MLR Institute of Technology

Unit-IV
The Transport Layer

Faculty: V.HEMA
Dept: CSE
EmpID:MLRIT1662

MLR Institute of Technology

Transport Layer

Topics:

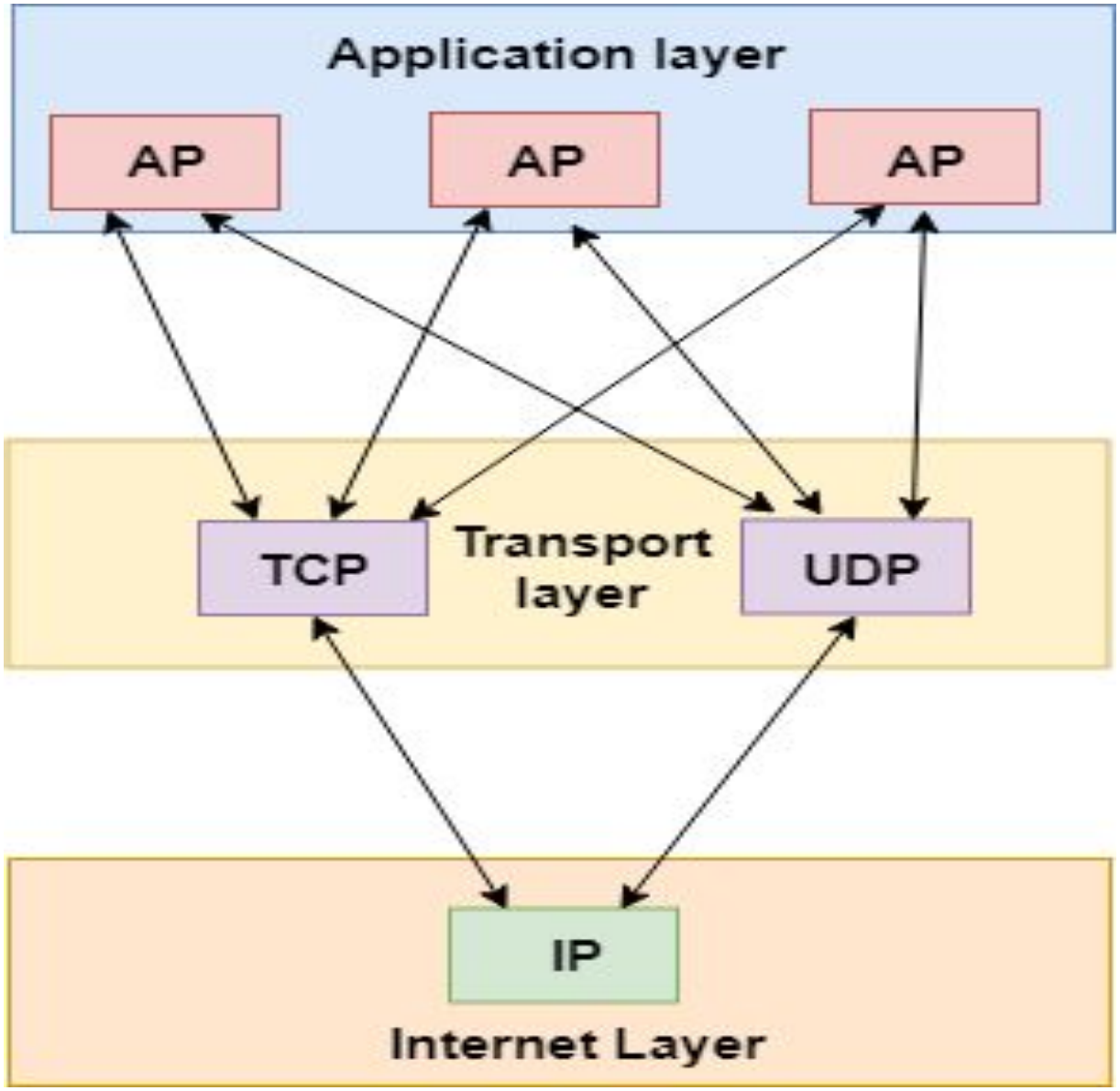
- Introduction.
- Transport services.
- Elements of transport protocol.
- Simple Transport Protocol.
- Internet transport layer protocols: UDP and TCP.

MLR Institute of Technology

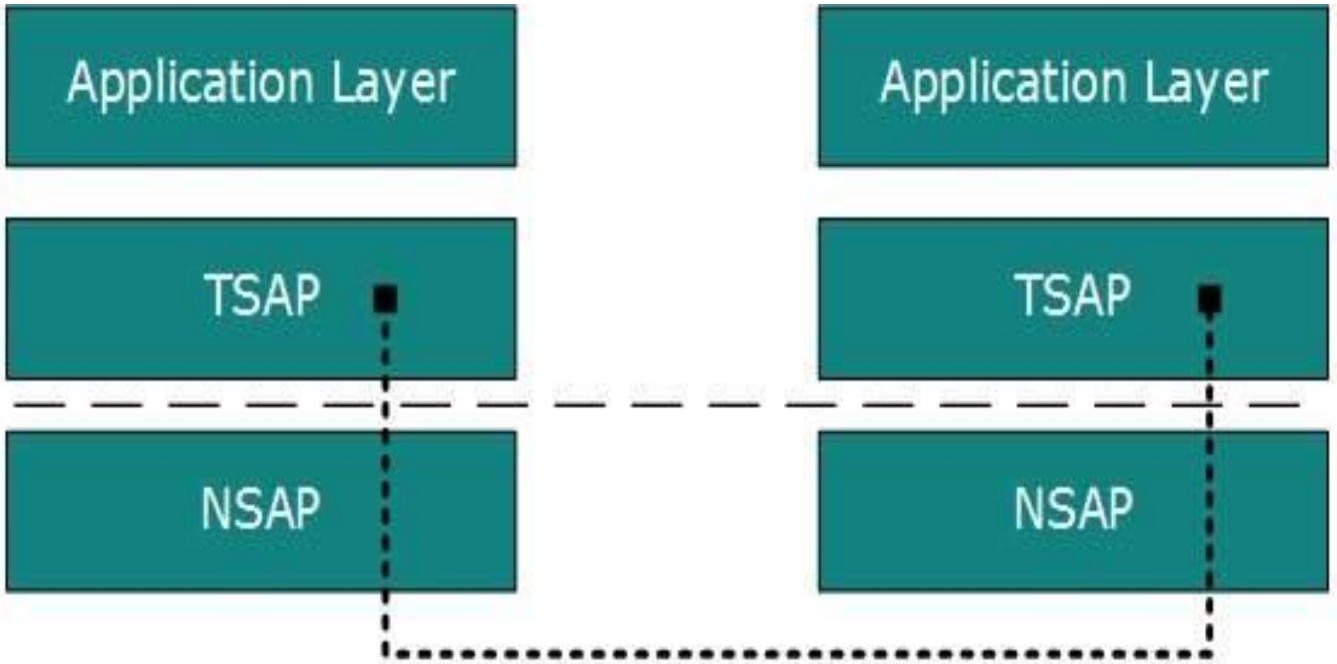
Introduction

- The transport layer is a 4th layer from the top.
- The main role of the transport layer is to provide the communication services directly to the application processes running on different hosts.
- Transport layer takes data from upper layer (i.e. Application layer) and then breaks it into smaller size segments, numbers each byte, and hands over to lower layer (Network Layer) for delivery.

Position of transport layer



Transport layer offers peer-to-peer and end-to-end connection between two processes on remote hosts.

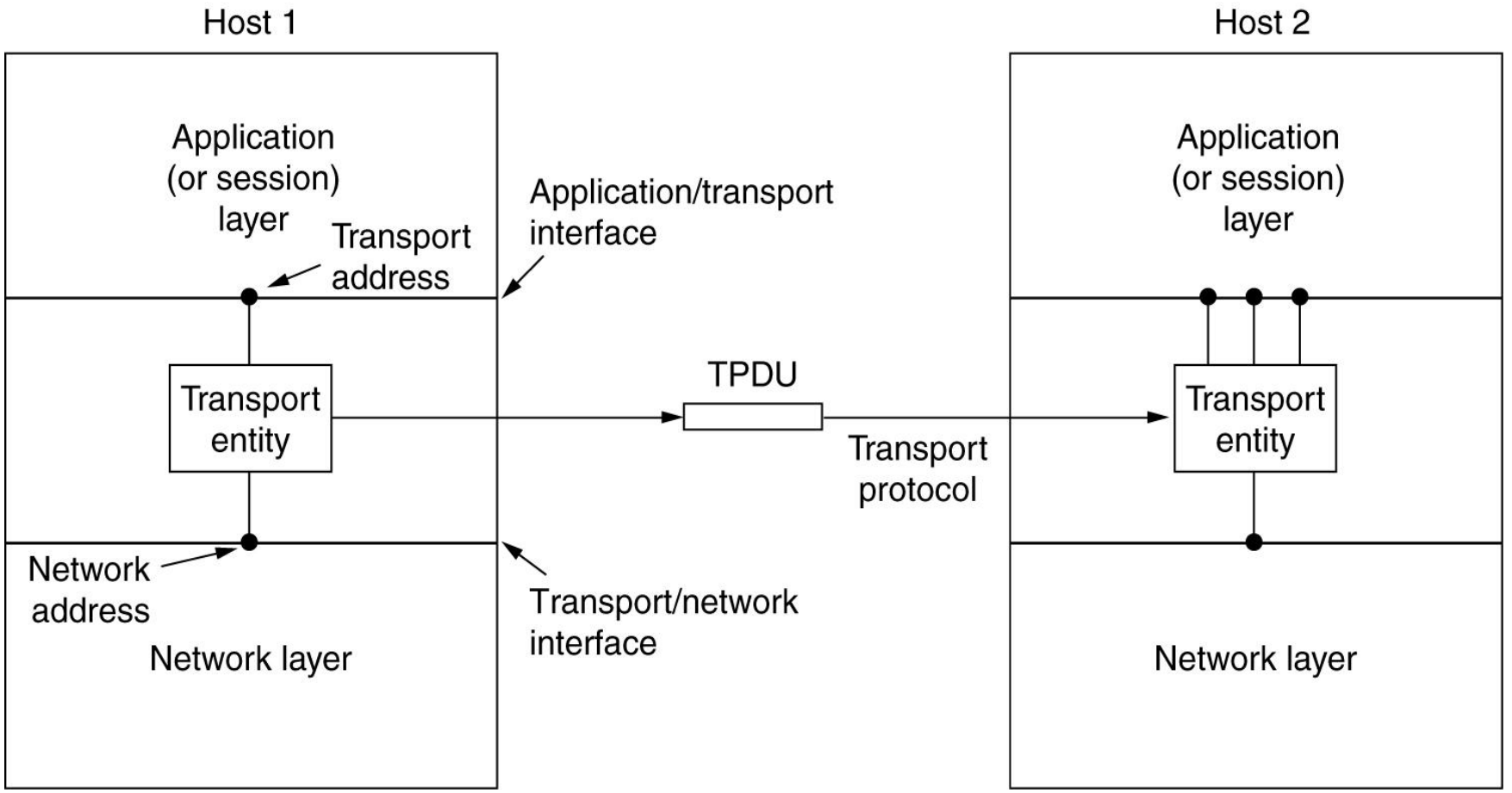


Functions:

- This Layer is the first one which breaks the information data, supplied by Application layer in to smaller units called segments.
- It numbers every byte in the segment and maintains their accounting.
- This layer ensures that data must be received in the same sequence in which it was sent.
- This layer provides end-to-end delivery of data between hosts which may or may not belong to the same subnet.
- All server processes intend to communicate over the network are equipped with well-known Transport Service Access Points (TSAPs) also known as port numbers.

