Computer Network(CSC 503)

Shilpa Ingoley

Lecture 32

Transport Layer

- It is the heart of the whole protocol hierarchy
- It is located between the application layer and the network layer.
- It provides services to the application layer and receives services from the network layer.
- Its task is to provide:
  - Reliable, cost-effective data transport from the source machine to the destination machine, independently of the physical network or networks currently in use.

Contd…

- The ultimate goal of the transport layer is to provide **efficient, reliable, and cost-effective** service to its users.
- To achieve this goal, the transport layer makes **use of the services provided by the network layer.**
- The hardware and/or software within the transport layer that does the work is called the **transport entity.**
- There are **two** types of transport service.
  - Connection Oriented Transport Service
  - Connectionless Transport Service
- Connections have three phases: **establishment, data transfer, and release.**
- Addressing and flow control are also similar in both layers.

Need for Transport Layer

- **If the transport layer service is so similar to the network layer service, why are there two distinct layers?**
- **Why is one layer not adequate?**
- The transport code runs entirely on the **users' machines**, but the network layer mostly runs on the routers, which are operated by the carrier.
- What happens if the network layer offers inadequate service? Suppose that it frequently **loses packets**? What happens if **routers crash** from time to time?
- The users have **no real control over the network layer**, so they cannot solve the problem of poor service by using better routers or putting more error handling in the data link layer

Contd…

In a connection-oriented subnet:

- A transport entity is informed halfway through a long transmission that its network **connection has been abruptly terminated**,
- What happens to the data currently in transit ?
- It can set up a new network connection to the remote transport entity.
- Using this new network connection, it can send a query to its **peer asking which data arrived and which did not, and then pick up from where it left off**.
- In this case the existence of the transport layer makes it possible for the transport service to be **more reliable** than the underlying network service.
- **Lost packets** and mangled data can be detected and compensated for by the transport layer.

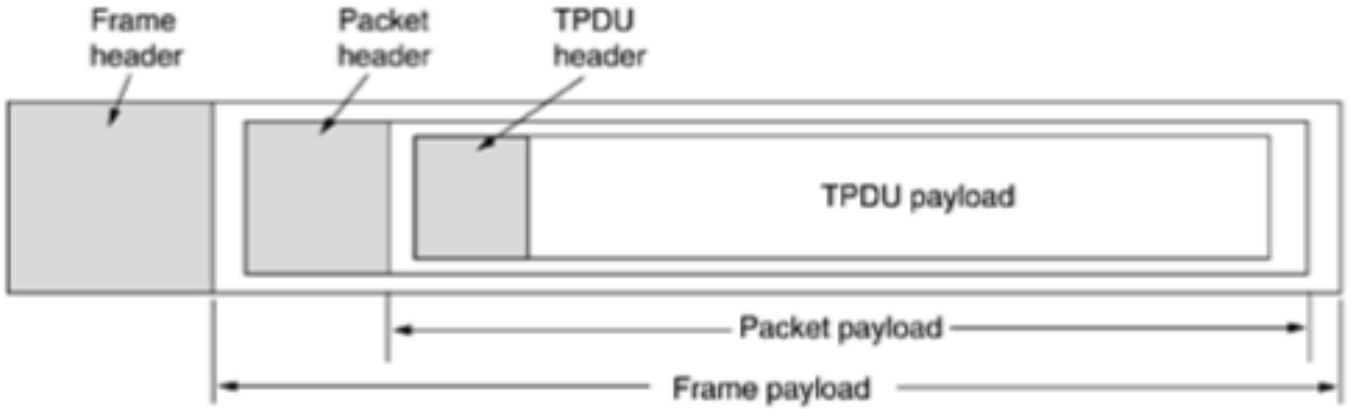
Contd…

- 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 subnet interfaces and unreliable transmission**
- These transport service primitives can be implemented as calls to **library procedures** in order to make them **independent of the network service primitives**. The network service calls may vary considerably from network to network
- **The bottom four layers can be seen as the transport service provider, whereas the upper layer are the transport service user**
- **Transport layer forms the major boundary between the provider and user of the reliable data transmission service.**

Transport Protocol Data Unit (TPDU)

- TPDUs exchanged by the transport layer are contained in packets exchanged by the **network layer**.
- In turn, packets are contained in frames exchanged by the data link layer.
- When a frame arrives, the data link layer processes the frame header and passes the contents of the frame payload field up to the network entity.
- The network entity processes the packet header and passes the contents of the packet payload up to the transport entity.

