Chapter 6 Transport Layer

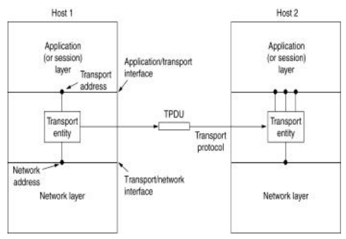
Prepared By:
Prof. Vishal A. Polara
Assistant Professor
Information Technology Department
Birla Vishvakarma Mahavidyalaya Engineering College

Outline

- The transport service: Services provided to the upper layers, Transport service primitives, Socket
- Elements of transport protocols: Addressing, Connection establishment, Connection release, Flow control, Multiplexing, Crash recovery.
- The transport protocol: UDP, TCP

6.1 Service provided to upper layer

The network, transport, and application layers.



Service provide to upper layer

- The goal of transport layer is to provide efficient, reliable and cost effective service to its users normally in application layer.
- To do this it will use service of network layer like hardware/software.
- Transport code runs on user machine while network on routers.
- User cannot have control on network service while user have control on transport layer service. Lost packet can be detected using transport layer. So we can have reliable communication.
- Bottom layer are known as transport service provider and upper layer are known as transport service user.

6.2 Transport service primitives

- To provide access top transport service, transport layer must provide some operations to application programs.
- Each transport service has its own interface
- It is used to provide reliable service compared to network service because network service is on live networks.
- Connection oriented service is used to hide network imperfection that shows an error free bit stream.
- Client server computing and multimedia are connectionless services.
- Transport services are convenient and easy to use.

6.2 Transport service primitives

- When client want to communicate with server , server start first and add listen primitive and client send connect primitives in order to make connection.
- Client then send message to another client or server with send and receive primitive.
- Then it will send disconnect primitive to server or client.

6.2 Transport Service Primitives

Primitive	Packet 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

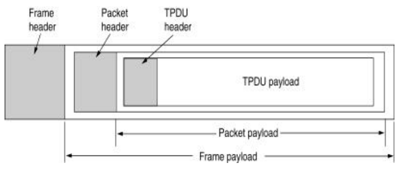
The primitives for a simple transport service.

6.2 Transport service primitives

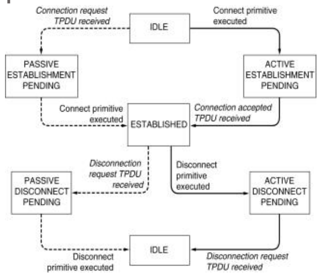
- Transport entity to transport entity message follows TDPU(transport protocol data unit) format.
- When frame arrive it follows frame header with packet payload of network entity.
- The network entity processes the packet header and passes the contents of the packet payload up to the transport entity.
- Client send connect TPDU to server then transport entity check server is blocked on a listen mode. It then unblock the server and sends a connection accepted TPDU to client and then client unblocked and connection is established.

Transport Service Primitives (2)

The nesting of TPDUs, packets, and frames.



Transport Service Primitives (3)

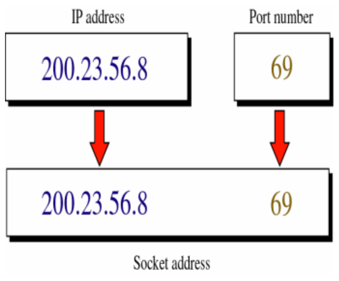

Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

6.3 Network Socket

- A **network socket** is an endpoint of an **inter-process communication flow** across a computer network. Today, most communication between computers is based on the **Internet Protocol**; therefore most network sockets are **Internet sockets**.
- A **socket API** is an application programming interface(API), usually provided by the **operating system**, that allows application programs **to control and use network sockets.** Internet socket APIs are usually based on the **Berkeley sockets** standard.
- A **socket address** is the combination of an **IP address and a port number,** much like one end of a telephone connection is the combination of a phone number and a particular extension. Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread.

How to create socket address…



Berkeley Sockets

The socket primitives for TCP.

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Berkeley socket

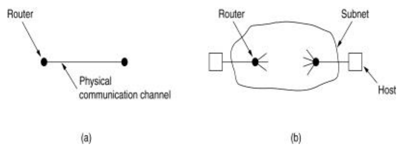
- The upper four primitive are server and below four are client primitive.
- When socket is created it does not have network address it will get when it using BIND primitive. Now remote client can connect it.
- Next is LISTEN call which allocated space to queue incoming calls that several client try to connect at same time.
- Now server executes and ACCEPT primitive.
- At client side CONNECT primitive will use to make connection here no bind is require. And send and receive is used to send and receive message.