MLR Institute of Technology

Transport services



MLR Institute of Technology

Transport services

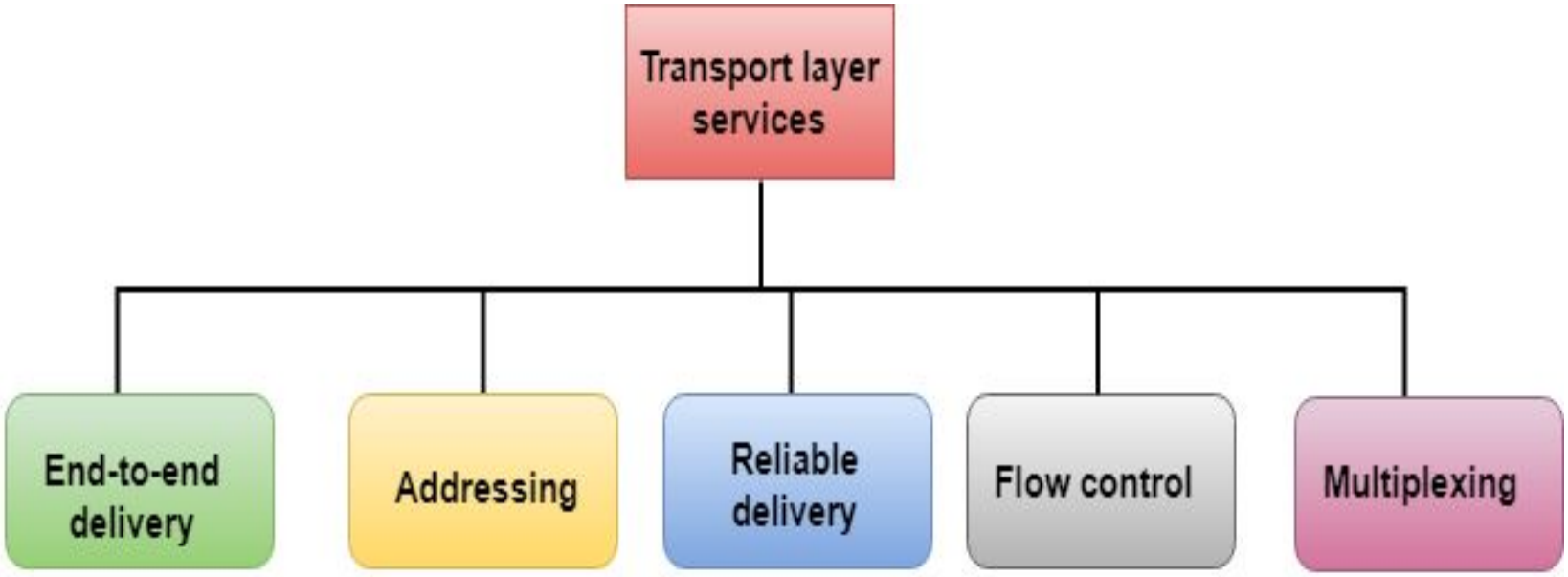
Services provided by the Transport Layer:

- The services provided by the transport layer are similar to those of the data link layer.
- The data link layer provides the services within a single network while the transport layer provides the services across an internetwork made up of many networks.
- The data link layer controls the physical layer while the transport layer controls all the lower layers.

Transport services

The services provided by the transport layer protocols can be divided into five categories:

1. End-to-end delivery
2. Addressing
3. Reliable delivery
4. Flow control
5. Multiplexing



MLR Institute of Technology

Transport services

End-to-end delivery:

The transport layer transmits the entire message to the destination.

Therefore, it ensures the end-to-end delivery of an entire message from a source to the destination.
Application-application

MLR Institute of Technology

Transport services

Reliable delivery:

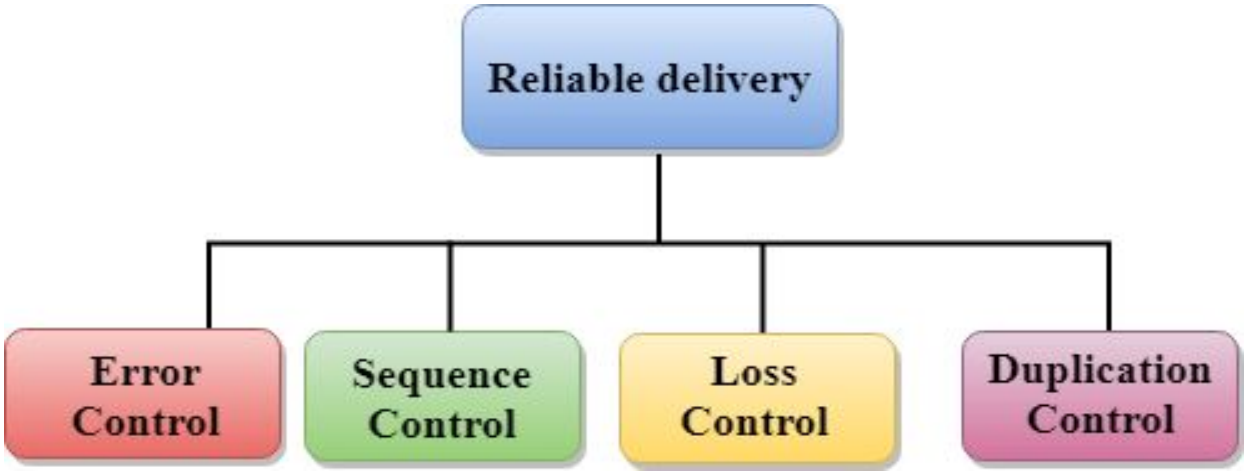
The transport layer provides reliability services by retransmitting the lost and damaged packets.

The reliable delivery has four aspects:

- Error control
- Sequence control
- Loss control
- Duplication control

Transport services

Reliable delivery:



MLR Institute of Technology

Transport services

Reliable delivery:

Error Control:

The primary role of reliability is **Error Control**. In reality, no transmission will be 100 percent error-free delivery. Therefore, transport layer protocols are designed to provide error-free transmission.

The data link layer also provides the error handling mechanism, but it ensures only node-to-node error-free delivery. However, node-to-node reliability does not ensure the end-to-end reliability.

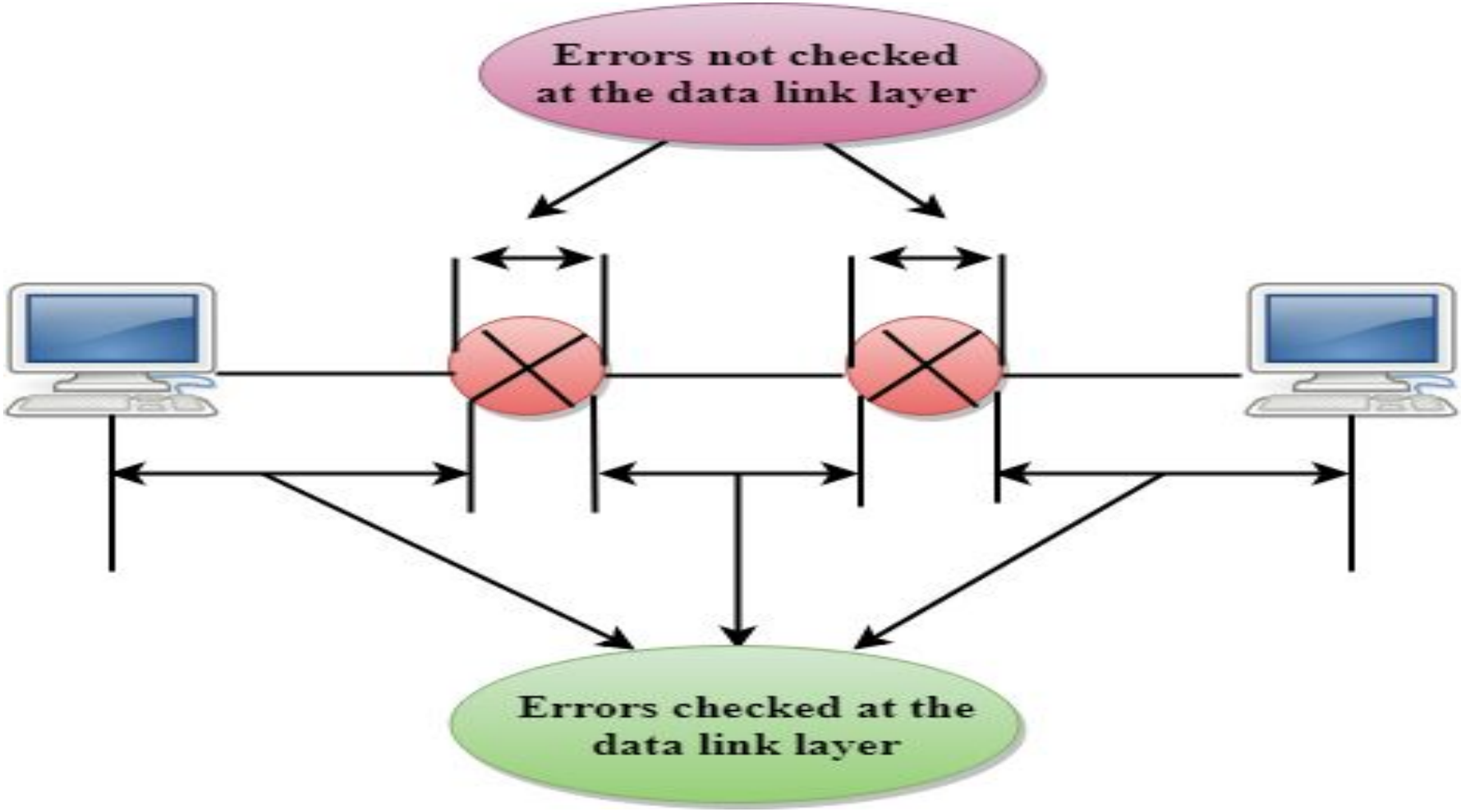
The data link layer checks for the error between each network. If an error is introduced inside one of the routers, then this error will not be caught by the data link layer. It only detects those errors that have been introduced between the beginning and end of the link.