- Nesting of TPDU, Packet and Frame



Transport Layer Service Primitives

- To allow users to access the transport service, the transport layer must provide some **operations** to application programs, that is known as **transport service interface**.
- Each transport service has its own interface
- **Difference between transport service and network service**:
  - Transport Service is **reliable** whereas network service is **un-reliable**
  - Network service is **used** only by the **transport entities**.
  - Few users write their own transport entities, and thus few users or programs ever see the bare network service. In contrast, many programs (and thus programmers) see the transport primitives. Consequently, the transport service must be convenient and easy to use.

Contd…

Sr. No	Primitive	Meaning
1.	LISTEN	When server is ready to accept request of incoming connection, it simply put this primitive into action. Listen primitive simply waiting for incoming connection request.
2.	CONNECT	This primitive is used to connect the server simply by creating or establishing connection with waiting peer.
3.	ACCEPT	It simply accepts incoming connection form peer.
4.	SEND	It is put into action by the client to transmit its request that is followed by putting receive primitive into action to get the reply. Send primitive simply sends or transfer the message to the peer.
5.	RECEIVE	These primitive afterwards block the server. Receive primitive simply waits for incoming message.
6.	DISCONNECT	It is simply used to terminate or end the connection after which no one will be able to send any of the message.

Example of Transport Layer Service Primitive

- Consider an application with a **server** and a number of **remote clients**.
- **Step-1**: The **server** executes a **LISTEN** primitive, by calling a library procedure that makes a system call to block the server until a client turns up.
- **Step-2:** When a **client** wants to talk to the server, it executes a **CONNECT** primitive. The transport entity carries out this primitive by blocking the caller and sending a packet to the server. The packet is encapsulated in the payload contains a transport layer message for the server's transport entity.

Contd…

- **Step-3:** The client's CONNECT call causes a **CONNECTION REQUEST TPDU** to be sent to the server. When it arrives, the transport entity checks to see that the server is blocked on a LISTEN It then unblocks the server and sends a **CONNECTION ACCEPTED TPDU** back to the client. When this TPDU arrives, the client is unblocked and the connection is established.