6.4 Elements of Transport Protocols

- Addressing
- Connection Establishment
- Connection Release
- Flow Control and Buffering
- Multiplexing
- Crash Recovery

Transport Protocol

- In data link layer routers are communicated directly via physical link.
- While in transport layer physical link is replaced by subnet.



(a) Environment of the data link layer.
(b) Environment of the transport layer.

- Transport protocol similar to data link protocols.
- Both do error control and flow control

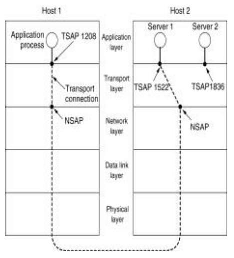
	Data link layer	Transport layer
Communication	directly via physical channel	over the entire network
Addressing	no need to specify address. Just select outgoing line	explicit addressing of destination is required
Connection establishment	over a wire is simple	more complicated
Delay	frame either arrives or lost, not stored and delayed	packets might be stored for seconds and delivered later
Buffering and flow control	simpler	more complicated; large and dynamic number of simultaneous connections

6.4.1 Addressing

- In transport layer TSAP(transport layer access point) and NSAP (network layer access point) is used for addressing.
- Application layer connect client and server through TSAP it then run through NSAP on each host.
- For example host 2 it self to TSAP 1522 to wait for an incoming call now call such as listen might be used to make request for outside the host.
- At host 1 it will send connect request with TSAP 1208 as the source and 1522 as destination. This will make connection.
- The application process then sends over a request for the same.
- The time server process responds with the current time.
- The transport connection is then released.

Addressing

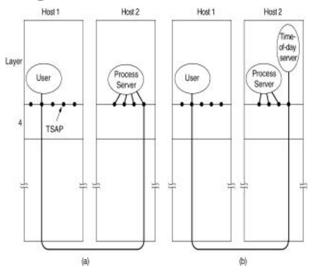
TSAPs, NSAPs and transport connections.



Addressing

- It is known as the initial connection protocol.
- Instead of every conceivable server listening at a well-known TSAP, each machine that wishes to offer services to remote users has a special process server that acts as a proxy for less heavily used servers.
- It listens to a set of ports at the same time, waiting for a connection request.
- Potential users of a service begin by doing a CONNECT request, specifying the TSAP address of the service they want. If no server is waiting for them, they get a connection to the process server.
- Problem here server has to ready on fly basis solution is **directory server** or **name server.** Client will register with particular service only like mail or file.

Addressing



How a user process in host 1 establishes a connection with a time-of-day server in host 2.

6.4.2 Connection Establishment

- Sounds easy; surprisingly tricky!
- Just send REQUEST, wait for ACCEPTED?
- Can lose, delay, corrupt, duplicate packets when acknowledgment arrives late or lost
- Duplicate may transfer bank money again!
- Protocols must work correct all cases
- Implemented efficiently in common cases
- Main problem is delayed duplicates
- Cannot prevent; must deal with (reject)

Connection Establishment

- A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person, and then releases the connection.
- Unfortunately, each packet in the scenario is duplicated and stored in the subnet. After the connection has been released, all the packets pop out of the subnet and arrive at the destination in order, asking the bank to establish a new connection, transfer money (again), and release the connection.
- The bank has no way of telling that these are duplicates. It must assume that this is a second, independent transaction, and transfers the money again.

Connection Establishment

- **Solutions for delayed duplicates**
- Not reuse transport address (TSAP) create new all time.
- **difficult to connect to process //problem**
- Give each connection unique ID
- seq # chosen by initiating party
- update table listing obsolete connections
- check new connections against table
- **requires maintain certain amount of history //problem**
- if machine crashes, no longer identify old con

Connection Establishment

- To simplify problem, restrict packet lifetime
  - restricted network design: prevent looping
  - hop counter in each packet: -1 at each hop
  - timestamp in each packet: clock must be synced
- Must also guarantee ACKs are dead
- Assume a value T of max packet lifetime
- *T sec after packet sent, sure traces are gone*
- In the Internet, T is usually 120 seconds