Therefore, the transport layer performs the checking for the errors end-to-end to ensure that the packet has arrived correctly.

MLR Institute of Technology

Transport services

Reliable delivery:



Transport services

Reliable delivery:

Sequence Control:

- The second aspect of the reliability is sequence control which is implemented at the transport layer.

- On the sending end, the transport layer is responsible for ensuring that the packets received from the upper layers can be used by the lower layers.

- On the receiving end, it ensures that the various segments of a transmission can be correctly reassembled.

Hang him not, leave him
Hang him, not leave him

MLR Institute of Technology

Transport services

Reliable delivery:

Loss Control:

- Loss Control is a third aspect of reliability.
- The transport layer ensures that all the fragments of a transmission arrive at the destination, not some of them.
- On the sending end, all the fragments of transmission are given sequence numbers by a transport layer.
- These sequence numbers allow the receivers transport layer to identify the missing segment.

MLR Institute of Technology

Transport services

Reliable delivery:

Duplication Control:

- Duplication Control is the fourth aspect of reliability.
- The transport layer guarantees that no duplicate data arrive at the destination.
- Sequence numbers are used to identify the lost packets; similarly, it allows the receiver to identify and discard duplicate segments.

Transport services

Flow Control:

- Flow control is used to prevent the sender from overwhelming the receiver.
- If the receiver is overloaded with too much data, then the receiver discards the packets and asking for the retransmission of packets.
- This increases network congestion and thus, reducing the system performance.
- The transport layer is responsible for flow control.
- It uses the sliding window protocol that makes the data transmission more efficient as well as it controls the flow of data so that the receiver does not become overwhelmed.
- Sliding window protocol is byte oriented rather than frame oriented.

Transport services

Multiplexing:

The transport layer uses the multiplexing to improve transmission efficiency.

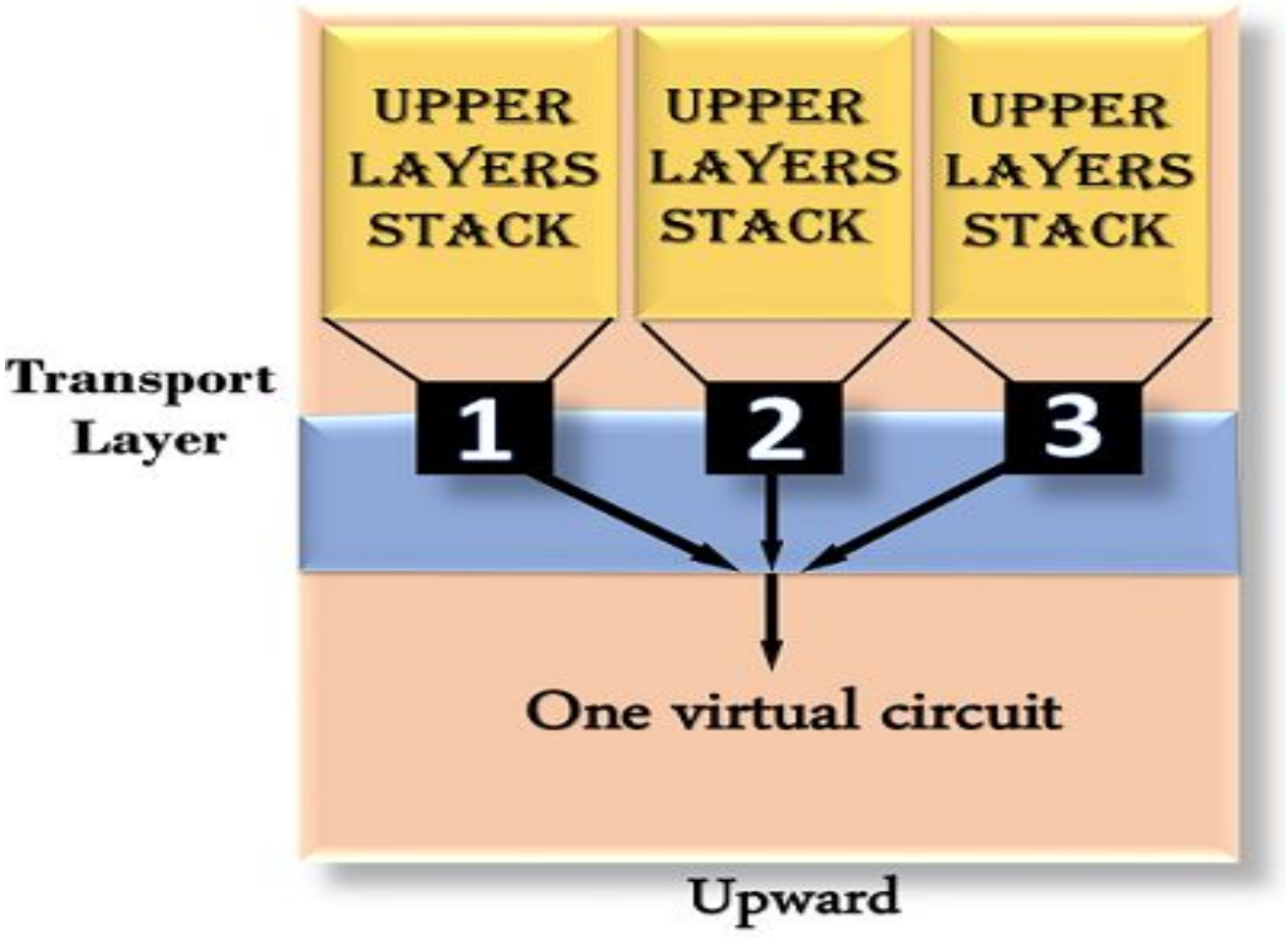
Multiplexing can occur in two ways:

Upward multiplexing: Upward multiplexing means multiple transport layer connections use the same network connection. To make more cost-effective, the transport layer sends several transmissions bound for the same destination along the same path; this is achieved through upward multiplexing.

MLR Institute of Technology

Transport services

Multiplexing:



Transport services

Multiplexing:

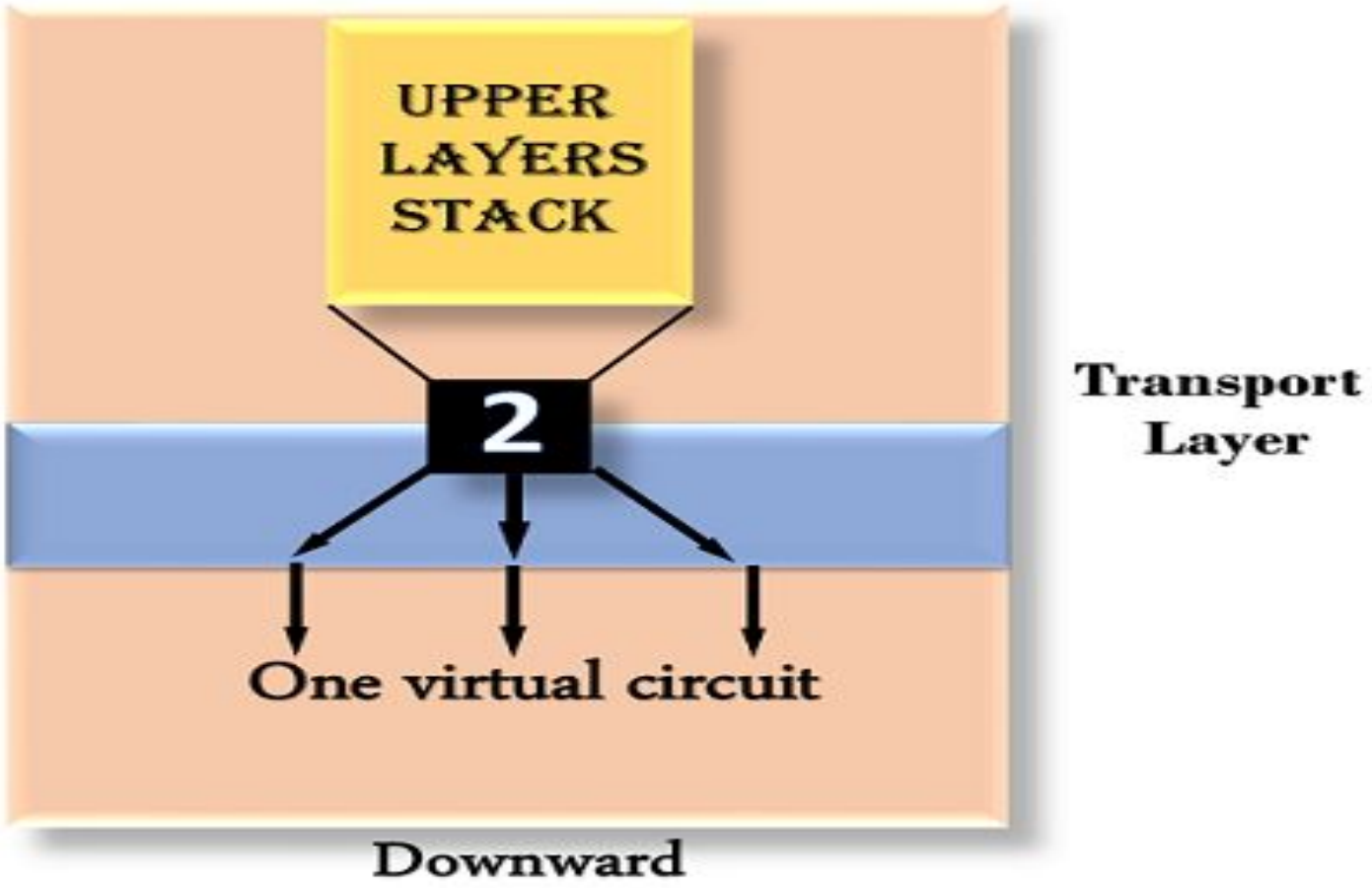
Downward multiplexing: Downward multiplexing means one transport layer connection uses the multiple network connections.

- Downward multiplexing allows the transport layer to split a connection among several paths to improve the throughput.
- This type of multiplexing is used when networks have a low or slow capacity.

MLR Institute of Technology

Transport services

Multiplexing:



MLR Institute of Technology

Transport services

Addressing:

- According to the layered model, the transport layer interacts with the functions of the session layer. Many protocols combine session, presentation, and application layer protocols into a single layer known as the application layer.

- In these cases, delivery to the session layer means the delivery to the application layer.