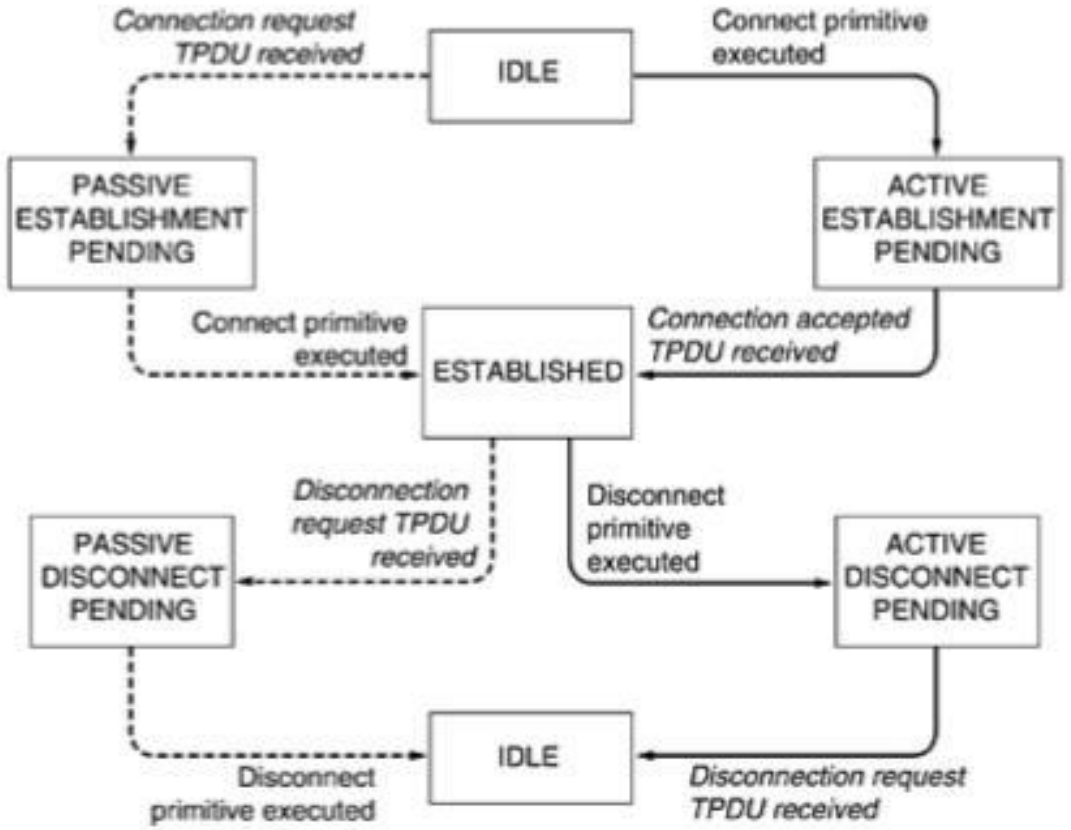
- **Step-4:** Data can now be exchanged using the **SEND** and **RECEIVE** primitives. Either party can do a (blocking) RECEIVE to wait for the other party to do a SEND. When the TPDU arrives, the receiver is unblocked. It can then process the TPDU and send a reply

Contd…

- **Step-5:** When a connection is no longer needed, it must be released to free up table space within the two transport entities. **Disconnection** has two variants: asymmetric and symmetric.

- **In the asymmetric variant**, either transport user can issue a **DISCONNECT** primitive, which results in a DISCONNECT TPDU being sent to the remote transport entity. Upon arrival, the connection is released.
- **In the symmetric variant**, each direction is closed separately, independently of the other one. When one side does a **DISCONNECT**, that means it has no more data to send but it is still willing to accept data from its partner.
- In this model, a connection is released when both sides have done a DISCONNECT

Example of Transport Layer Service Primitive



Berkeley Sockets

- It is another set of transport primitives known as the socket primitives
- These primitives are widely used for Internet programming or Socket Programming
- They follow the same model of but offer more features and flexibility

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

Transmission Control Protocol (TCP)

- TCP (Transmission Control Protocol) was specifically designed to provide a **reliable end-to end byte** stream over an unreliable internetwork.
- TCP service is obtained by both the sender and 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**.
- A port is the TCP name. For TCP service to be obtained, a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine

TCP Port numbers

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial file transfer protocol
79	Finger	Lookup information about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

TCP

- Every byte on a TCP connection has its own **32-bit sequence number**
- The basic protocol used by TCP entities is the **sliding window protocol.**
- The sending and receiving TCP entities exchange data in the form of **segments.**
- A TCP segment consists of a fixed **20-byte header** (plus an optional part) followed by zero or more data bytes.
- The TCP software decides **how big segments** should be.
- It can accumulate data from several writes into one segment or can split data from one write over multiple segments.