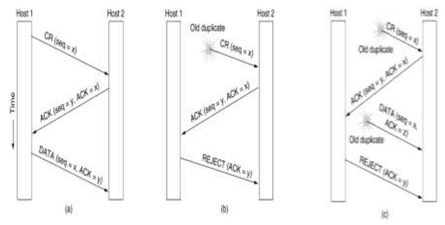
25 Prof. Vishal A. Polara

Connection Establishment

- **New method with packet lifetime bounded**
- Label segments with seq # not reused in T
- T and packet rate determine size of seq #s
- 1 packet w given seq # may be outstanding
- Duplicates may still occur, but discarded dst
- Not possible to have delayed duplicate old
- packet with same seq # accepted at dest
- **How to deal with losing memory after crash?**
- Each host has time-of-day clock
- clocks at different host need not be synced
- binary counter increments uniform intervals
- no. of bits must be ≥ of seq #
- clock must be running even if host goes down

26 Prof. Vishal A. Polara

Connection Establishment



Three protocol scenarios for establishing a connection using a **three-way handshake.** CR denotes CONNECTION REQUEST.
(a) Normal operation,
(b) Old CONNECTION REQUEST appearing out of nowhere.
(c) Duplicate CONNECTION REQUEST and duplicate ACK.

27 Prof. Vishal A. Polara

- Normal Procedure
- H1 choses initial s# *x*
- H2 replies
  - ACKs *x*
  - announce own s# y
- H1replies
  - ACKs *y*
  - with 1st data segment

28 Prof. Vishal A. Polara

- Delayed duplicate CR
  - H2 sends ACK to H1
  - H1 rejects
  - H2 knows it was tricked
- Worst case
- DD CR, old ACK floating
  - H2 gets delayed CR, replies
  - H1 rejects
  - H2 gets old DATA, discards
  - *(z received instead of y)*

29 Prof. Vishal A. Polara

Connection Establishment

- **Three way Handshake:**
  - Check with other peer that con req is current
  - Used in TCP, with 32-bit seq #
  - Clock not used in TCP; attacker can predict
- (a) Host 1 chooses a sequence number, x, and sends a CONNECTION REQUEST TPDU containing it to host 2.
- Host 2 replies with an ACK TPDU acknowledging x and announcing its own initial sequence number, y.
- Finally, host 1 acknowledges host 2's choice of an initial sequence number in the first data TPDU that it sends.
- (b) the first TPDU is a delayed duplicate CONNECTION REQUEST from an old connection. This TPDU arrives at host 2 without host 1's knowledge. Host 2 reacts to this TPDU by sending host 1 an

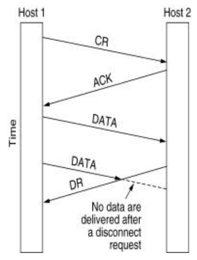
30 Prof. Vishal A. Polara

Connection Establishment

- ACK TPDU, in effect asking for verification that host 1 was indeed trying to set up a new connection. When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.
- (c) host 2 gets a delayed CONNECTION REQUEST and replies to it. At this point it is crucial to realize that host 2 has proposed using y as the initial sequence number for host 2 to host 1 traffic, knowing full well that no TPDUs containing sequence number y or acknowledgements to y are still in existence.
- When the second delayed TPDU arrives at host 2, the fact that z has been acknowledged rather than y tells host 2 that this, too, is an old duplicate.
- The important thing to realize here is that there is no combination of old TPDUs that can cause the protocol to fail and have a connection set up by accident when no one wants it.

31 Prof. Vishal A. Polara

6.4.3 Connection Release



Abrupt disconnection with loss of data.

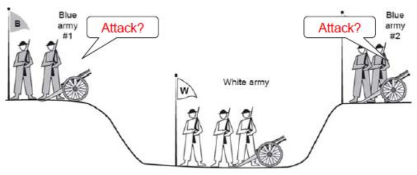
32 Prof. Vishal A. Polara

Connection Establishment

- Easier than establish, However, some pitfalls
- There are two types of release Asymmetric and symmetric
- Asymmetric release
  - It is like telephone system one party hand up the connection broken
  - each con term separately
  - abrupt; may cause data loss
  - better protocol needed
- Symmetric release
  - Each direction is released independently (two unidirectional line), used only a fixed amount of data with each process
  - Can receive data after sending DISCONNECT
  - H1: I am done, are you done too?
  - H2: I am done too, goodbye
  - Two-army problem: unreliable channel

