmission Control Protocol with the feature of TCP including Connection Establishment, release, TCP sliding window and TCP congestion control.

List of sub topics

5.1 Transport Layer Services and Primitives

5.2 Internet Transport Protocols

- 5.2.1 User Datagram Protocol (UDP)
- 5.2.2 Transmission Control Protocol (TCP)

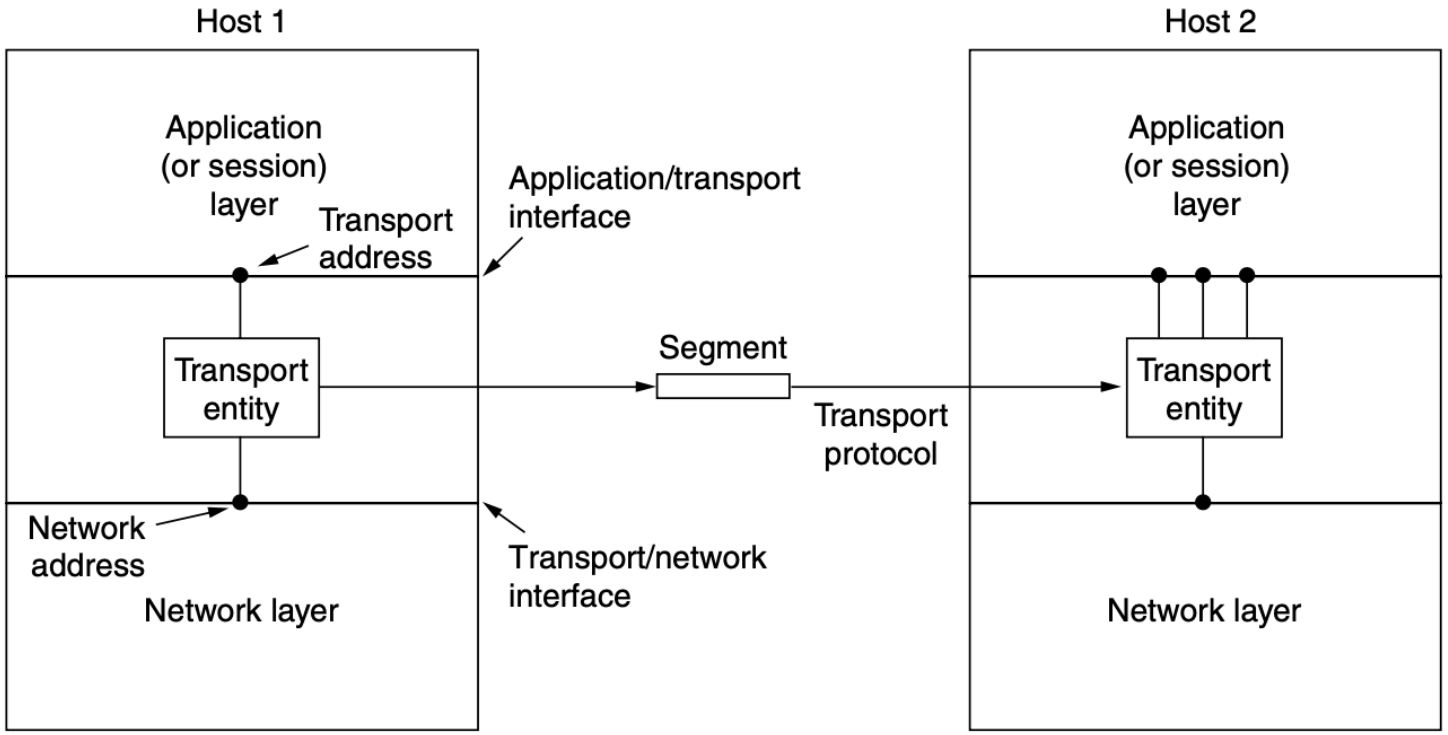
Transport Layer

- The Network Layer enables end-to-end packet delivery by using the datagrams or virtual circuits.
- The transport layer provide data transport from a process of the source machine to the process of the destination machine.
- Transport layer provides a desired level of reliability independent of the physical network used.

Transport Layer Services

- Services Provided to the Upper Layer
  - Main objective of the Transport layer is to provide **efficient**, **reliable**, and **cost-effective** data transmission service to users/ processes in the application layer.
  - In order to provide the services to the application layer, the transport layer should obtain the services from the Network layer.
  - The software/ hardware component which perform this task within the Transport layer is the **Transport Entity**.
    - It can be located in the kernel of the operating system.
    - Or in a library package in the network application.
    - Or on the network interface card (NIC).

Transport Layer Services...(2)



Ref1: pg 496

Transport Layer Services...(3)

- There are two types of transport services
  - Connection-oriented - Similar to connection-oriented network service.
    - It has three phases
      - Connection establishment
      - Data transfer
      - Release
  - Connectionless - Similar to connectionless network service.

Think!!!
If both the network and transport services are similar, what is the requirement to have the same services in two different layers?

Transport Layer Services…(4)

- If both the network and transport services are similar, what is the requirement to have the same services in two different layers?
  - Transport services executes on users' machine
  - Network services mainly run on devices like routers
- Users do not have any control on the devices in the network such as router. Therefore, if there is a performance issue/ error/ poor service, users cannot replace the device or make any changes.
- The solution is to put another layer on top of network layer to improve the quality of the service (Transport Layer).
- Transport layer makes it possible for the transport service to be more reliable than the underlying network.
- Application programmers can write code according to a standard set of primitives and have these programs work on a wide variety of networks, without having to worry about dealing with different network interfaces and levels of reliability.

Transport Layer Primitives

- The main difference between Network services and Transport services
  - Network layer is working with the real networks (devices, routers.. etc). Real network can loose packets. Therefore, the network service is unreliable.
    - But connection-oriented transport service is reliable. The purpose of the transport layer is to provide a reliable service on top of an unreliable network.
      - The Transport service is all about hiding the imperfections of the network service.
      - User processes can just assume the existence of an error-free bit stream even when they are on different machines.
      - Transport layer can also provide unreliable service. (Eg: video streaming applications)

Transport Layer Primitives...(2)

- Another difference between Network services and Transport services is
  - The network services is only used by the Transport entities.
  - But, many programs see/use the transport primitives.

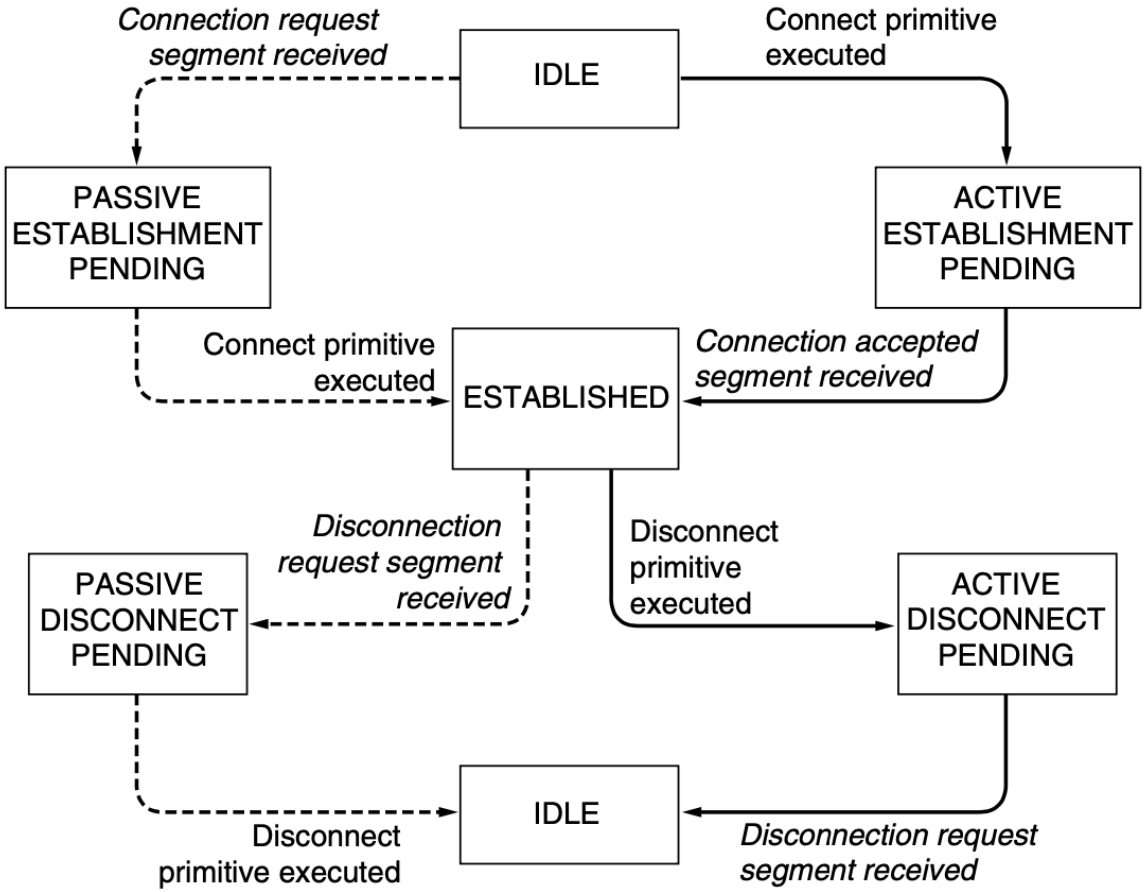
Transport Layer Primitives...(3)