- **Data generated by an application on one machine must be transmitted to the correct application on another machine. In this case, addressing is provided by the transport layer.**

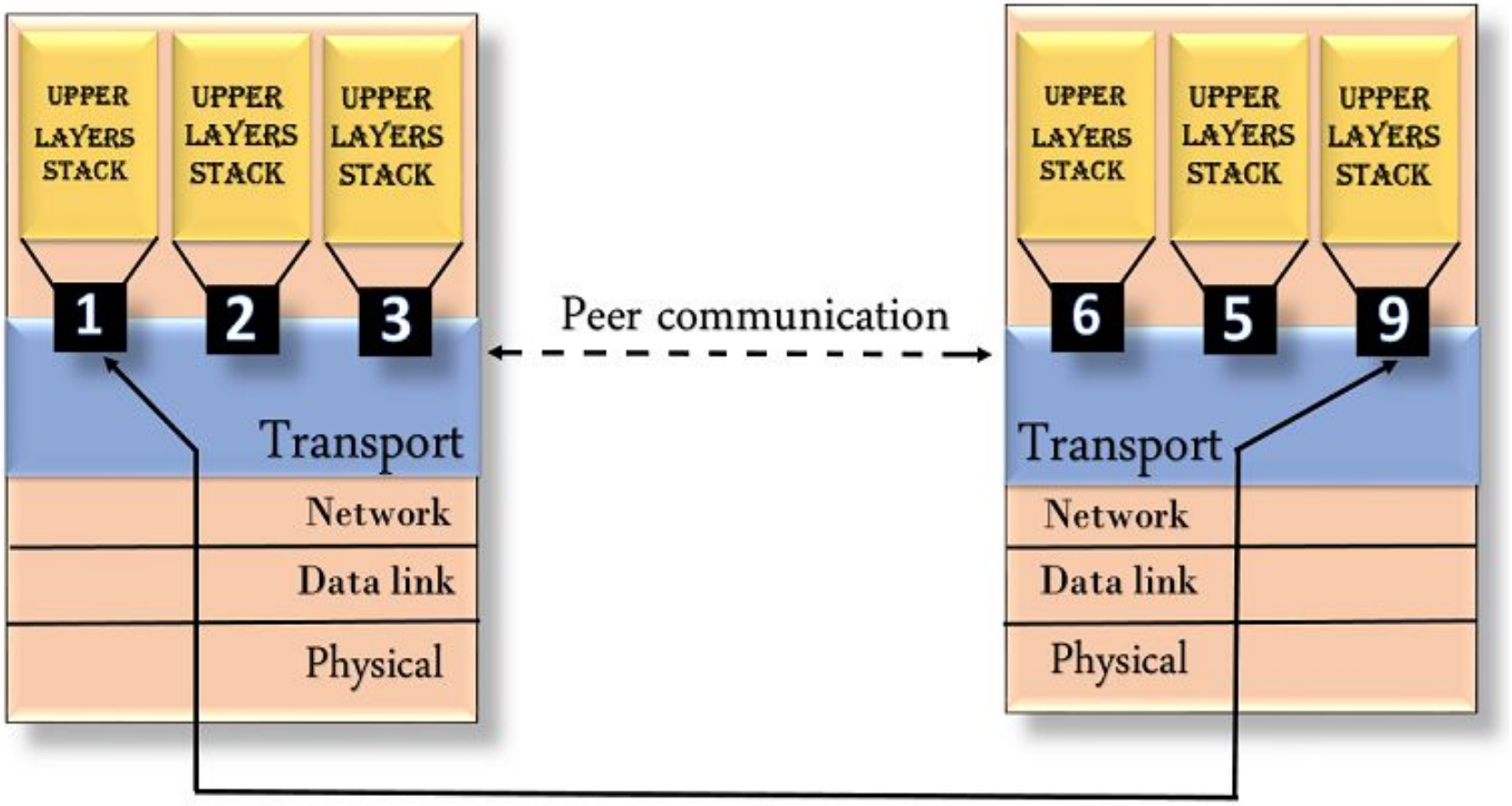
- The transport layer provides the user address which is specified as a station or port. The port variable represents a particular TS user of a specified station known as a Transport Service access point (TSAP). Each station has only one transport entity.

- The transport layer protocols need to know which upper-layer protocols are communicating.

MLR Institute of Technology

Transport services

Addressing:



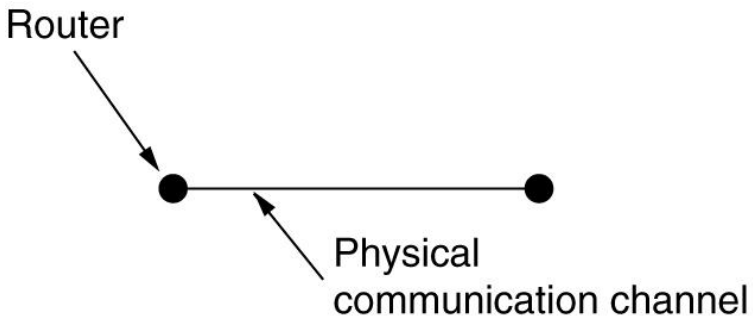
MLR Institute of Technology

Elements of Transport Protocols

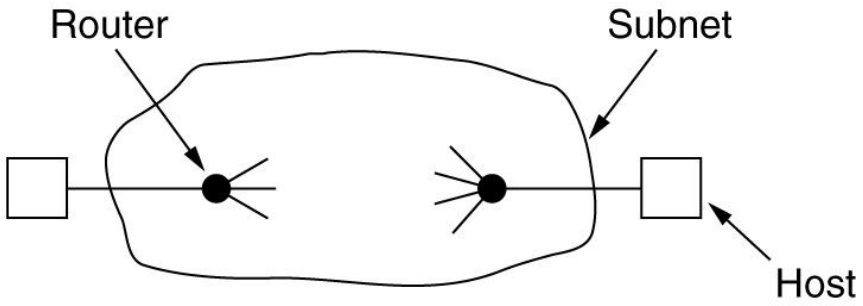
- Addressing
- Connection Establishment
- Connection Release
- Flow Control and Buffering
- Multiplexing
- Crash Recovery

MLR Institute of Technology

Transport Protocol



(a)

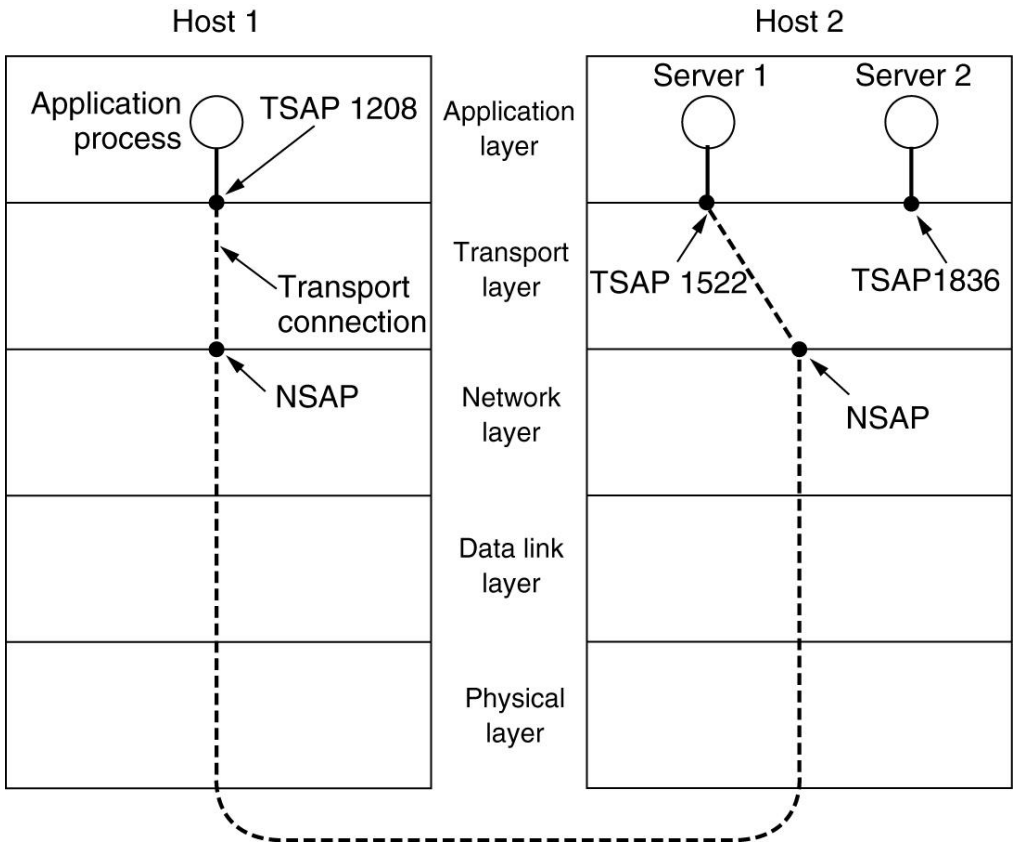


(b)

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

MLR Institute of Technology

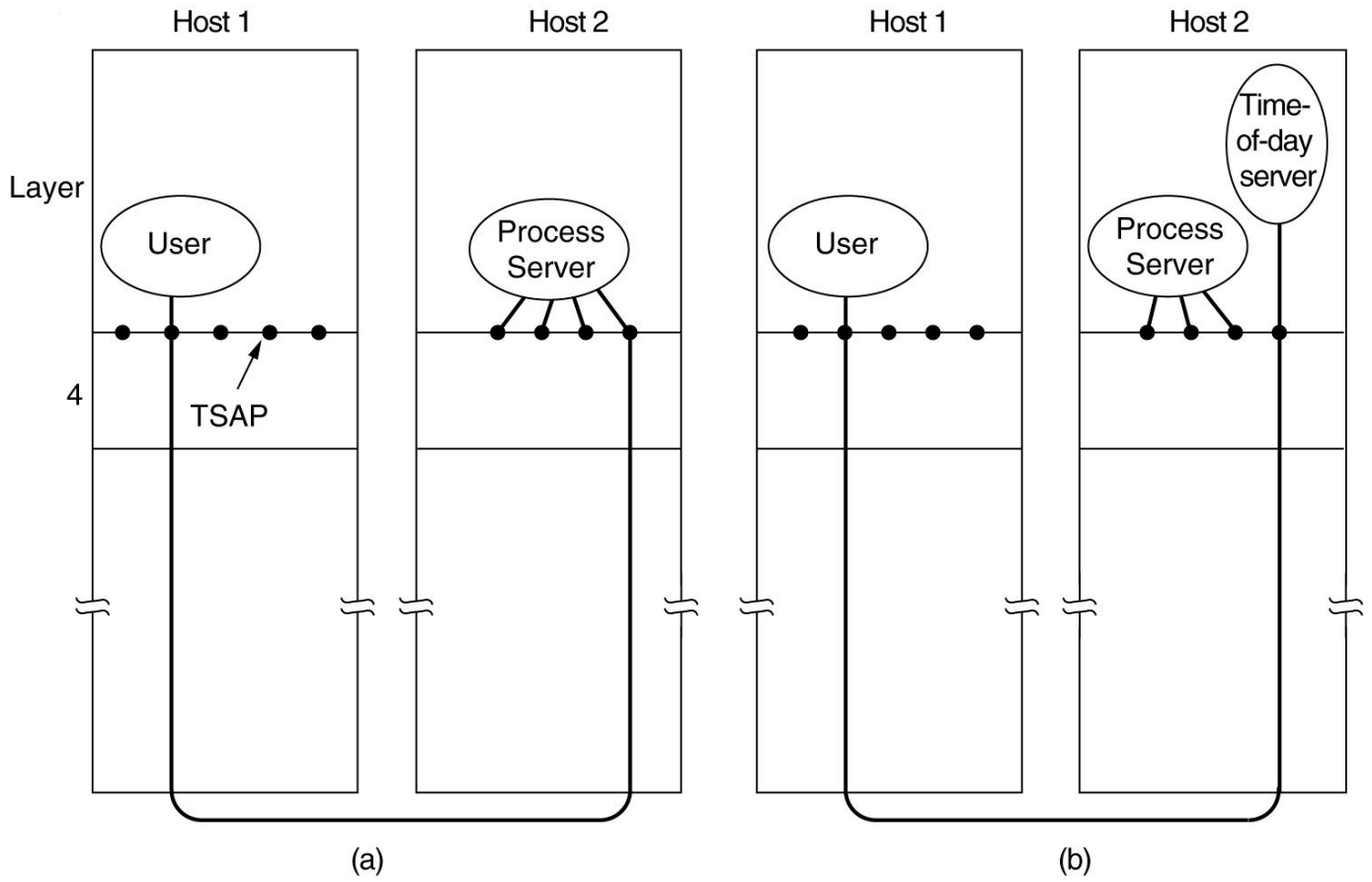
Addressing



TSAPs, NSAPs and transport connections.