33 Prof. Vishal A. Polara

- Two-army problem
- each blue army < white army, but together are larger
- need to sync attack
- however, only com channel is the valley (unreliable)
- 3-way handshake? B1 can't know ACK arrived
- making 4-way handshake doesn't help either



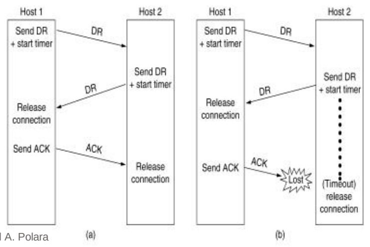
34 Prof. Vishal A. Polara

Connection Release

Four protocol scenarios for releasing a connection.

(a) Normal case of a three-way handshake.

(b) final ACK lost.

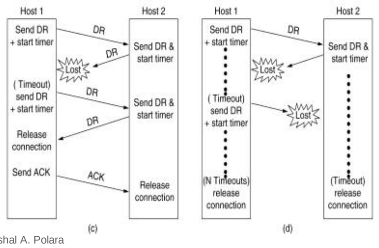


35 Prof. Vishal A. Polara

Connection Release

(c) Response lost.

(d) Response lost and subsequent DRs lost.



36 Prof. Vishal A. Polara

Connection Release

- Protocol usually suffices; can fail in theory
- after N lost attempts; half open connection
- Not allowing give up, can go on forever
- To kill half open connections, automatically
- disconnect if no received segments in X sec
- Must have timer reset after each segment
- Send dummy segments to keep con alive
- TCP normally does symmetric close, with
- each side independently close ½ con w FIN

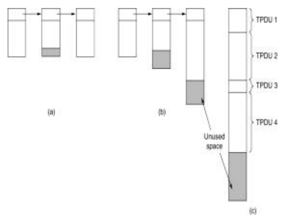
6.4.4 Flow Control and Buffering

- Flow control is require at data link layer and transport layer.
- both layers a sliding window or other scheme is needed on each connection to keep a fast transmitter from overrunning a slow receiver.
- The main difference is that a router usually has relatively few lines, whereas a host may have numerous connections. So it is not possible to implement at transport layer.
- In data link layer both sender and receiver are required to dedicate MAX_SEQ + 1 buffers to each line, half for input and half for output.
- For a host with a maximum of, say, 64 connections, and a 4-bit sequence number, this protocol would require 1024 buffers.

Flow Control and Buffering

- (a) If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short TPDU arrives.
- If the buffer size is chosen less than the maximum TPDU size, multiple buffers will be needed for long TPDUs, with the attendant complexity.
- (b) user variable size buffer, The advantage here is better memory utilization, at the price of more complicated buffer management.
- (c) A third possibility is to dedicate a single large circular buffer per connection.
- This system also makes good use of memory, provided that all connections are heavily loaded, but is poor if some connections are lightly loaded.
- For low-bandwidth busty traffic, it is better to buffer at the sender, and for high bandwidth smooth traffic, it is better to buffer at the receiver.

Flow Control and Buffering



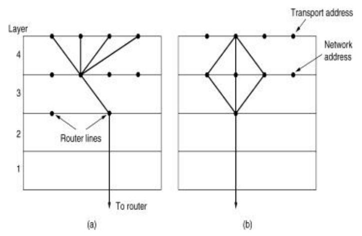
(a) Chained fixed-size buffers. (b) Chained variable-sized buffers.
(c) One large circular buffer per connection.

6.4.5 Multiplexing

- Multiplexing several conversations onto connections, virtual circuits.
- In the transport layer the need for multiplexing can arise in a number of ways. For example, if only one network address is available on a host, all transport connections on that machine have to use it.
- When a TPDU comes in, some way is needed to tell which process to give it to. This situation, called **upward multiplexing.**
- **Downward multiplexing** is used to share available bandwidth. When user need more bandwidth it will take bandwidth of unutilized stations.

Multiplexing

(a) Upward multiplexing. (b) Downward multiplexing.



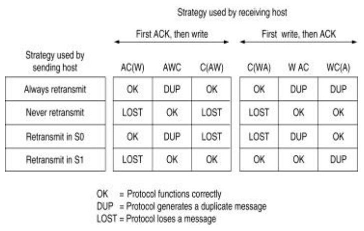
6.4.6 Crash Recovery

- If hosts and routers are subject to crashes, recovery from these crashes becomes an issue.
- If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward.
- If the network layer provides datagram service, the transport entities expect lost TPDUs all the time and know how to cope with them.
- If the network layer provides connection-oriented service, then loss of a virtual circuit is handled by establishing a new one and then probing the remote transport entity to ask it which TPDUs it has received and which ones it has not received. The latter ones can be retransmitted.