- Primitives for a simple transport service
  - LISTEN - Block until some process tries to connect
  - CONNECT - Actively attempt to establish a connection
  - SEND - Send information
  - RECEIVE - Block until a DATA packet arrives
  - DISCONNECT - Request a release of the connection

Transport Layer Primitives...(4)

- Consider an application with a server and number of clients connected.
  - The server executes a LISTEN primitive
  - When a client wants to talk to the server, it executes a CONNECT primitive.
  - The client's CONNECT call causes a CONNECTION REQUEST segment to be sent to the server.
  - When it arrives, the transport entity checks to see that the server is blocked on a LISTEN. If so, it then unblocks the server and sends a CONNECTION ACCEPTED segment back to the client. When this segment arrives, the client is unblocked and the connection is established.
  - Data can now be exchanged using the SEND and RECEIVE primitives.
  - Disconnection has two variants
    - Asymmetric - Either user send DISCONNECT primitive and the connection get closed
    - Symmetric - When one user send DISCONNECT primitive, it implies that there is not data to send. Then other user also has to send DISCONNECT primitive to close the connection.

Transport Layer Primitives...(5)

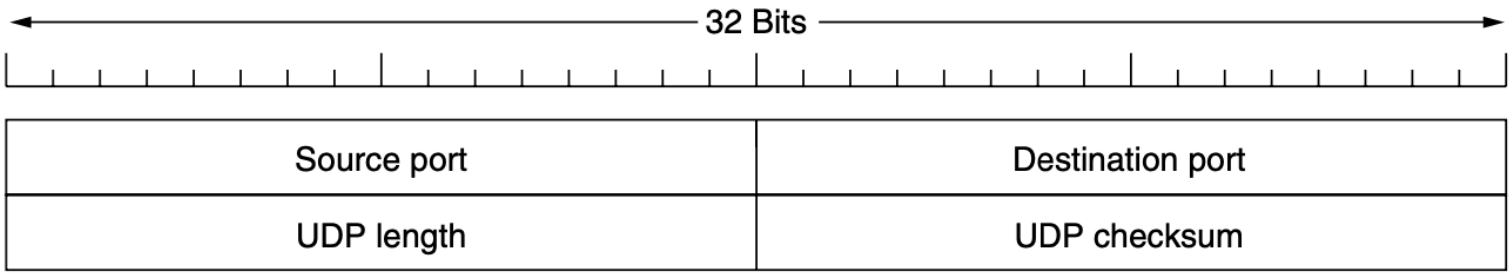


Internet Transport Protocols

- There are two main types of Transport Protocols
  - Connectionless protocol
    - UDP (User Datagram Protocol)
  - Connection-oriented protocol
    - TCP (Transmission Control Protocol)

User Datagram Protocol

- User Datagram Protocol (UDP) is a connection-less protocol.
- It support application to sending encapsulated IP datagrams without having establishing a connection.
- UDP has a 8byte header followed by the payload.



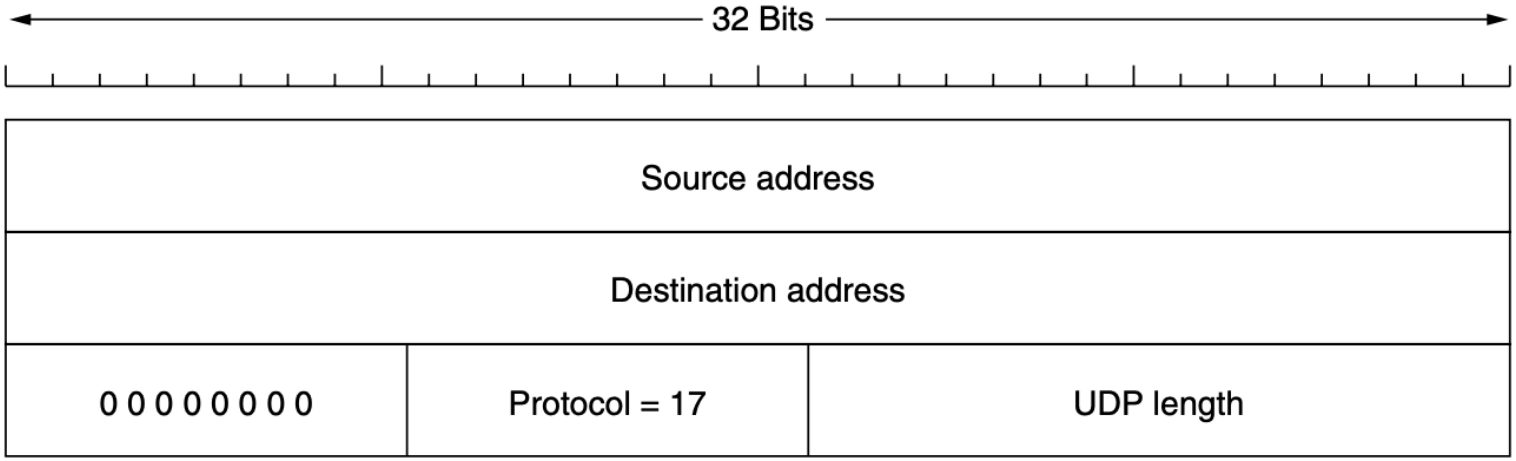
Ref1: pg 542

User Datagram Protocol…(2)

- UDP Header
  - ***Source port and Destination port*** - without the ports, transport layer does not know what to do with the incoming segments. By using the ports, the transport layer deliver the segments to the correct application.
  - ***UDP Length*** - length includes the total segment length including the 8 bytes header and the data. Minimum length has to be 8 bytes (the size of header) and maximum length is 65,515 bytes.
  - ***UDP Checksum*** - it is included to provide reliability. It checksums the header, the data, and a conceptual IP pseudo-header.

User Datagram Protocol...(3)

- UDP Header
  - IPv4 pseudo-header included in the UDP checksum



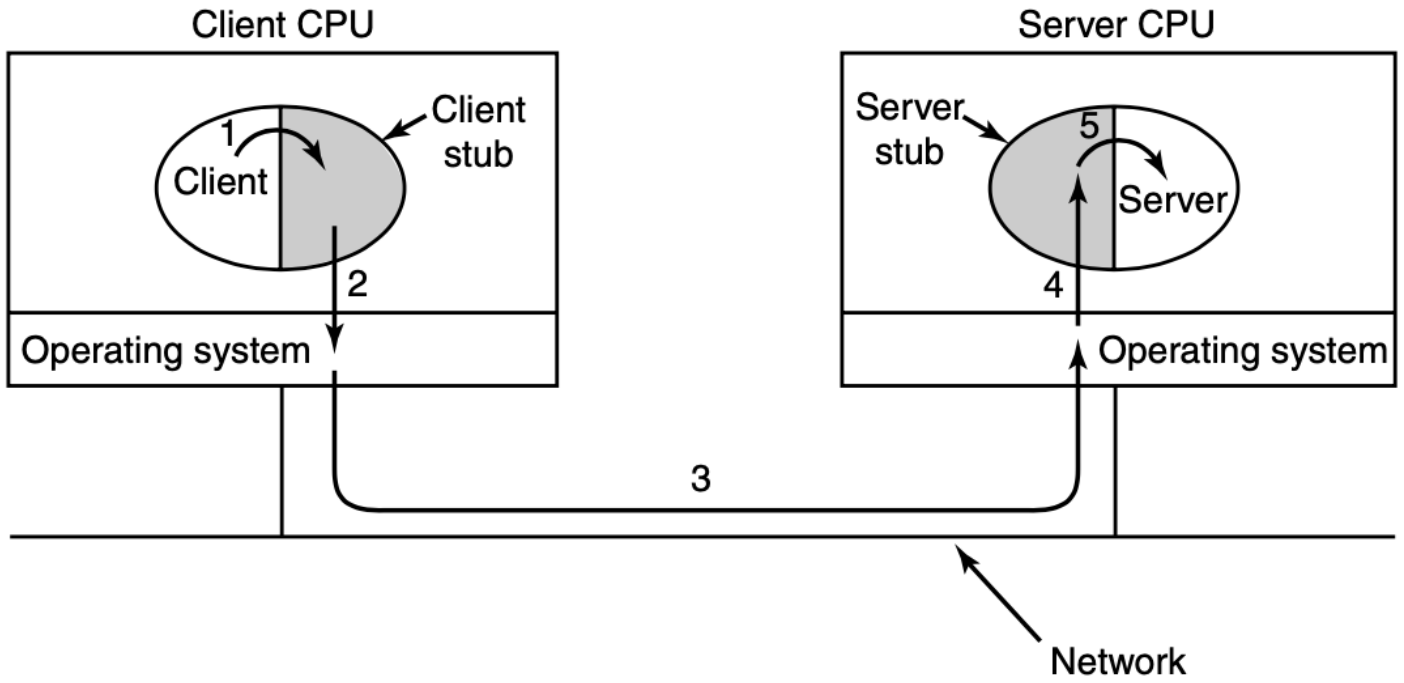
Ref1: pg 543

User Datagram Protocol : Remote Procedure Call (RPC)

- Remote Procedure Call (RPC)
  - Image there are two machines as ***Machine 1*** and ***Machine 2***. ***Machine 1*** need to execute a function (procedure) in the the ***Machine 2***. Which is a complex operation and known as ***Remote Procedure Call.***
  - In the above scenario the Machine 1 is the client which the calling procedure and Machine 2 is the server which is the called procedure.
  - In the RPC the procedure call look as much as possible as a local one.
    - To achieve this there are two library procedures,
      - Client stub - represent the server procedure in the client's address space.
      - Server stub - represent the client procedure in the server's address space.
    - These procedures will hide the fact that the procedure called from client to the server is not local.