MLR Institute of Technology

Connection Establishment



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

Connection Establishment…

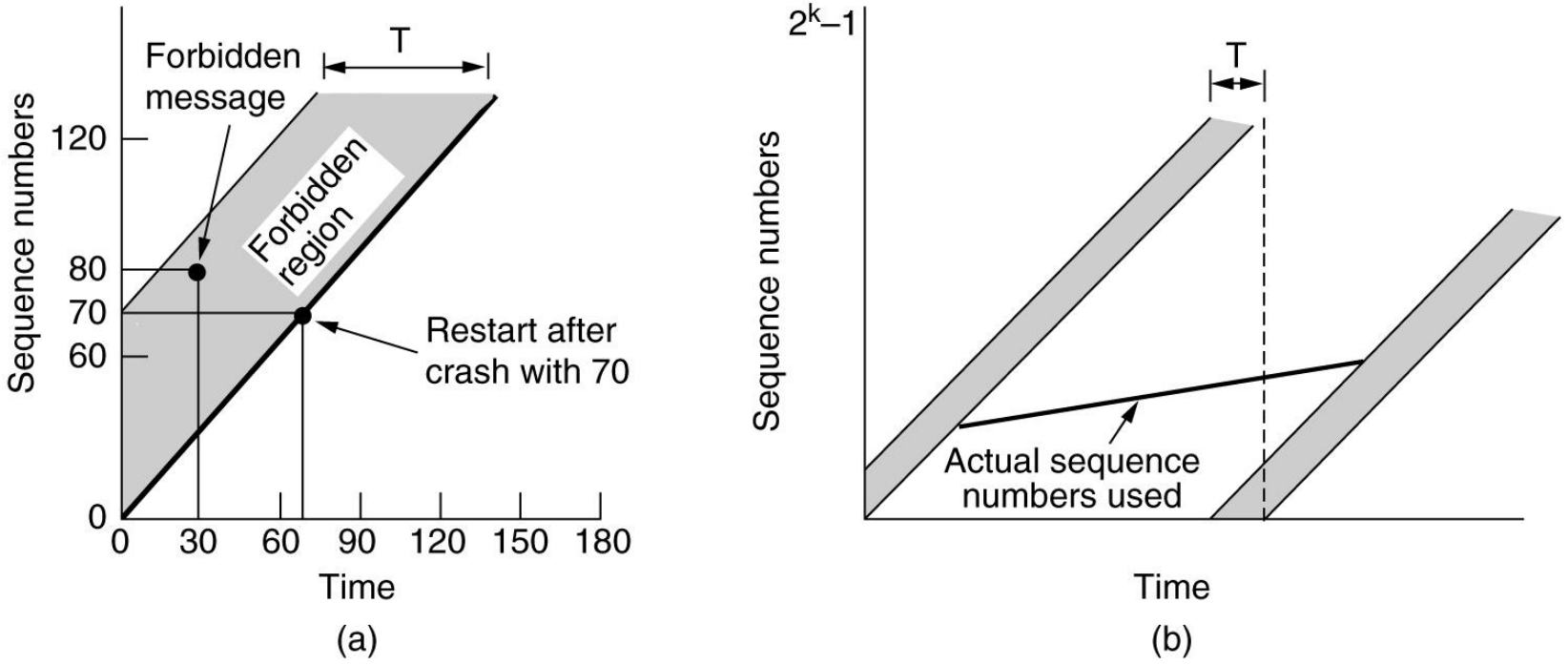
- Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted subnet design.
2. Putting a hop counter in each packet.
3. Timestamping each packet.

- To get around the problem of a machine losing all memory of where it was after a crash, **Tomlinson proposed equipping each host with a time-of-day clock**.

1. The clocks at different hosts need not be synchronized. Each clock is assumed to take the form of a binary counter that increments itself at uniform intervals.
2. Furthermore, the number of bits in the counter must equal or exceed the number of bits in the sequence numbers.
3. Last, and most important, the clock is assumed to continue running even if the host goes down.

Connection Establishment (2)

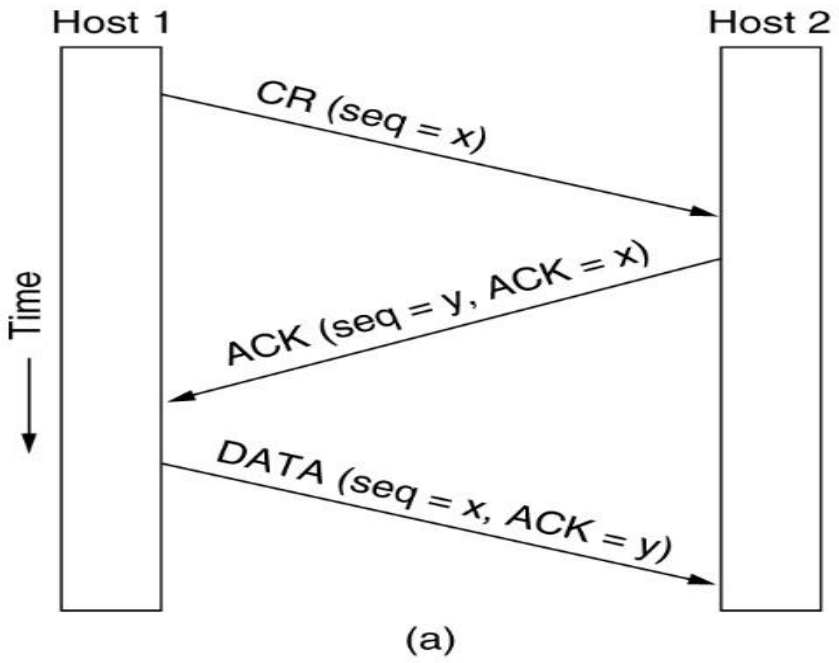


(a) TPDUs may not enter the forbidden region.

(b) The resynchronization problem.

MLR Institute of Technology

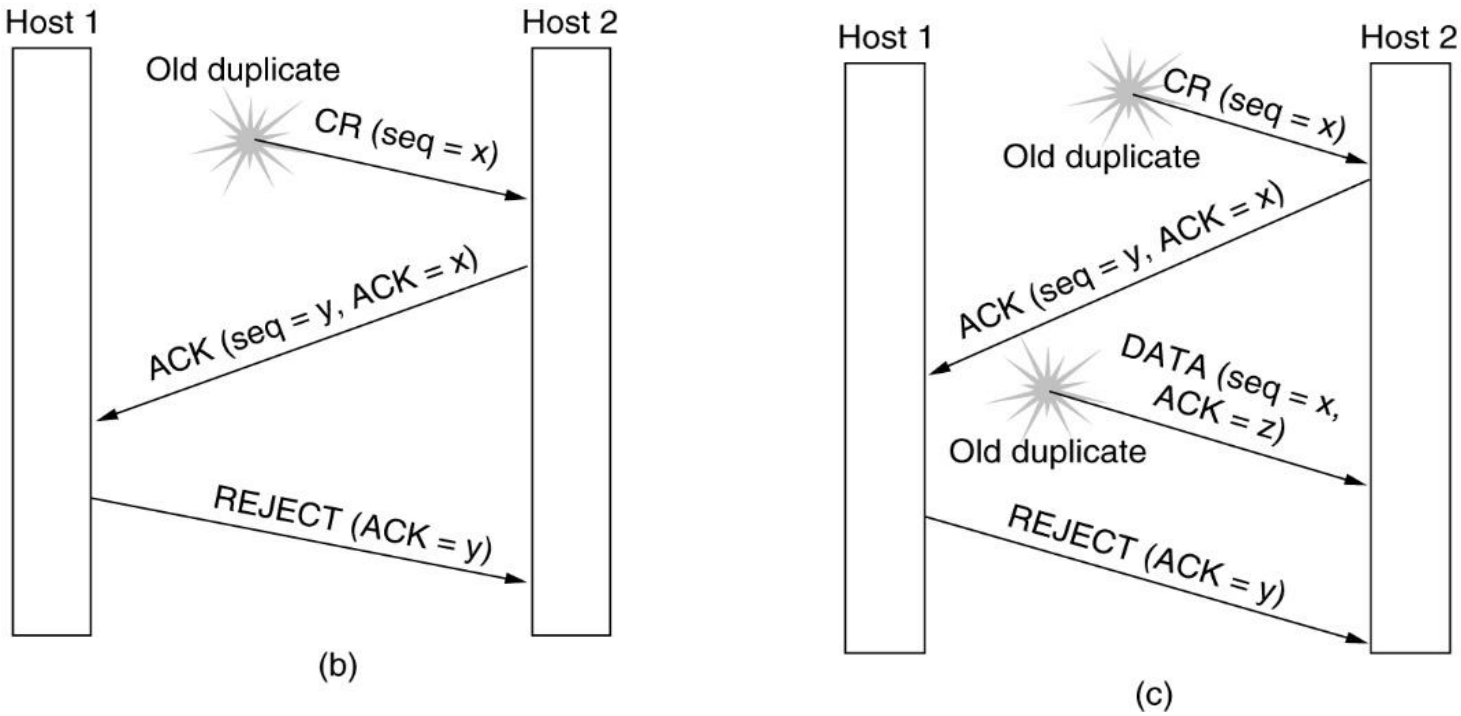
Connection Establishment (3)



Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST.
(a) Normal operation,

MLR Institute of Technology

Connection Establishment (3)



(b) Old CONNECTION REQUEST appearing out of nowhere.
(c) Duplicate CONNECTION REQUEST and duplicate ACK.