- If server crash what happen?
- In an attempt to recover its previous status, the server might send a broadcast TPDU to all other hosts, announcing that it had just crashed and requesting that its clients inform it of the status of all open connections.
- Each client can be in one of two states: one TPDU outstanding, S1, or no TPDUs outstanding, S0. Based on only this state information, the client must decide whether to retransmit the most recent TPDU.

Crash Recovery

Different combinations of client and server strategy.



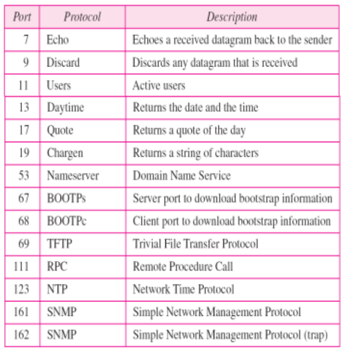
	Strategy used by receiving host					
	First ACK, then write			First write, then ACK		
Strategy used by sending host	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK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

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

6.5 USER DATAGRAM PROTOCOL (UDP)

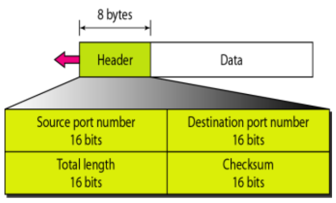
The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.

Table *Well-known ports used with UDP*



Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Figure *User datagram format*

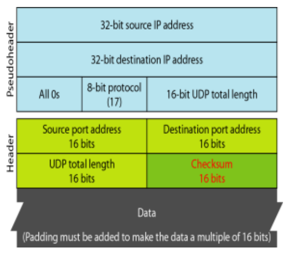


__Note__: UDP length = IP length – IP header's length

- UDP packets, called user datagram's have a fixed size header of 8 bytes.
- **Source Port Number:** it is a 16 bit long which is assign to process running on source host.
- **Destination Port Number:** it is a 16 bit long which is assign to process running on destination host.
- **Length:** it is a 16 bit that defines the total length of the user datagram, header plus data.
- A user datagram is encapsulated in IP datagram. There is a field in the IP datagram that defines the total length.
- **Checksum:** this field is used to detect errors over the entire user datagram.

- The checksum includes three sections: a pseudoheader, the UDP header and the data coming from the application layer.
- the pseudoheader is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields with 0s.
- If the checksum does not include the pseudoheader a user datagram may arrive safe.
- The protocol filed is added to ensure that the packet belongs to UDP and not to other transport layer protocols. The value of protocol field is 17. if this value is changed checksum will changed.

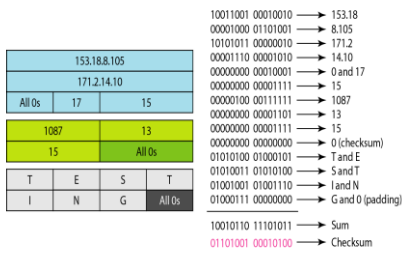
Figure *Pseudoheader for checksum calculation*


Pseudoheader
Header
32-bit source IP address
32-bit destination IP address
All 0s
8-bit protocol (17)
16-bit UDP total length
Source port address 16 bits
Destination port address 16 bits
UDP total length 16 bits
Checksum 16 bits
Data
(Padding must be added to make the data a multiple of 16 bits)

Example

Figure shows the checksum calculation for a very small user datagram with only 7 bytes of data. Because the number of bytes of data is odd, padding is added for checksum calculation. The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP.