User Datagram Protocol : Remote Procedure Call (RPC)…(2)

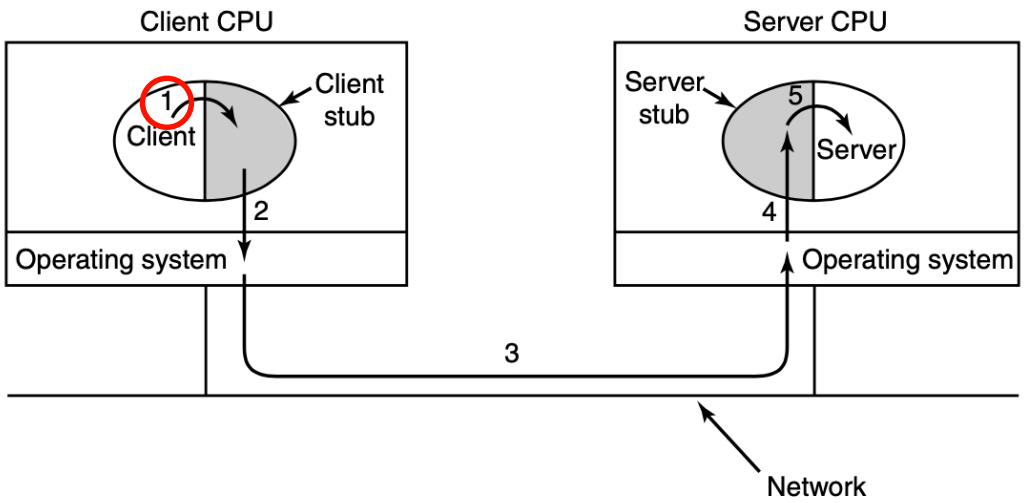
- Steps of Remote Procedure Call



Ref 1: pg 545

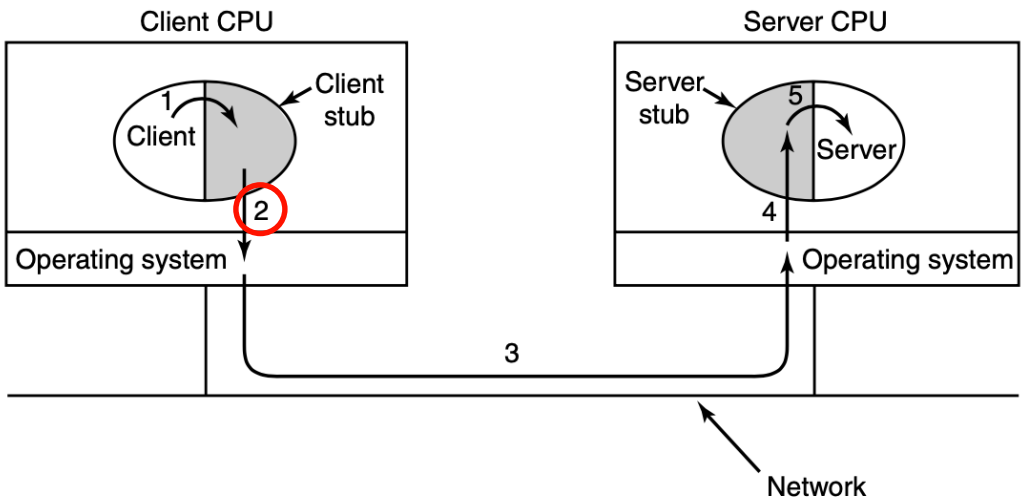
User Datagram Protocol : Remote Procedure Call (RPC)…(3)

- Steps of Remote Procedure Call
  - Step 1 : Client calling the client stub.
    - Local procedure call.
    - Parameters are pushed onto the stack in the normal way.



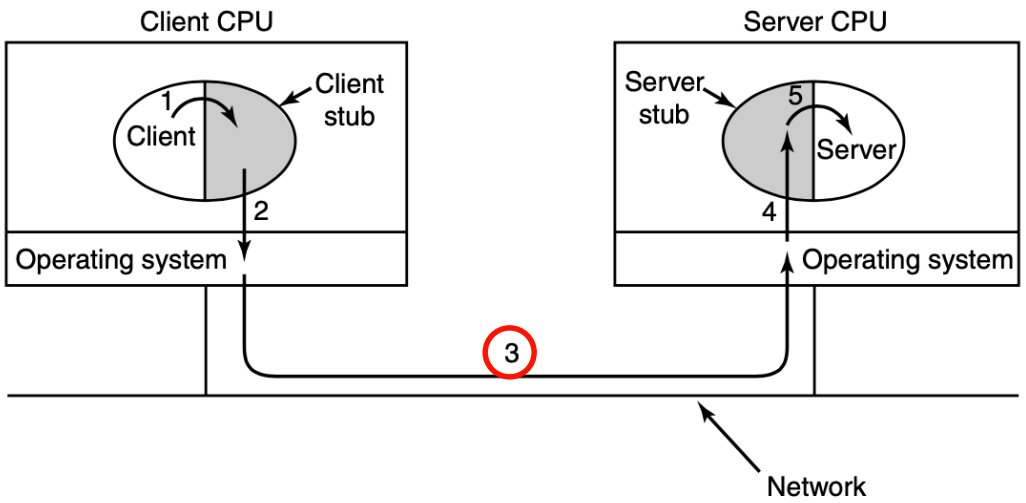
User Datagram Protocol : Remote Procedure Call (RPC)...(4)

- Steps of Remote Procedure Call
  - Step 2 : client stub packing the parameters into a message and making a system call to send the message.
    - Packing the parameters is called **marshaling**.



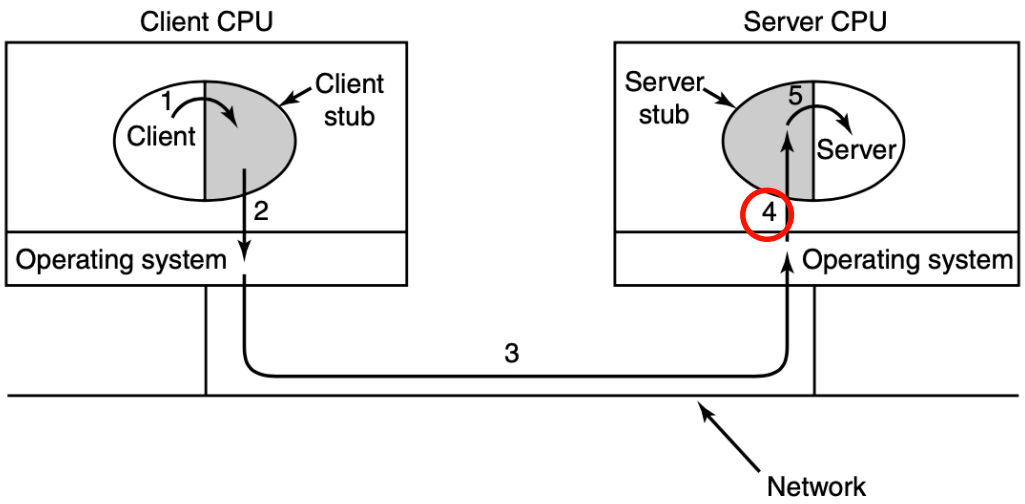
User Datagram Protocol : Remote Procedure Call (RPC)…(5)

- Steps of Remote Procedure Call
  - Step 3 : The operating system sending the message from the client machine to the server machine.