MLR Institute of Technology

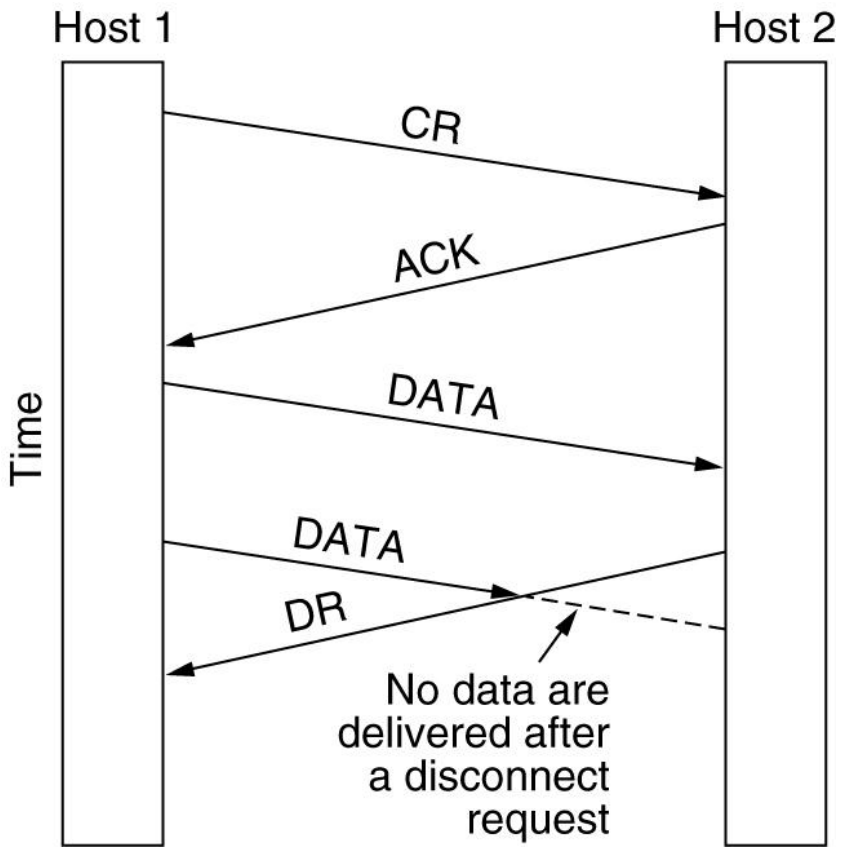
Connection Release

Disconnection connection between two users.

- Asymmetric Release
- Symmetric Release

MLR Institute of Technology

Connection Release



Abrupt disconnection with loss of data.

MLR Institute of Technology

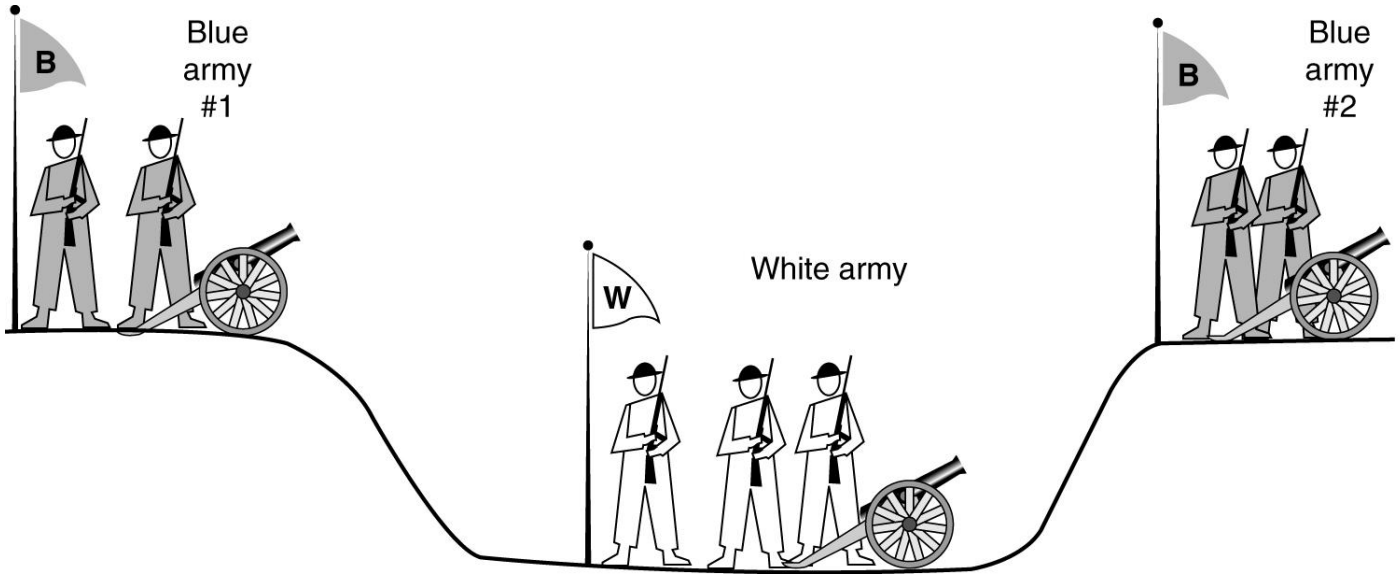
Connection Release

- One way to avoid data loss is to use symmetric release.
  In which each direction is released independently of the other one.

- One can envision a protocol in which:host 1 says I'm and you too?
  If host 2 says

MLR Institute of Technology

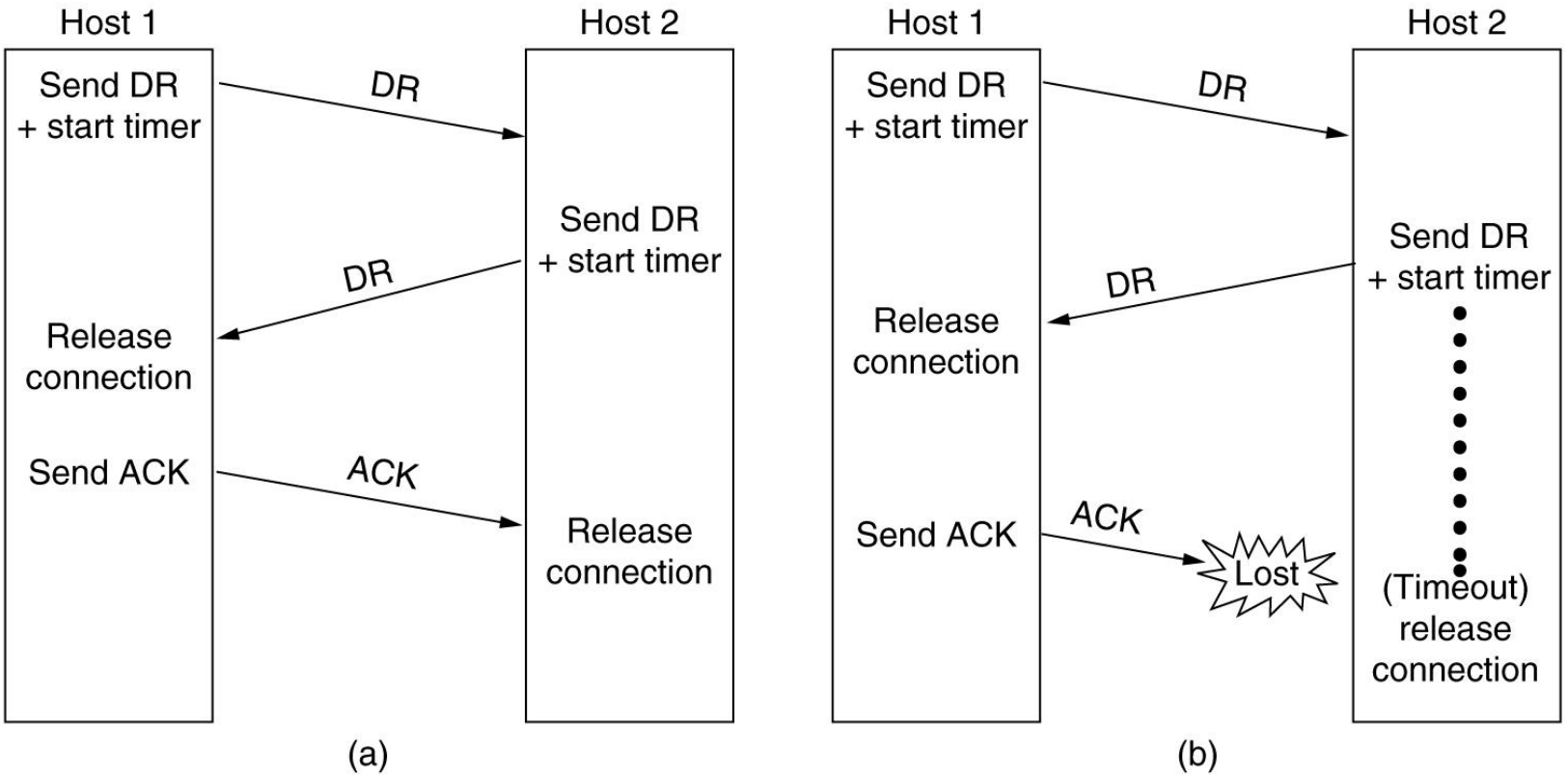
Connection Release (2)



The two-army problem.