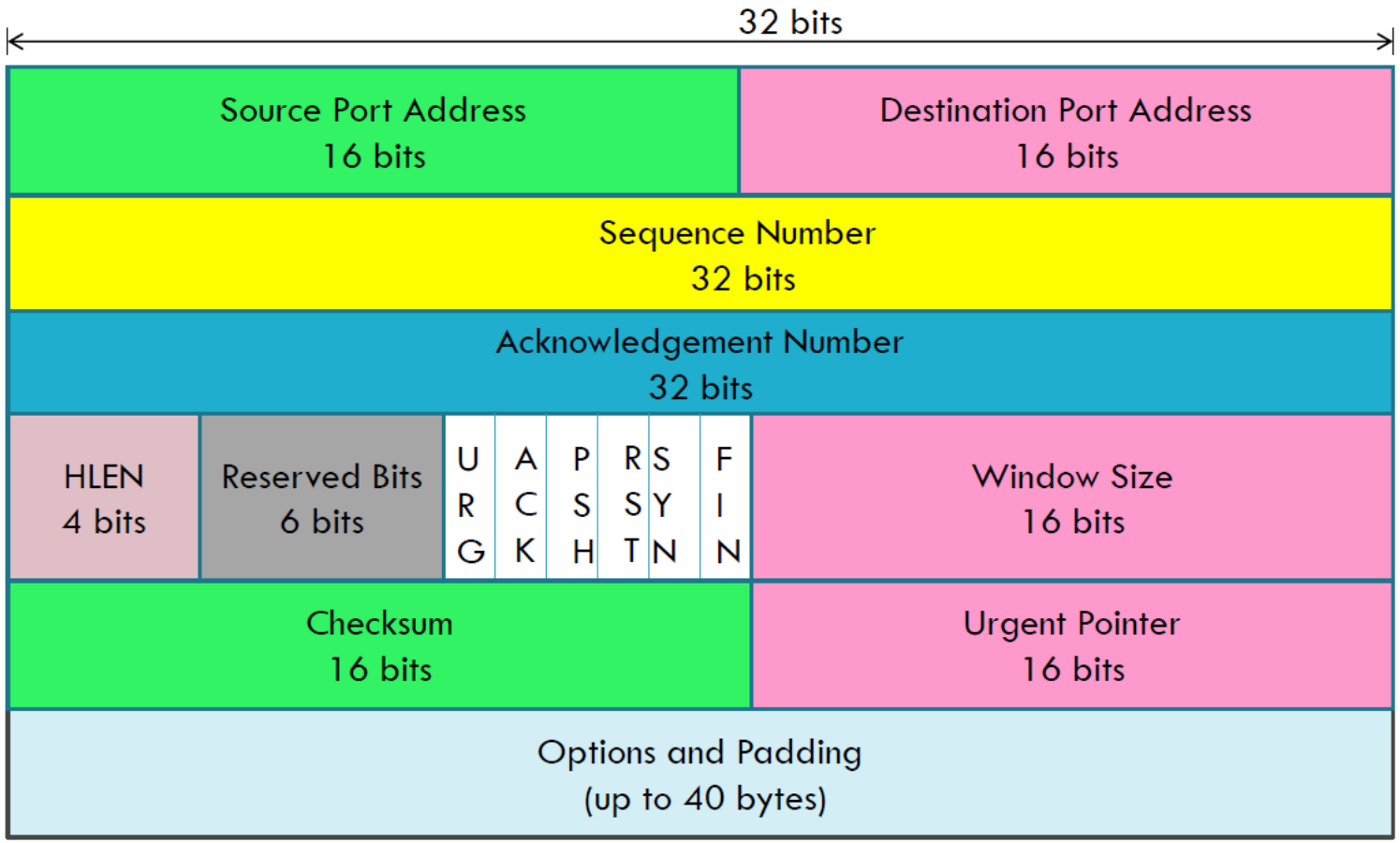
TCP segment

- A TCP segment **encapsulates the data** received from the application layer. The TCP segment itself is encapsulated in an IP datagram, which in turn is encapsulated in a frame at the data link layer.
- **Two limits** restrict the segment size:
  - Each segment, including the **TCP header**, must fit in the **65,515-byte** IP payload.
  - Each network has a maximum transfer unit, or **MTU**, and each segment must fit in the MTU. In practice, the MTU is generally 1500 bytes (the Ethernet payload size) and thus defines the upper bound on segment size.