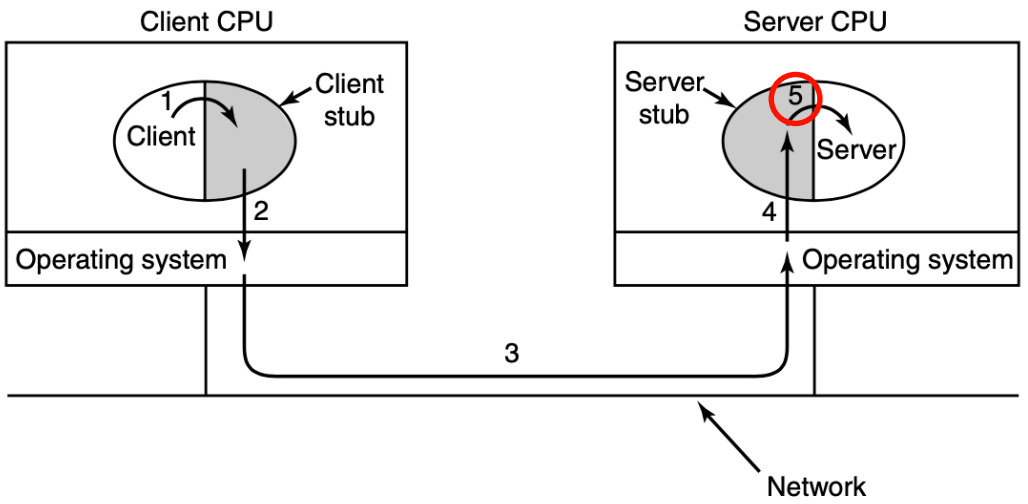
User Datagram Protocol : Remote Procedure Call (RPC)…(6)

- Steps of Remote Procedure Call
  - Step 4 : The operating system passing the incoming packet to the server stub.



User Datagram Protocol : Remote Procedure Call (RPC)...(7)

- Steps of Remote Procedure Call
  - Step 5 : the server stub calling the server procedure with the unmarshaled parameters.
    - The reply traces the same path in the other direction.

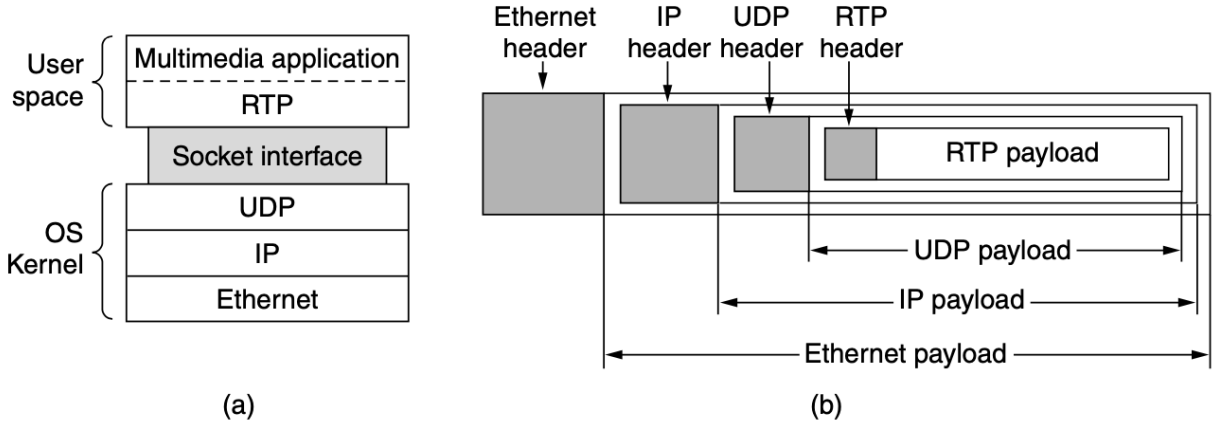


User Datagram Protocol : Remote Procedure Call (RPC)…(8)

- UDP is used for client-server remote procedure calls
  - Eg: DNS requests and replies.
- Problems in RPC
  - Use of pointer parameters is difficult, because calling and the called procedures are not in the same virtual address.
  - If the size of the parameters are not clearly defined, it is impossible for the client stub to marshal the parameter.
  - Difficulty in deducing the type of the parameter.
  - If the calling function and called function are in the same machine, the global variables can be used. But when the called function is moved to a remote machine it does not have the access to the global variable.

User Datagram Protocol : Real-Time Transport Protocols

- Two aspects of Real Time Transport Protocols
  - Transporting audio and video data in packets
  - Play out the audio and video at the right time



(a) - Position of RTP in the protocol stack
(b) - Packet nesting

- (b) shows how the RTP packet is nested inside the PDU.

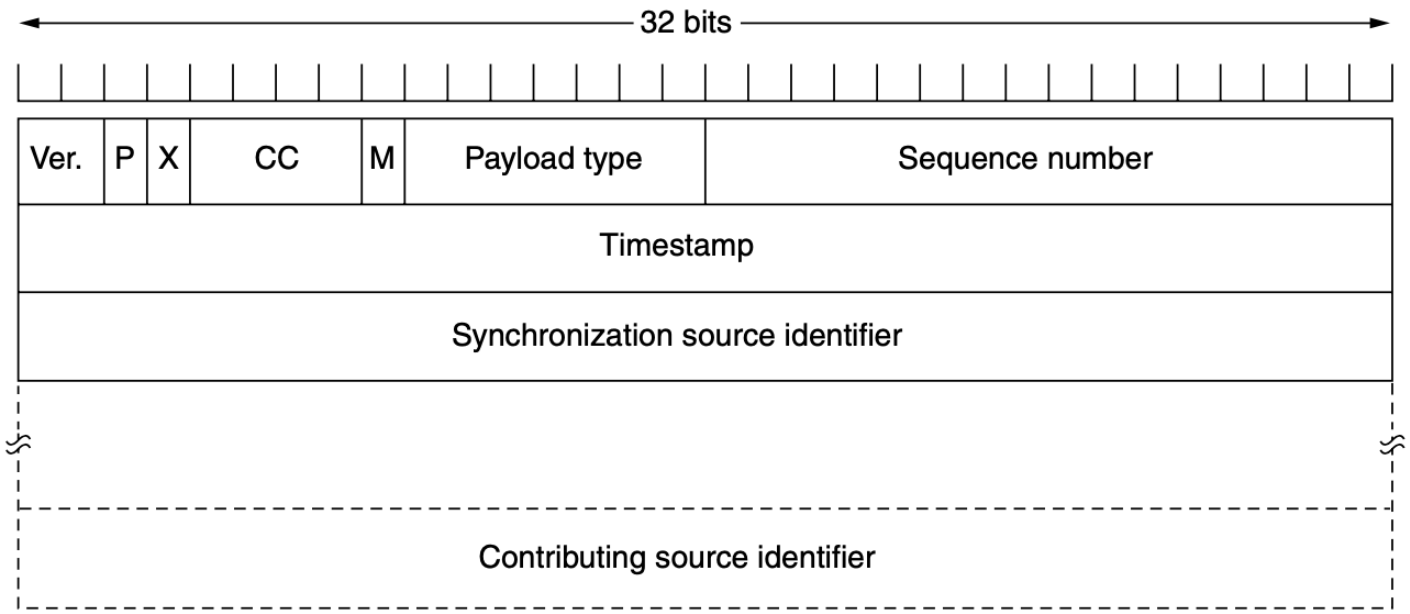
Ref 1: pg 547

User Datagram Protocol : Real-Time Transport Protocols…(2)

- Basic functionality of RTP is to multiplex several real-time data streams onto a single stream of UDP packets.
- RTP use UDP. Therefore,
  - no special guarantees about delivery, and packets may be lost, delayed or corrupted.
- Each packet sent in RTP is given a number higher than the predecessor. With that number the Operating System identify whether there is a missing packet.
- If it notifies that a packet is missing, the application will skip the frame of the video or control it from the application, but no retransmission is happened.
- Timestamp is another important feature should be available with real time applications.
  - RTP has timestamp which can be used to buffer the content at the destination and used for the synchronisation.

User Datagram Protocol : Real-Time Transport Protocols…(3)

- RTP header



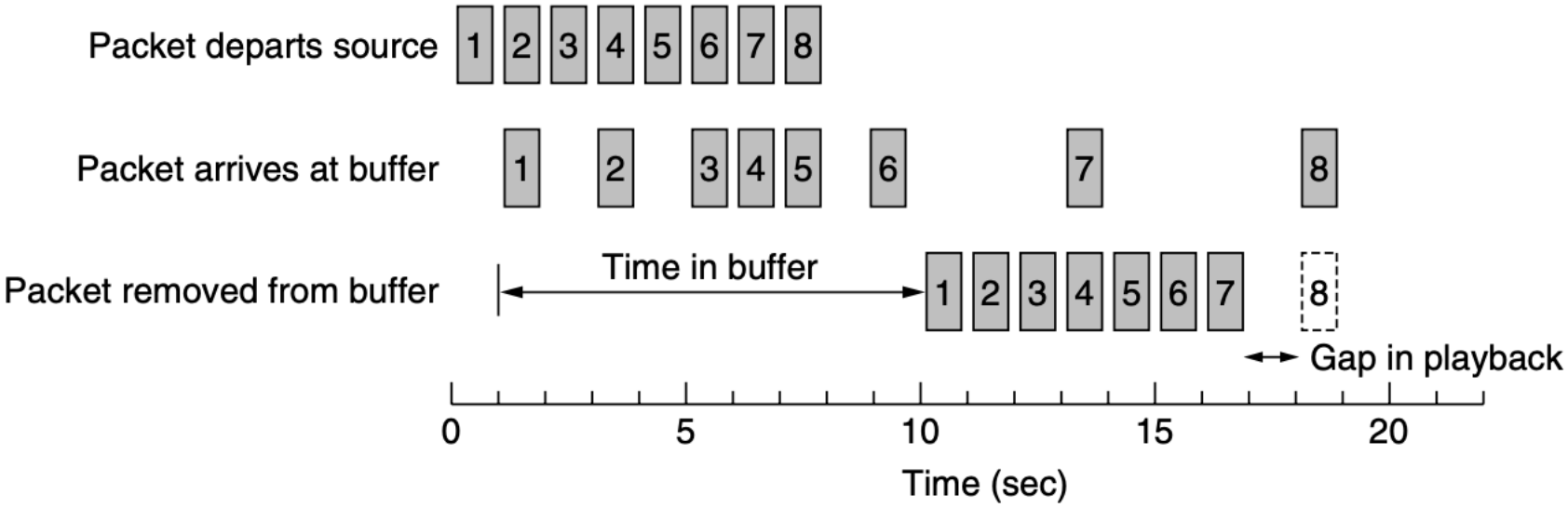
User Datagram Protocol : Real-Time Transport Control Protocols (RTPC)