MLR Institute of Technology

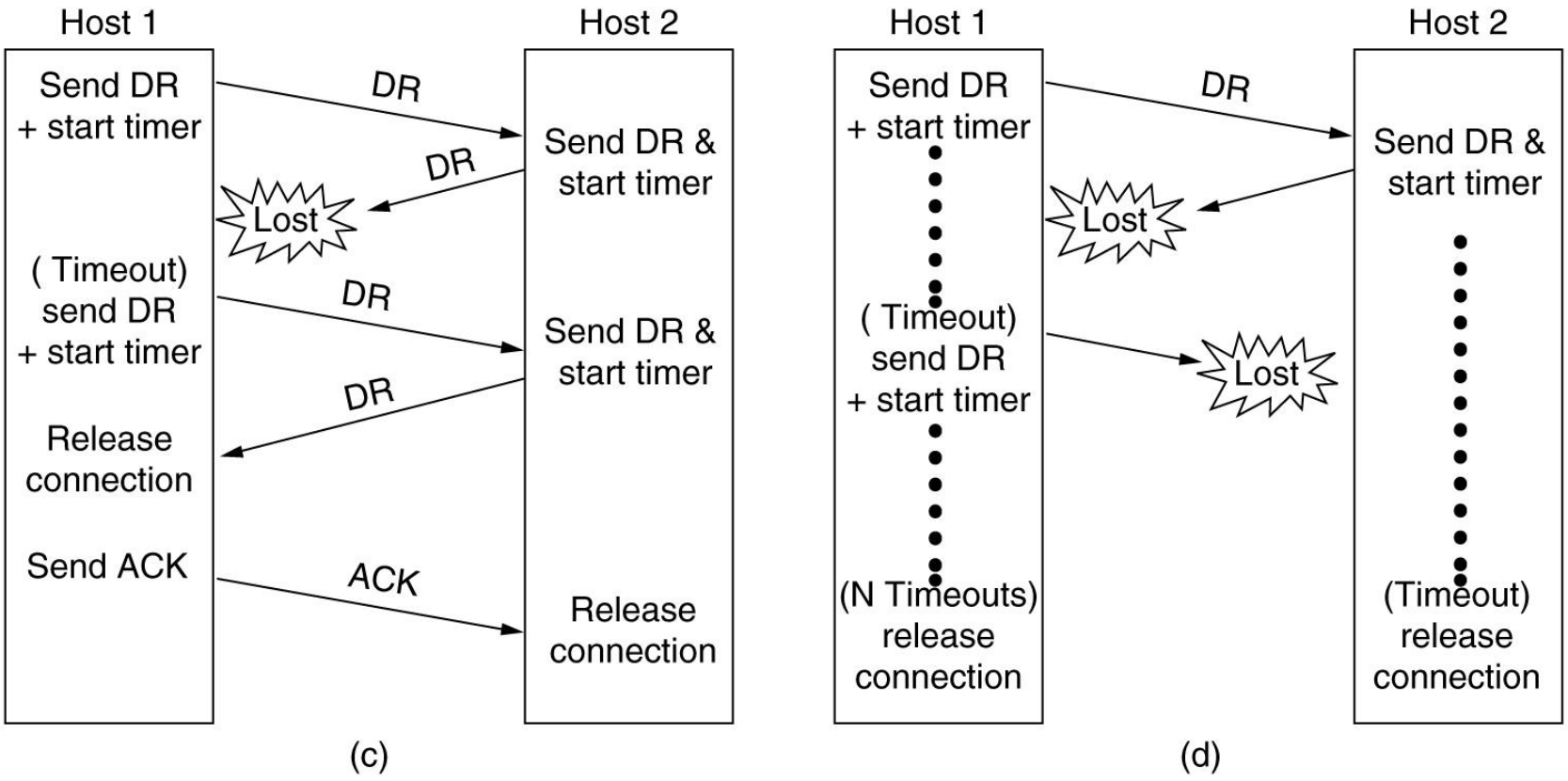
Connection Release (3)



Four protocol scenarios for releasing a connection. (a) Normal case of a three-way handshake. (b) final ACK lost.

MLR Institute of Technology

Connection Release (4)



(c) Response lost. (d) Response lost and subsequent DRs lost.

MLR Institute of Technology

Flow Control and Buffering

- In some ways the flow control problem in the transport layer is the same as in the data link layer, but in other ways it is different.
- The basic similarity is that in 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.
- This difference makes it impractical to implement the data link buffering strategy in the transport layer.

- if the network service is unreliable, the sender must buffer all TPDUs sent, just as in the data link layer.

Flow Control and Buffering

- Even if the receiver has agreed to do the buffering, there still remains the question of the buffer size.
- If most TPDUs are nearly the same size, it is natural to organize the buffers as a pool of identically-sized buffers, with one TPDU per buffer,
- If there is wide variation in TPDU size, from a few characters typed at a terminal to thousands of characters from file transfers, a pool of fixed-sized buffers presents problems.
- 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.

MLR Institute of Technology

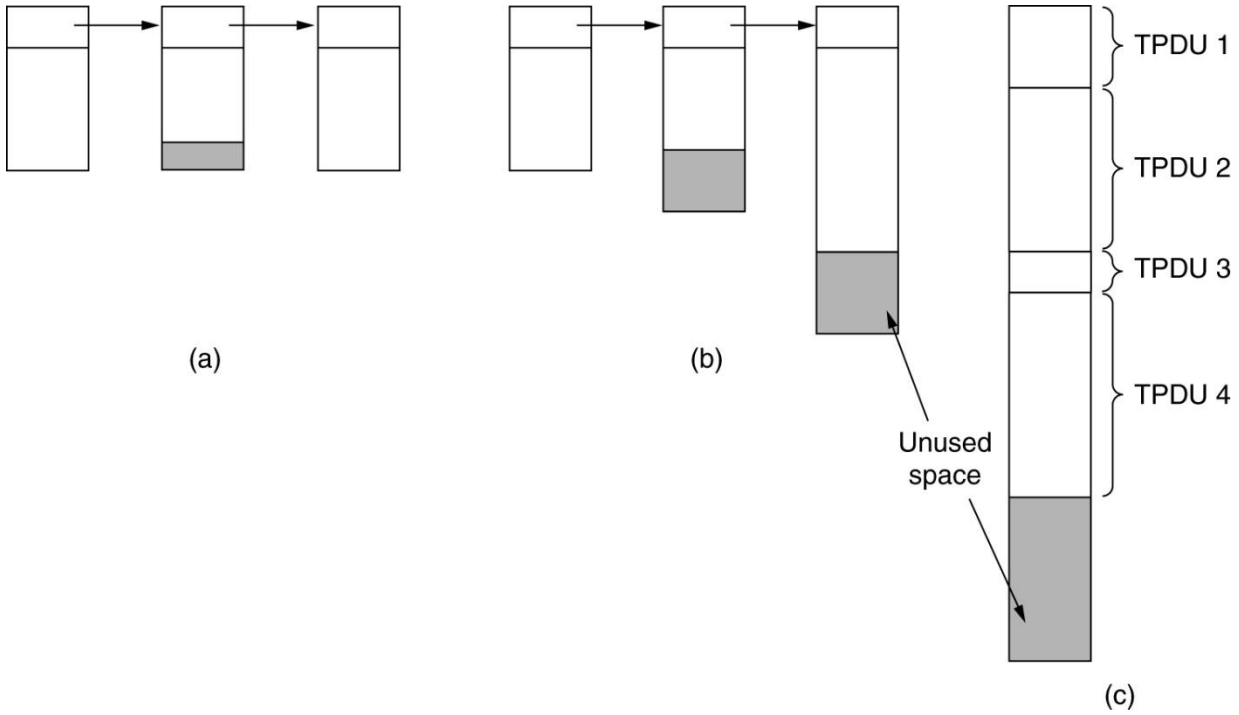
Flow Control and Buffering

- Another approach to the buffer size problem is to use variable-sized buffers.
- The advantage here is better memory utilization, at the price of more complicated buffer management.
- 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 bursty traffic, it is better to buffer at the sender, and for highbandwidth smooth traffic, it is better to buffer at the receiver.

MLR Institute of Technology

Flow Control and Buffering



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

MLR Institute of Technology

Dynamic Buffer Management.

- Dynamic buffer management means, in effect, a variable-sized window.
- Initially, the sender requests a certain number of buffers, based on its perceived needs.
- The receiver then grants as many of these as it can afford.
- Every time the sender transmits a TPDU, it must decrement its allocation, stopping altogether when the allocation reaches zero.
- The receiver then separately piggybacks both acknowledgements and buffer allocations onto the reverse traffic.

MLR Institute of Technology

Flow Control and Buffering (2)

	A	Message	B	Comments
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	• • •	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has 1 buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	• • •	<ack = 6, buf = 4>	←	Potential deadlock

Dynamic buffer allocation. The arrows show the direction of transmission. An ellipsis (…) indicates a lost TPDU.

Dynamic buffer allocation.

- an example of how dynamic window management might work in a datagram subnet with 4-bit sequence numbers.
- Assume that buffer allocation information travels in separate TPDUs, as shown, and is not piggybacked onto reverse traffic.
- Initially, *A* wants eight buffers,
- but is granted only four of these.
- It then sends three TPDUs, of which the third is lost.
- TPDU 6 acknowledges receipt of all TPDUs up to and including sequence number 1,
- thus allowing *A* to release those buffers,
- and furthermore informs *A* that it has permission to send three more TPDUs starting beyond 1 (i.e., TPDUs 2, 3, and 4). *A* knows that it has already sent number 2, so it thinks that it may send TPDUs 3 and 4, which it proceeds to do.

Dynamic buffer allocation.

- At this point it is blocked and must wait for more buffer allocation.
- Timeout-induced retransmissions (line 9), however, may occur while blocked, since they use buffers that have already been allocated.
- In line 10, *B* acknowledges receipt of all TPDUs up to and including 4 but refuses to let *A* continue.
- Such a situation is impossible with the fixed window protocols
- The next TPDU from *B* to *A* allocates another buffer and allows *A* to continue.
- Look at line 16. *B* has now allocated more buffers to *A*, but the allocation TPDU was lost. Since control TPDUs are not sequenced or timed out,
- *A* is now deadlocked.
- To prevent this situation, each host should periodically send control TPDUs giving the acknowledgement and buffer status on each connection. That way, the deadlock will be broken, sooner or later.