TCP Header Format

- The segment consists of a header of **20 to 60** bytes, followed by data from the application program.
- The header is **20** bytes if there are **no options** and up to **60** bytes if it contains **options.**
- **TCP segment**

TCP Header


32 bits
Source Port Address
16 bits
Destination Port Address
16 bits
Sequence Number
32 bits
Acknowledgement Number
32 bits
HLEN
4 bits
Reserved Bits
6 bits
URG
ACK
PSH
RST
SYN
FIN
Window Size
16 bits
Checksum
16 bits
Urgent Pointer
16 bits
Options and Padding
(up to 40 bytes)

TCP Header Format

- **Source port address-** This is a **16-bit** field that defines the port number of the application program in the host that is sending the segment.
- **Destination port address-** This is a **16-bit** field that defines the port number of the application program in the host that is receiving the segment.
- **Sequence number-** This **32-bit** field defines the number assigned to the first byte of data contained in this segment.
  - TCP is a **stream** transport protocol.
  - To ensure connectivity, **each byte** to be transmitted is **numbered**.
  - The sequence number tells the destination which byte in this sequence is the **first byte** in the segment.
  - During connection establishment each party uses a **random number generator** to create an **initial sequence number (ISN)**, which is usually different in each direction

Contd…

- **Acknowledgment number-** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party.

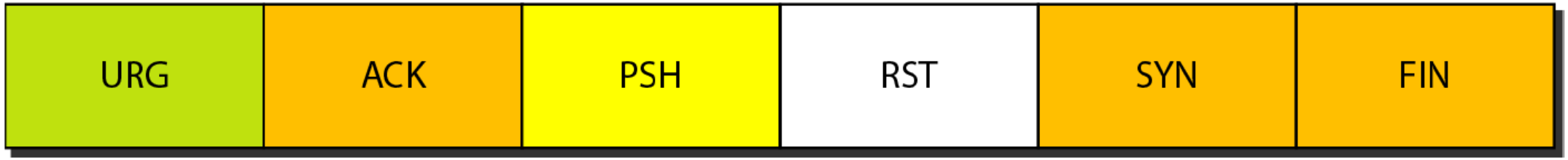
- **Header length-** This 4-bit field indicates the number of 4-byte words in the TCP header.
  - The length of the header can be between 20 and 60 bytes. Therefore, the value of this field is always between 5 ($5 \times 4 = 20$) and 15 ($15 \times 4 = 60$).
- **Reserved Bits:-** This 6-bit field is reserved for future use. The value is set to 0

Contd…

- **Control-** This field defines **6** different control **bits or flags**
  - One or more of these bits can be set at a time.
  - These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP

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



Contd…

- **Window size-** This field defines the window size of the sending TCP in bytes.
  - Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes.
  - This value is normally referred to as the **receiving window (rwnd)** and is determined by the receiver. The sender must obey the dictation of the receiver in this case

- **Checksum.** This 16-bit field contains the checksum.
  - Use of the checksum for TCP is mandatory.

Contd…

- **Urgent pointer**- This **16-bit** field, is valid only if the **urgent flag is set**
  - It is used when the segment contains urgent data.
  - It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.

- **Options-** There can be up to **40 bytes** of optional information in the TCP header.

Example

- **Ex. 1 :** The following is a dump of a TCP header in hexadecimal format :
- 05320017 00000001 00000000 500207FF 00000000
- (i) What is the source port number?
- (ii)What is the destination port number
- (iii) What is the length of the header?
- (iv) What is the type of segment?
- (v) What is the window size?

Soln. :

The dump of a TCP header in hexadecimal format : 05320017 00000001 00000000 500207FF 00000000

- 1) Source port number:- (2 byte) ->) 0532(in hex) or 1330 (in decimal)
- 2) Destination port number:- (2 byte) -> 0017(in hex) or 23(in decimal))
- 3) Length of the header (4 bits) – 5 (in hex and decimal) Means 5 * 4 = 20 bytes header
- 4) Type of the segment - 0X02:-is control field and this indicates a SYN packet.
- 5) Window size - 07FF ((in hex) or 2047 (in decimal)