- RTPC
  - This is used to provide feedback on delay, variation in delay or jitter, bandwidth, congestion, and other network properties to the sources.
  - Based on this continues feedback, he encoding algorithms can be continuously adapted to provide the best quality possible under the current circumstances.

User Datagram Protocol : Real-Time Transport Control Protocols (RTPC)…(2)

- Playout with Buffering and Jitter Control
  - Packets will not arrive at the destinations in the same time interval as it is injected to the network.
  - The variation of this delay is the jitter.
  - Even a small amount of packet jitter can cause distracting media artifacts, such as jerky video frames and unintelligible audio, if the media is simply played out as it arrives.
  - Solution for this matter is to **buffer** packets at the receiver before played out.

User Datagram Protocol : Real-Time Transport Control Protocols (RTPC)...(3)



Ref 1: pg 551

User Datagram Protocol : Real-Time Transport Control Protocols (RTPC)…(4)

- If the packets are delayed for considerable amount of time,
  - The packet can be skipped,
  - Or playback can be stopped.
- Playback point - time waiting at the receiver for media before playing. This is depending on the jitter.

The Internet Transport Protocol - TCP

- Transmission Control Protocol - specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork.
- Different parts of the Internet has different topologies, bandwidths, delays, packet sizes, and and so on.
- TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures.
- Each machine supporting TCP has a **TCP transport entity**, either a library procedure, a user process, or most commonly part of the kernel.
  - A TCP entity accepts user data streams from local processes, breaks them up into pieces not exceeding 64 KB (fit into a single Ethernet frame), and sends each piece as a separate IP datagram.
  - When a datagrams containing TCP data arrive at a machine, they are given to the TCP entity, which reconstructs the original byte streams.
- All the TCP connections are full duplex and point to point.

TCP Service Model

- TCP service is obtained by both the sender and the receiver creating end points, called **sockets**.
- Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port**.
- For TCP service to be obtained, a connection must be explicitly established between a socket on one machine and a socket on another machine.

TCP Service Model...(2)

- Ports
  - Port numbers below 1024 are reserved for standard services that can usually only be started by privileged users (well-known ports).
    - FTP - 20,21
    - SSH - 22
    - HTTP - 80
    - HTTPs - 443

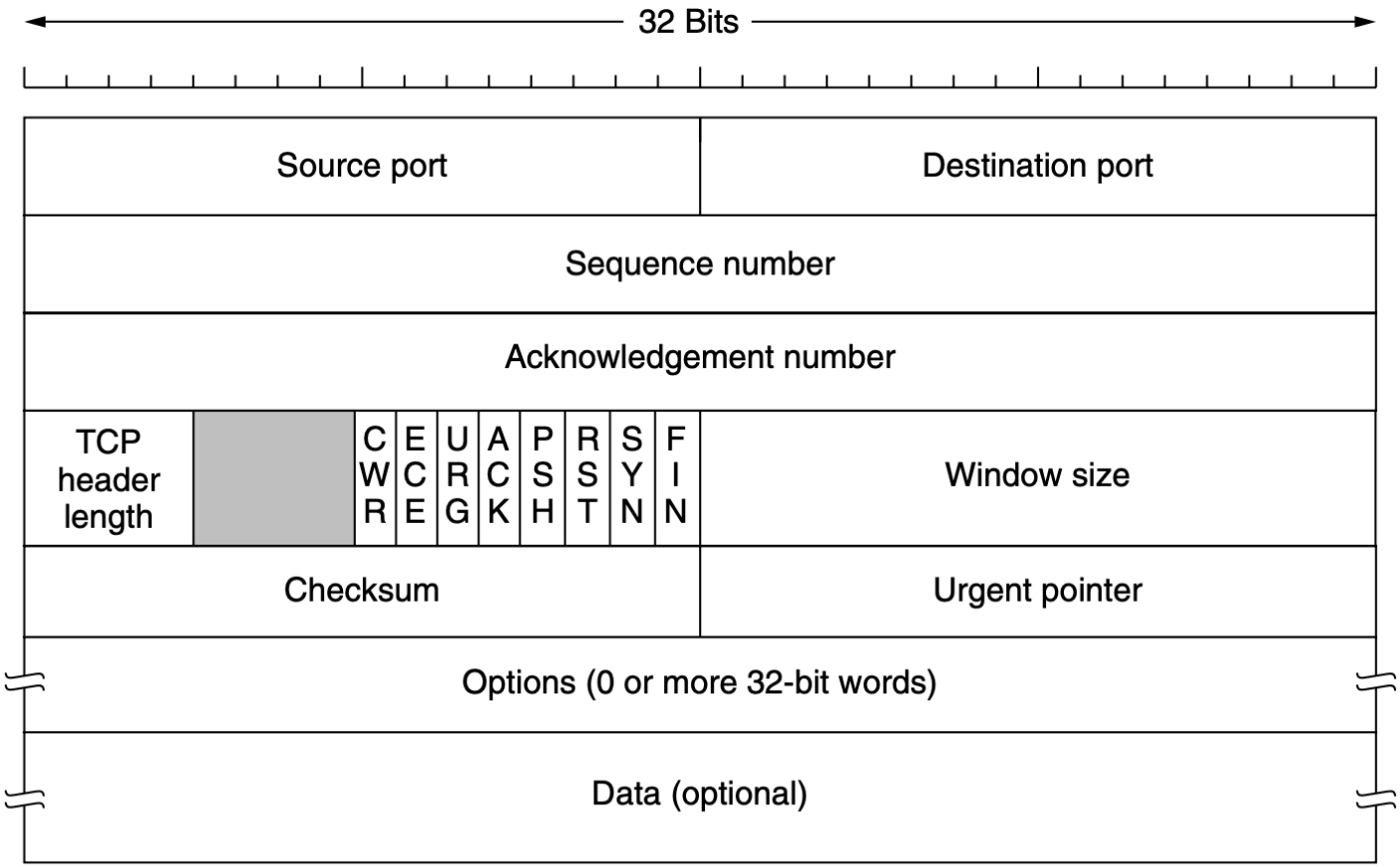
The TCP Protocol

- Every byte of TCP connection has its own 32 bit sequence number.
- Sending and receiving TCP entities exchange data in the form of **segments**.
- TCP segment consists of a fixed 20-byte header.
- TCP software can decide the size of the segment.
  - It can accumulate data from several writes into one segment of split or can split data from one write over multiple segments.
- Limits that restrict the size of segment
  - TCP segment including the deader must fit in the 65,515 byte IP payload.
  - Link Maximum Transfer Unit (MTU)

The TCP Protocol…(2)

- When sender sends a segment, it starts a timer.
- After the receiver receives the segment, it reply with an acknowledgement containing the sequence number it expects to receive.
- If the sender's timer goes off before the acknowledgement receives, the sender will retransmit.
- If the segments receives out of order,
  - Segment "B" cannot acknowledge if the segment "A" has not received. (Assume the correct order of segments are A and B)

The TCP Segment Header



Ref 1: pg 557

The TCP Segment Header…(2)

- The *source port* and the *destination port* is used to identify the local end points of the connection.
  - TCP port plus its host's IP address forms a 48-bit unique end point.
- Sequence number and the acknowledgement number fields carries the sequence number and the acknowledgement numbers.
- TCP header length contains how many 32bit words are contain in the TCP header.
  - This information is important because the size of the Option field is changing.

The TCP Segment Header…(3)

- 1 bit flags
  - CWR, ECE - used in signal congestion (to slow down the sending of segments).
  - URG - immediately give the data to the application layer.
  - ACK - used in acknowledgement segment.
  - PSH - if this flag is set, immediately the segment will send to the network layer, and at the receiver's end the segment will immediately send to the application layer.
  - RST - used to terminate the connection when there is a problem in the connection.
  - SYN - used in synchronisation step of three way handshake
  - FIN - used to terminate the connection (when no more data to send).