Figure *Checksum calculation of a simple UDP user datagram*



UDP Operations & Uses

Operations:

- Connectionless Services – can send only short messages
- No Flow and Error Control so no window mechanism
- Encapsulation and Decapsulation
- Queuing

Uses:

- For a process that requires simple request-response communication
- For a process with internal flow and error control mechanisms
- For multicasting
- For management processes such as SNMP
- For some route updating protocols such as routing information protocol

6.6 TCP (Transmission Control Protocol

TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

Table *Well-known ports used by TCP*

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

- **Services Offered by TCP:**
- **1. Process to Process Communication:** It provide process to process communication using port.
- **2. Stream Delivery Service:** It is stream oriented protocol. In UDP a process sends messages with predefined boundaries, to UDP for delivery. UDP adds its own header to each of these messages and delivers them to IP for transmission.
- In TCP , it allows sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. It creates and imaginary tubes that carries their data across the internet.
- **3. Sending and receiving buffers:** TCP needs buffer as sending and reading is not done at same speed. There are two buffers. It is implemented using circular array of 1 byte locations.
- Sender side buffer has three types of chambers. White section contains empty chambers, the gray area holds bytes that require ack. Colored area contains bytes need to send.

Figure *Stream delivery*

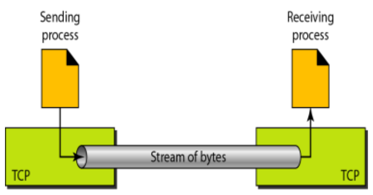
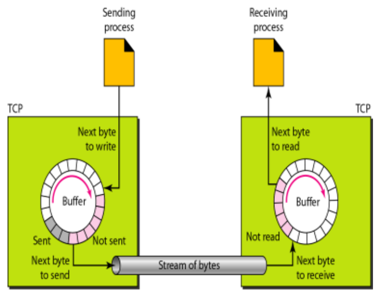
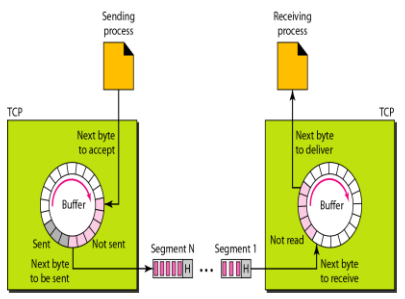


Figure *Sending and receiving buffers*



- After byte in the grey chambers are acknowledged the chambers are recycled and available for use by the sending process.
- The operation of the buffer at the receiver site is simpler. The circular buffer is divided into two areas.
- The white area contains empty chambers to be filled by bytes received from the network.
- The colored sections contain received bytes that can be read by the receiving process. When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.
- **4. Segments:** TCP groups a number of bytes together into a packet called a segment. TCP adds a header to each segment and delivers the segment to the IP layer for transmission. It is not necessary to have segment of same size.

Figure *TCP segments*



- **5. Full Duplex communication:** TCP offers full duplex service in which data can flow in both directions at the same time.
- **6. Connection oriented service:** TCP establish virtual connection between two host not physical. Host would like to transmit data will do following.
  - The two TCPs establish a connection between them
  - data are exchanged in both directions
  - The connection is terminated.
- **7. Reliable Service:** TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

TCP Features – *Numbering System*

- The bytes of data being transferred in each connection are numbered by TCP.
- It has sequence number and the acknowledgment number.
- The numbering starts with a randomly generated number.
- The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.
- The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.

TCP Features

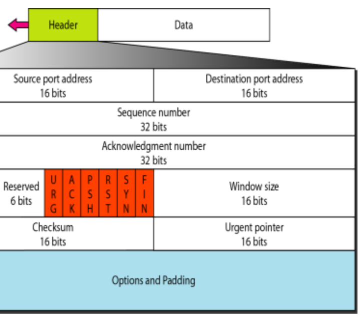
- Flow control
- Error control
- Congestion Control

Example

The following shows the sequence number for each segment:

Segment 1	➡	Sequence Number: 10,001 (range: 10,001 to 11,000)
Segment 2	➡	Sequence Number: 11,001 (range: 11,001 to 12,000)
Segment 3	➡	Sequence Number: 12,001 (range: 12,001 to 13,000)
Segment 4	➡	Sequence Number: 13,001 (range: 13,001 to 14,000)
Segment 5	➡	Sequence Number: 14,001 (range: 14,001 to 15,000)

Figure *TCP segment format*



- **Source Port Address:** It is a 16 bit field defines the port number of the source program.
- **Destination port address:** it is a 16 bit field defines the port number of destination program.
- **Sequence number:** this 32 bit field defines the number assigned to the first byte of data contained in this segment.
- **Acknowledgment numbers**: this 32 bit field defines the byte number that the receiver of the segment is expecting to receive from the other party.
- **Header Length:** This 4 bit field indicates the number of 4 byte words in the TCP header. It is from 20 to 60 bytes. If 5(5*4=20).
- **Reserved:** this is a 6 bit field reserved for future used.
- **Control:** This filed defines 6 different control bits or flags.