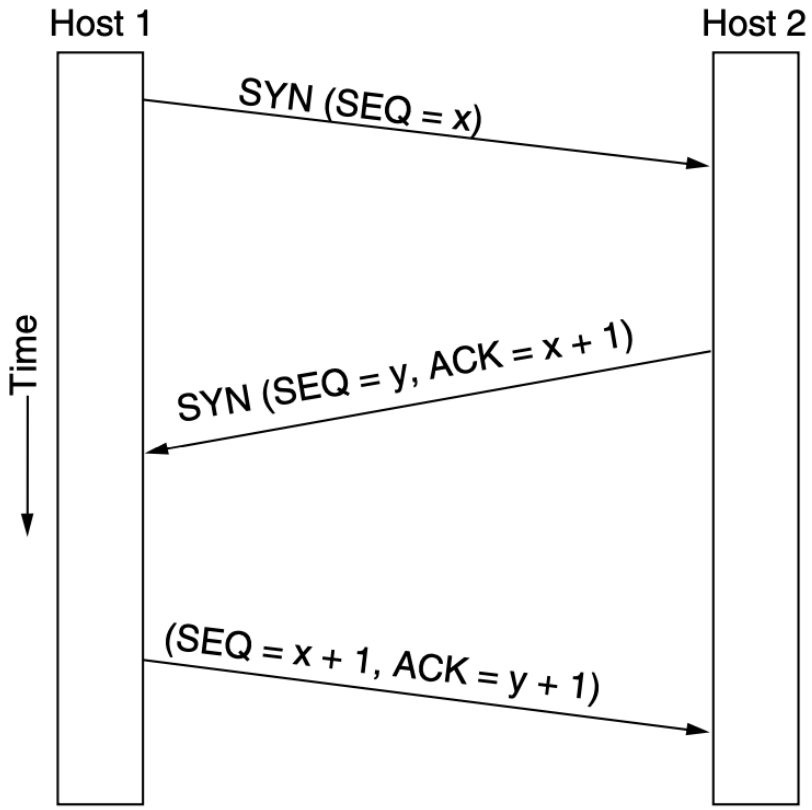
The TCP Segment Header…(4)

- Window size - define how many bytes may be send.
- *Urgent pointer* is used to indicate a byte offset from the current sequence number at which urgent data are to be found.
- The *Options* field provides a way to add extra facilities not covered by the regular header.

TCP Connection Establishment

- The server wait for incoming connection by executing LISTEN and ACCEPT primitives.
- The client execute CONNECT primitive specifying the IP address and port which it need to connect.
- The CONNECT primitive sends a TCP segment with SYN bit on and ACK bit off and wait for the response.
- At the destination the segment is checked if there is a process that has done a LISTEN on the port given in the destination port field.
- If so the connection established and if not it will reply with RST bit on to reject the connection.

TCP Connection Establishment...(2)



Ref 1: pg 561

TCP Connection Release

- To release a connection
  - One party can send FIN bit set, saying that there is not data to be sent.
  - When the FIN is acknowledge, that direction is shutdown for new data.
  - But data can flow to other direction.
  - When both directions are shutdown the connection closed.
  - TCP need four segments to release a connection.
    - One FIN and one ACH each direction.

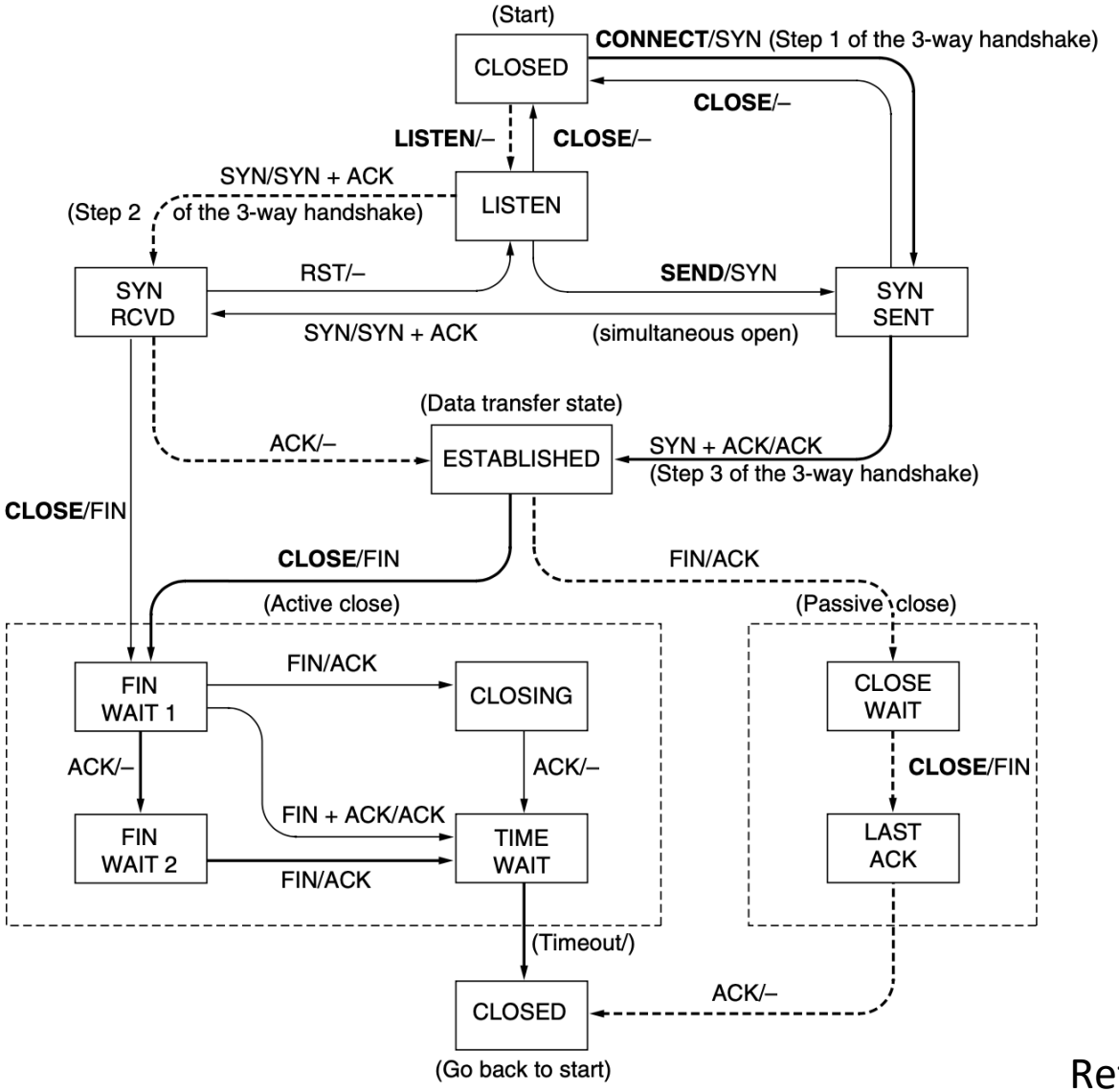
TCP Connection Management Modeling

- The steps required to establish and release connection can be represented in a finite state machine. Following are the states

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIME WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