Figure *Control field*

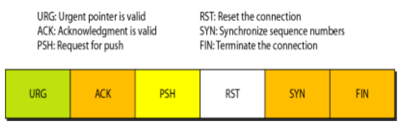

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection
URG
ACK
PSH
RST
SYN
FIN

Table *Description of flags in the control field*

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

- This control bits enable flow control, connection establishment and termination, connection abortion and the mode of data transfer in TCP.
- **Window Size:** This field defines the size of the window in bytes that the other party must maintain. It is 16 bits field.
- **Checksum:** this 16 bit field contain checksum.
- **Urgent Pointer:** It is 16 bit field which is valid only if the urgent flag is set is used when the segment contains urgent data.
- **Options:** there can be up to 40 bytes of optional information in the TCP header.

Connection Establishment

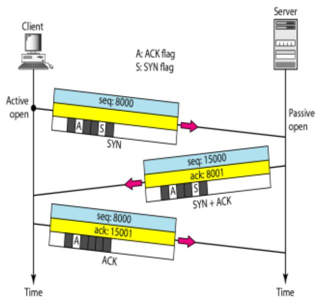
- TCP transmit data in full duplex mode.
- It use Three way handshaking.
- In this process server start first and tell its TCP that it is ready. This is called **passive open**.
- The client program issues a request for an **Active open**.
- Client sends the first segment, a SYN segment in which only the SYN flag is set. When data transfer start this number will increment.
- The server sends the second segment a SYN+ACK segment, with 2 flag bit set. It does not carry data but consume one sequence number.
- The client send third segment this is just an Ack segment. The ACK segment does not consume any sequence numbers.

Connection Establishment

- **SYN flooding attack:** It has a problem of SYN flooding attack.
- When malicious attacker sends a large number of SYN segments to a server, pretending that each of them is coming form a different client by faking the source IP address in the datagram.
- The TCP server then sends the SYN+ACK segments to the fake clients, which are lost.
- If during this short time the number of SYN segments is large, the server eventually runs out of resources and may crash. It is also known as **a denial of service** attack.

- A SYN segment cannot carry data, but it consumes one sequence number.
- A SYN + ACK segment cannot carry data, but does consume one sequence number.
- An ACK segment, if carrying no data, consumes no sequence number.

Figure *Connection establishment using three-way handshaking*



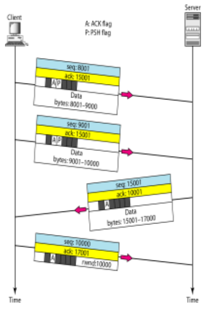
73 Prof. Vishal A. Polara

Data transfer

- After data connection bidirectional data transfer take place.
- The client and server can both send data and acknowledgments.
- The acknowledgment is piggy backed with the data.
- The client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.
- When delay is not allowed in data it can be handled by push data operation. Sending application program send a push request operation that data must reach immediately.
- Urgent data is used by receiver when it would like to receive data urgent. Urgent field will set.

74 Prof. Vishal A. Polara

Figure *Data transfer*



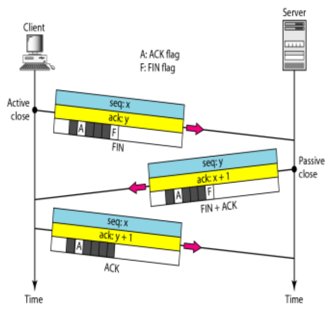
75 Prof. Vishal A. Polara

Connection Termination

- Any of two parties can close the connection.
- Client TCP after receiving a close command from the client process sends the first segment , a FIN segment in which the FIN flag is set. It include last chunk of data sent by the client or a control segment. It consume one sequence number if not data.
- The server TCP then send FIN + ACK segment to confirm the receipt of FIN.
- It consume one sequence number if no data.
- The client TCP sends the last segment an ACK segment , to confirm the receipt of the FIN segement.
- It is also possible to **half close** the connection. One end stop sending data while other is still receive.
- Sorting is an example of half close where client send data to sort and then close sending connection but still receive data.

76 Prof. Vishal A. Polara

Figure *Connection termination using three-way handshaking*



77 Prof. Vishal A. Polara

Thank You

78 Prof. Vishal A. Polara