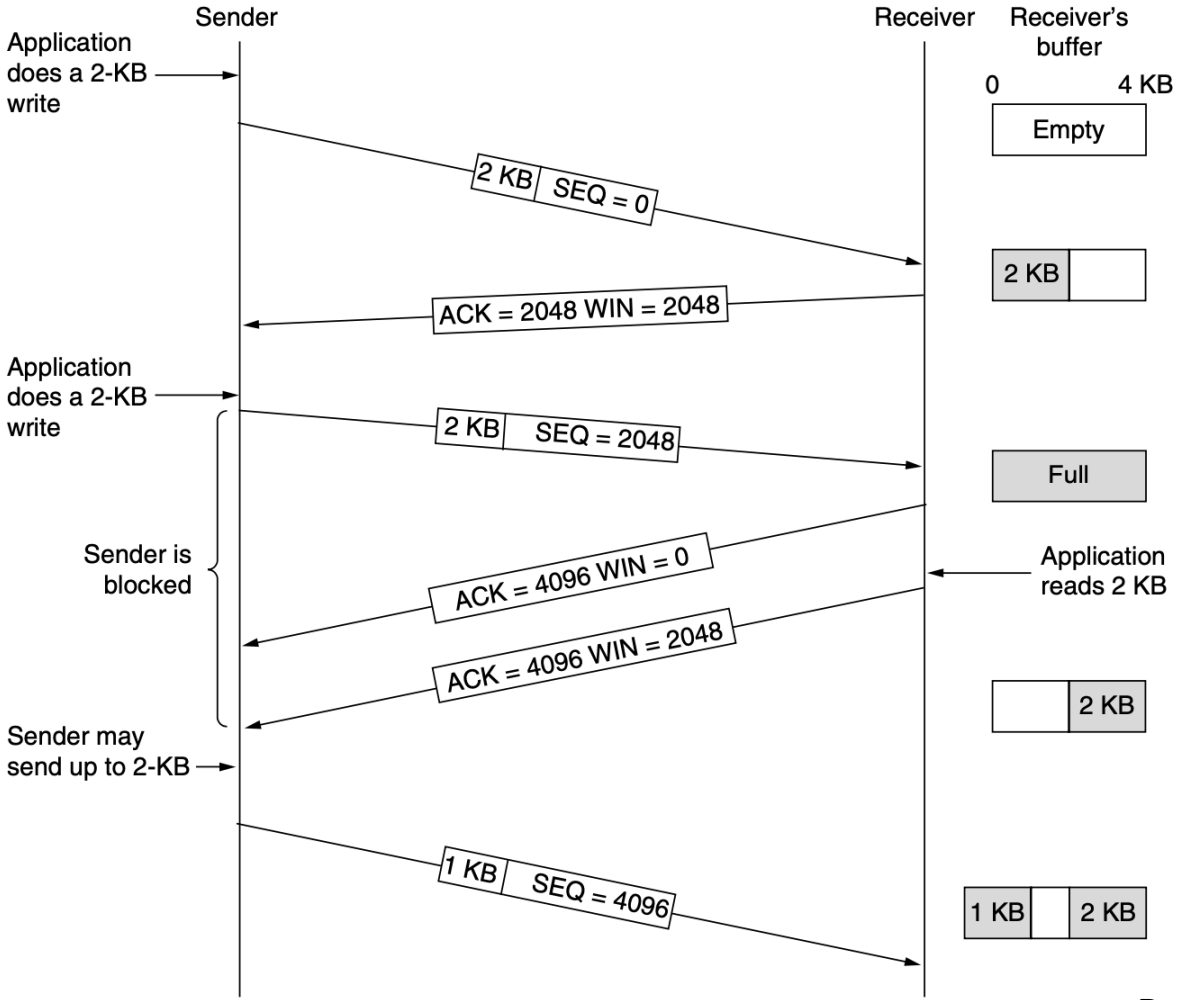
Ref 1: pg 563

TCP Connection Management Modeling...(2)



Ref 1: pg 564

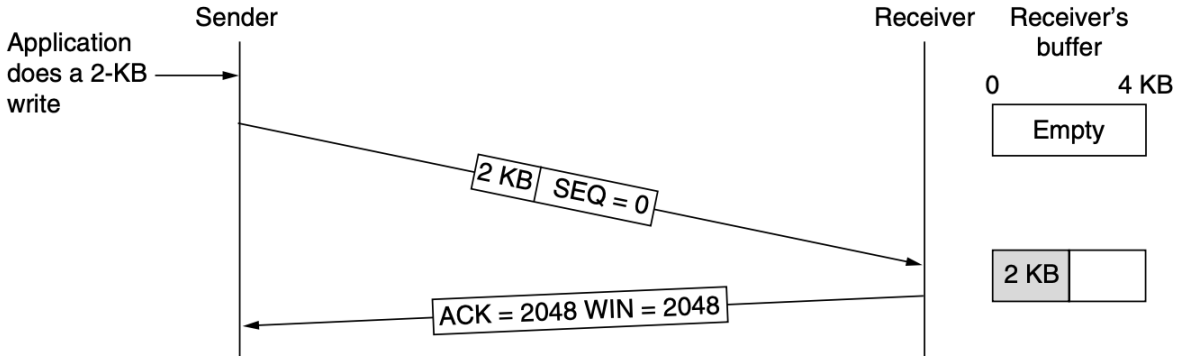
TCP Sliding Window



Ref 1: pg 565

TCP Sliding Window…(2)

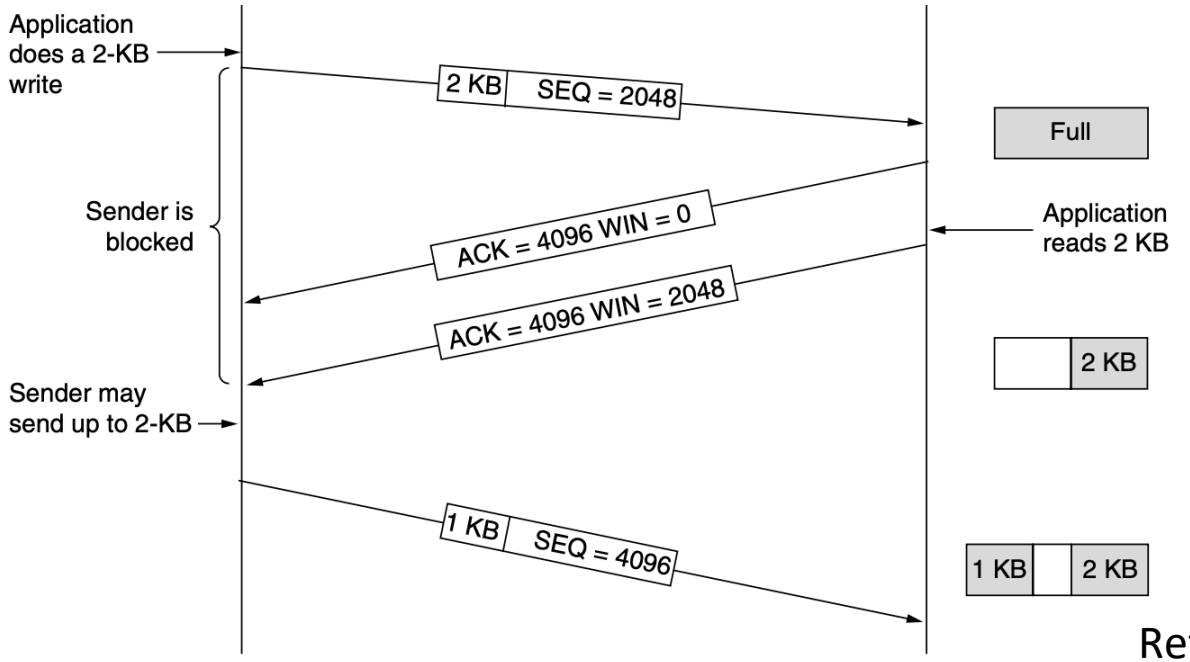
- Suppose the receiver has 4096 byte buffer.
- Sender sends 2048 bytes segment and after receiving successfully, the receiver acknowledge the segment.
- Now the receiver has 2048 byte remaining in the buffer.
  - It will advertise 2048 as the remaining space (Window).



Ref 1: pg 565

TCP Sliding Window…(3)

- Again the sender will 2048 bytes in the next segment and receiver will acknowledge after successfully receiving the segment.
- Then the buffer is full and receiver mention the window is 0 (zero)



Ref 1: pg 565

TCP Sliding Window…(4)

- When the buffer is 0, the sender will not send segments unless,
  - If there are any urgent data to be send (eg: to kill the process of running of the receiver)
  - Or the sender may send a 1-byte segment to force the receiver to re-announce the next byte expected and the window size (window probe).

TCP Sliding Window…(5)

- Delayed Acknowledgements
  - Delay the acknowledgements and window updates up to 500 msec.
  - This will reduce the number of short packets send by the sender.
- In Nagle's algorithm, when data come into the sender in small pieces, just send the first piece and buffer all the rest until the first piece is acknowledged. Then send all the buffered data in one TCP segment and start buffering again until the next segment is acknowledged.

TCP Sliding Window…(6)

- The silly window syndrome
  - Data are passed to the sending TCP entity in large blocks, but an interactive application on the receiving side reads data only 1 byte at a time (Remaining space in the buffer).
    - A solution for this is not send a window update util receiver can handle substantial amount of data.
- Receiver has to handle the segments which receive out of order. It will buffer the data until it can be passed up to the application in order.

TCP Sliding Window…(7)

- Cumulative acknowledgement
  - Acknowledgements can be sent only when all the data up to the byte acknowledged have been received.
    - If the receiver has received segments 0, 1, 2, 3, 5, 6, 7
      - It can acknowledge up to segment 3.
      - When sender's timeout, it retransmit the segment 4.
      - Receiver has buffered from segment 5 to 7.
      - When it receive segment 4, it can acknowledge up to segment 7.

TCP Congestion Control

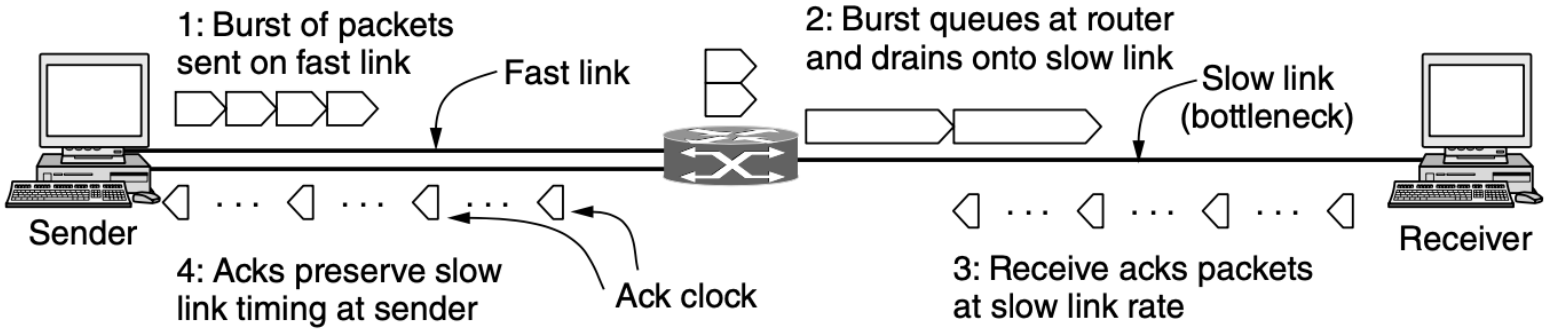
- When the load offered to any network is more than it can handle, congestion builds up.
- TCP handle this condition by using a window and with packet loss as the binary signal.
- The size of the *congestion window* is the number of bytes the sender may have in the network at any time.
    - The corresponding rate is the window size divided by the round-trip time of the connection.
- Congestion window is maintain parallel to the flow control window (sliding windows discusses earlier).
- TCP will stop sending data if either the congestion or the flow control window is temporarily full.

TCP Congestion Control…(2)

- Loss of packets is a single of the congestion.
- It is difficult to build a router that does not drop packets when it is overloaded.
- Transmission errors leads to packet loss. But congestion signal depends on transmission errors is relatively rare. Because,
  - Wireless technologies such as 802.11 have their own retransmission mechanism at the link layer.
  - It is rare on other links because wires and optical fibers typically have low bit-error rates.

TCP Congestion Control…(3)

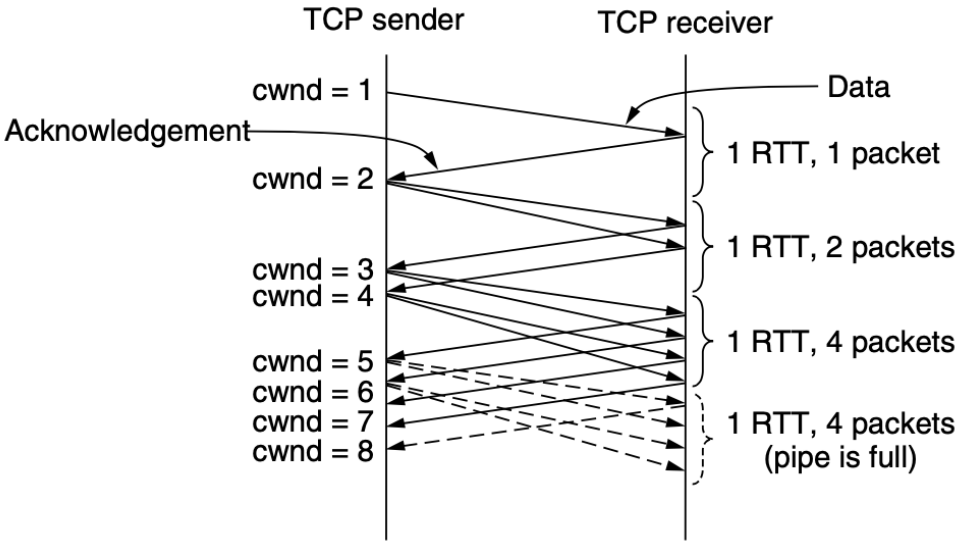
- Sender is using a fast link but the receiver is in a slow network.
- At the router they are queued because it takes longer time to send over a slow link.
- After receiving the data at the receiver, it send the acknowledgement.
- The acknowledgements return to the sender at about the rate that packets can be sent over the slowest link in the path.



Ref 1: pg 573

TCP Congestion Control…(4)

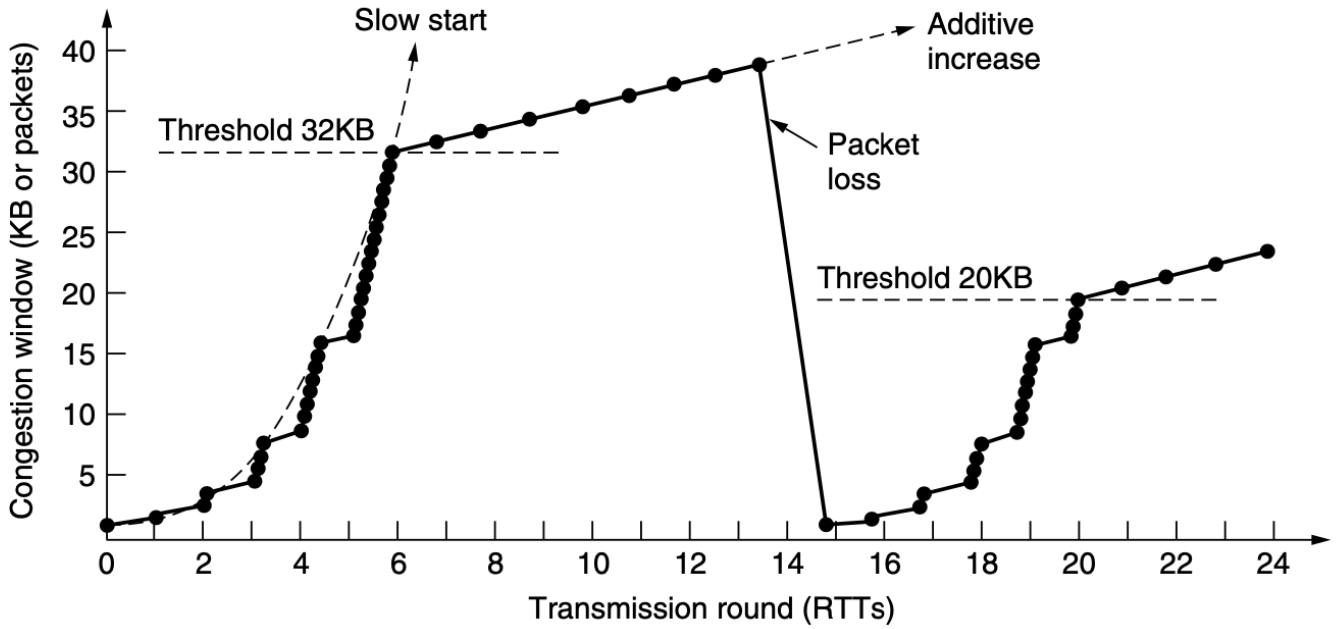
- Slow start algorithm
  - In the first round-trip time, the sender injects one packet into the network (and the receiver receives one packet).
  - Two packets are sent in the next round-trip time, then four packets in the third round-trip time.



Ref 1: pg 575

TCP Congestion Control…(5)

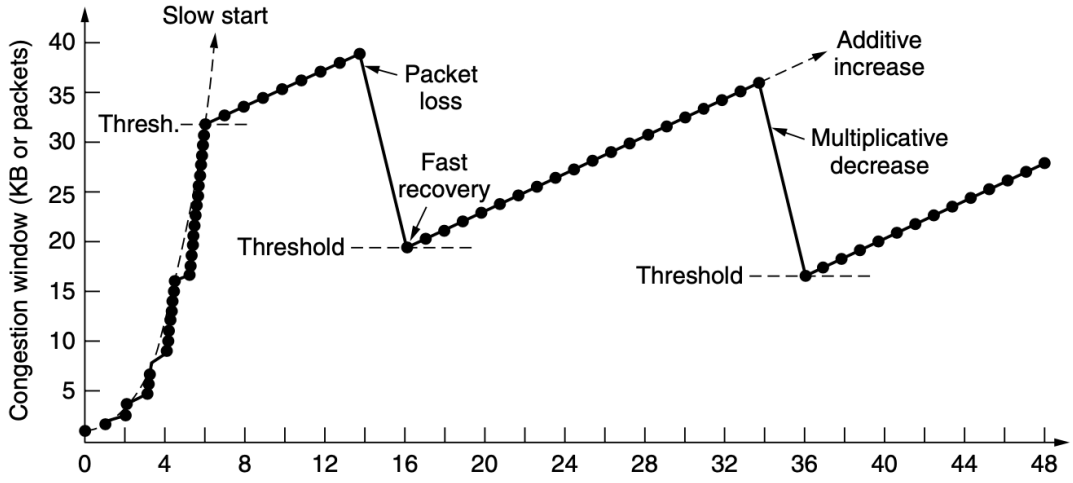
- Slow start increased up to the threshold of 32KB.
- Around 13 packet losses and packet retransmitted.
- Threshold dropped to 20KB and slow start initiated again.



Ref 1: pg 578

TCP Congestion Control…(6)

- Fast recovery
  - Instead of dropping the congestion window all the way down, it set the congestion window at a new threshold or set to the half of the value of congestion window when retransmission occurs.
  - This is sawtooth pattern.



Ref 1: pg 578

References

- Ref 01. Computer Networks by Andrew Tannenbaum, 5th edition, Pearson