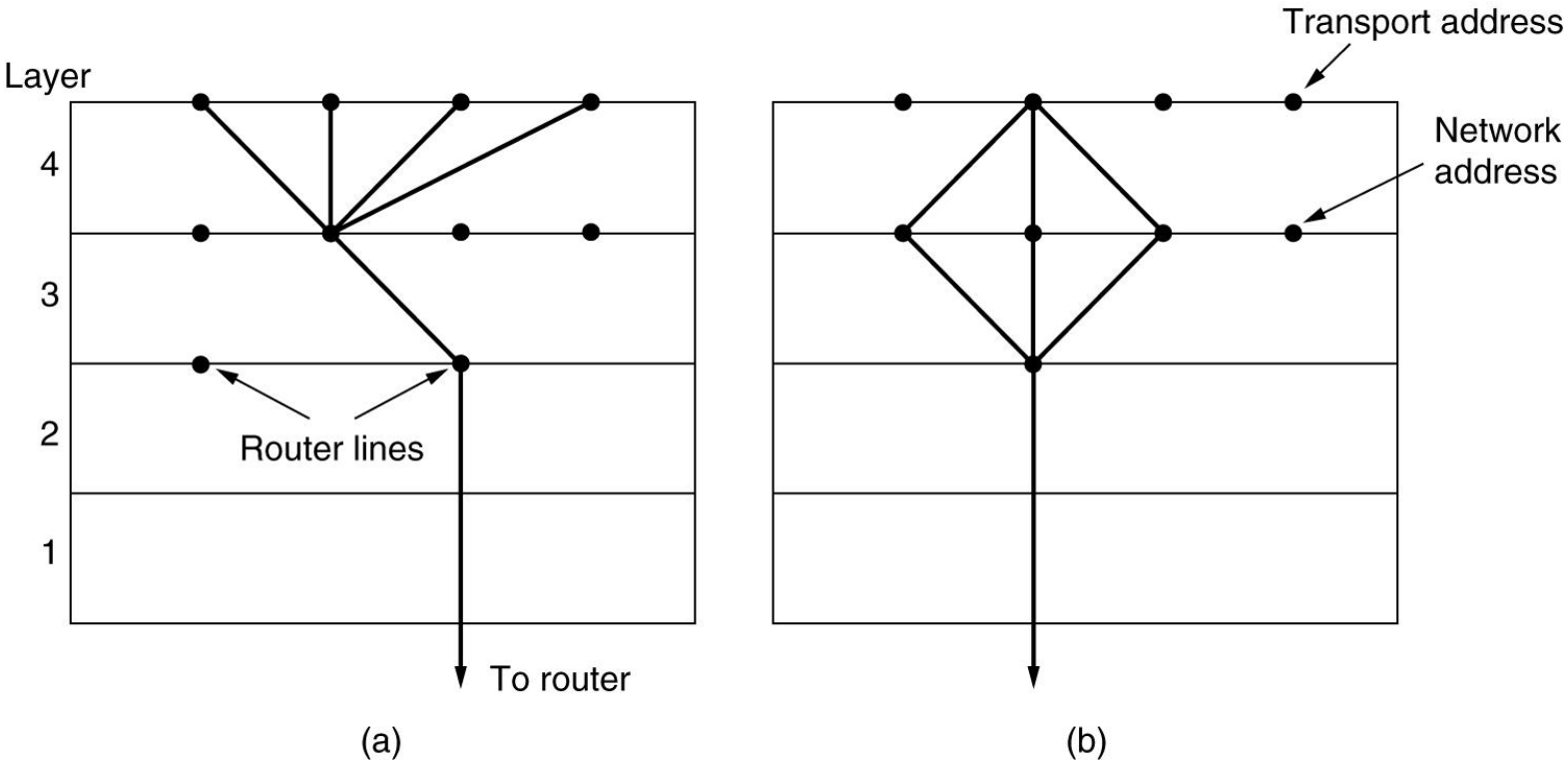
MLR Institute of Technology

Multiplexing

- 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**,
- Multiplexing can also be useful in the transport layer for another reason. Suppose, for example, that a subnet uses virtual circuits internally and imposes a maximum data rate on each one.
- If a user needs more bandwidth than one virtual circuit can provide, a way out is to open multiple network connections and distribute the traffic among them on a round-robin basis, This modus operandi is called **downward multiplexing**.

MLR Institute of Technology

Multiplexing



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

MLR Institute of Technology

Crash Recovery

- No matter how the client and server are programmed, there are always situations where the protocol fails to recover properly.
- The server can be programmed in one of two ways:
  - acknowledge first or
  - write first.
- The client can be programmed in one of four ways:
  - always retransmit the last TPDU,
  - never retransmit the last TPDU,
  - retransmit only in state *S0*, or
  - retransmit only in state *S1*.

MLR Institute of Technology

Crash Recovery

- Always Retransmit.
- First Acknowledgment then write.
  - Retransmit in S0(Sent 2 messages, Ack of both received).
- No outstanding Packet is present.
  - retransmit in S1(Sent 2 messages, Ack of only 1 received). Here Outstanding Packet is present

Crash Recovery

events are possible at the server:

- sending an acknowledgement (*A*),
- writing to the output process (*W*), and
- crashing (*C*).

- The three events can occur in six different orderings: *AC*(*W*), *AWC*, *C*(*AW*), *C*(*WA*), *WAC*, and *WC*(*A*),
- where the parentheses are used to indicate that neither *A* nor *W* can follow *C* (i.e., once it has crashed, it has crashed).
- Figure shows all eight combinations of client and server strategy and the valid event sequences for each one.
- Notice that for each strategy there is some sequence of events that causes the protocol to fail.
- For example, if the client always retransmits, the *AWC* event will generate an undetected duplicate, even though the other two events work properly.

MLR Institute of Technology

Crash Recovery

Strategy used by sending host	Strategy used by receiving host: First ACK, then write			Strategy used by receiving host: First write, then ACK		
	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

Different combinations of client and server strategy.

A Simple Transport Protocol

- The Example Service Primitives
- The Example Transport Entity
- The Example as a Finite State Machine

The Example Service Primitives(1)

- Our concrete service definition consists of five primitives: CONNECT, LISTEN, DISCONECT, SEND, and RECEIVE.

- Each primitive corresponds exactly to a library procedure that executes the primitive.

- The parameters for the service primitives and library procedures are as follows:

  - Connum = LISTEN(local)
  - Connum = CONNECT(local, remote)
  - Status = SEND (connum, buffer, bytes)
  - Status = RECEIVE (connum, buffer, bytes)
  - Status = DISCONNECT (connum)

- LISTEN primitive announces the caller's willingness to accept connection request.
- The CONNECT primitive takes two parameters, a local TSAP and remote TSAP.
- The SEND primitive transmits the contents of the buffer as a message.
- The RECEIVE primitive indicates the caller's desire to accept data.
- The DISCONNECT primitive terminates a transport connection.

The Example Transport Entity

Network packet	Meaning
CALL REQUEST	Sent to establish a connection
CALL ACCEPTED	Response to CALL REQUEST
CLEAR REQUEST	Sent to release a connection
CLEAR CONFIRMATION	Response to CLEAR REQUEST
DATA	Used to transport data
CREDIT	Control packet for managing the window

The network layer packets used in our example.

The Example Transport Entity (2)

Each connection is in one of seven states:

1. Idle – Connection not established yet.
2. Waiting – CONNECT has been executed, CALL REQUEST sent.
3. Queued – A CALL REQUEST has arrived; no LISTEN yet.
4. Established – The connection has been established.
5. Sending – The user is waiting for permission to send a packet.
6. Receiving – A RECEIVE has been done.
7. DISCONNECTING – a DISCONNECT has been done locally.

The Example as a Finite State Machine

- The example protocol as a finite state machine.
- Each entry has an optional predicate, an optional action, and the new state.
- The tilde indicates that no major action is taken.
- An overbar above a predicate indicate the negation of the predicate.
- Blank entries correspond to impossible or invalid events.

The Example as a Finite State Machine

		State						
		Idle	Waiting	Queued	Established	Sending	Receiving	Dis-connecting
Primitives	LISTEN	P1: ~/Idle P2: A1/Estab P2: A2/Idle		~/Estab				
	CONNECT	P1: ~/Idle P1: A3/Wait						
	DISCONNECT				P4: A5/Idle P4: A6/Disc			
	SEND				P5: A7/Estab P5: A8/Send			
	RECEIVE				A9/Receiving			
Incoming packets	Call_req	P3: A1/Estab P3: A4/Queu'd						
	Call_acc		~/Estab					
	Clear_req		~/Idle		A10/Estab	A10/Estab	A10/Estab	~/Idle
	Clear_conf							~/Idle
	DataPkt						A12/Estab	
	Credit				A11/Estab	A7/Estab		
Clock	Timeout			~/Idle				

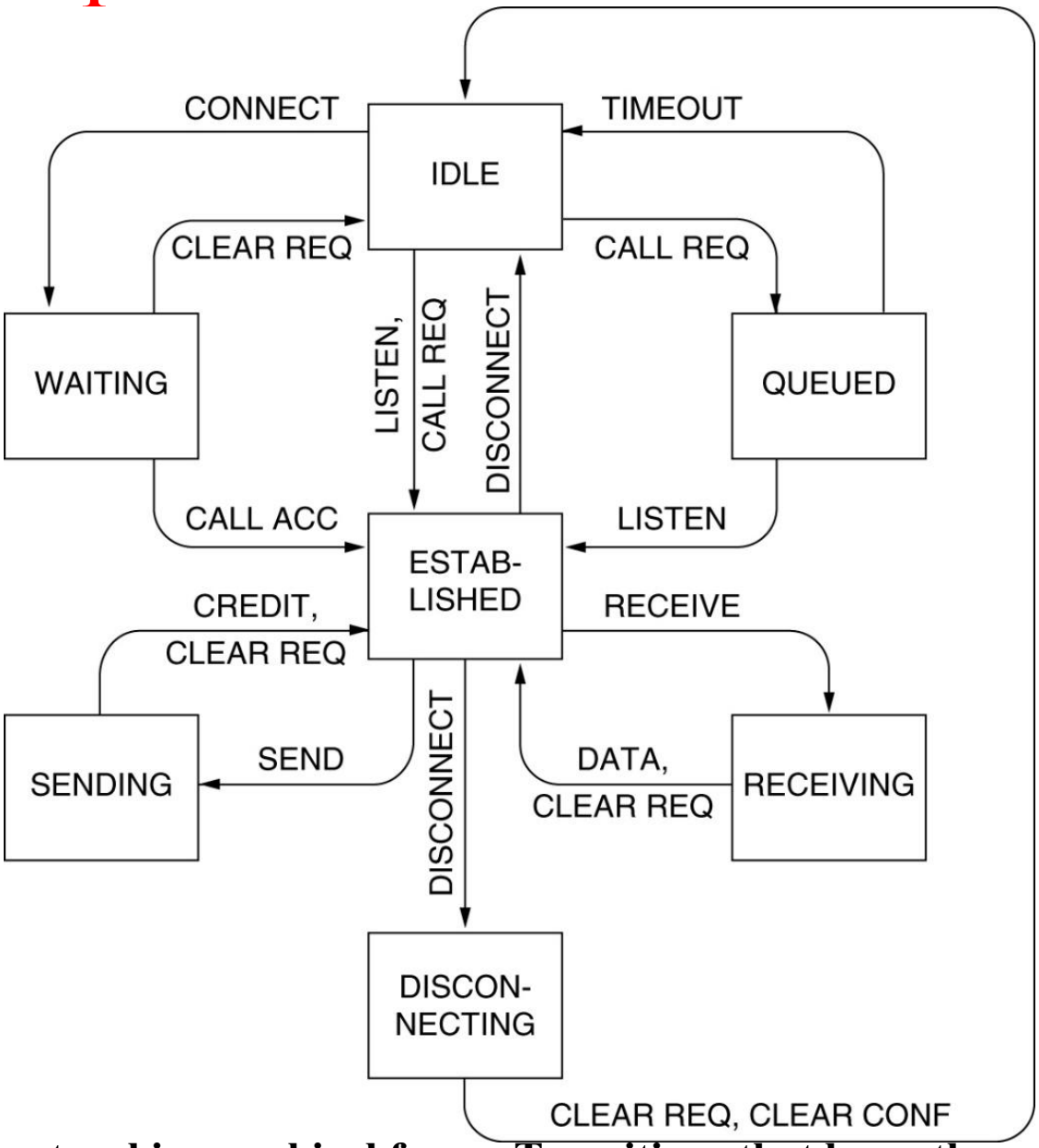
Predicates

P1: Connection table full
P2: Call_req pending
P3: LISTEN pending
P4: Clear_req pending
P5: Credit available

Actions

A1: Send Call_acc
A2: Wait for Call_req
A3: Send Call_req
A4: Start timer
A5: Send Clear_conf
A6: Send Clear_req
A7: Send message
A8: Wait for credit
A9: Send credit
A10: Set Clr_req_received flag
A11: Record credit
A12: Accept message

The Example as a Finite State Machine (2)



The example protocol in graphical form. Transitions that leave the connection state unchanged have been omitted for simplicity.

MLR Institute of Technology

Internet transport layer protocols: UDP and TCP

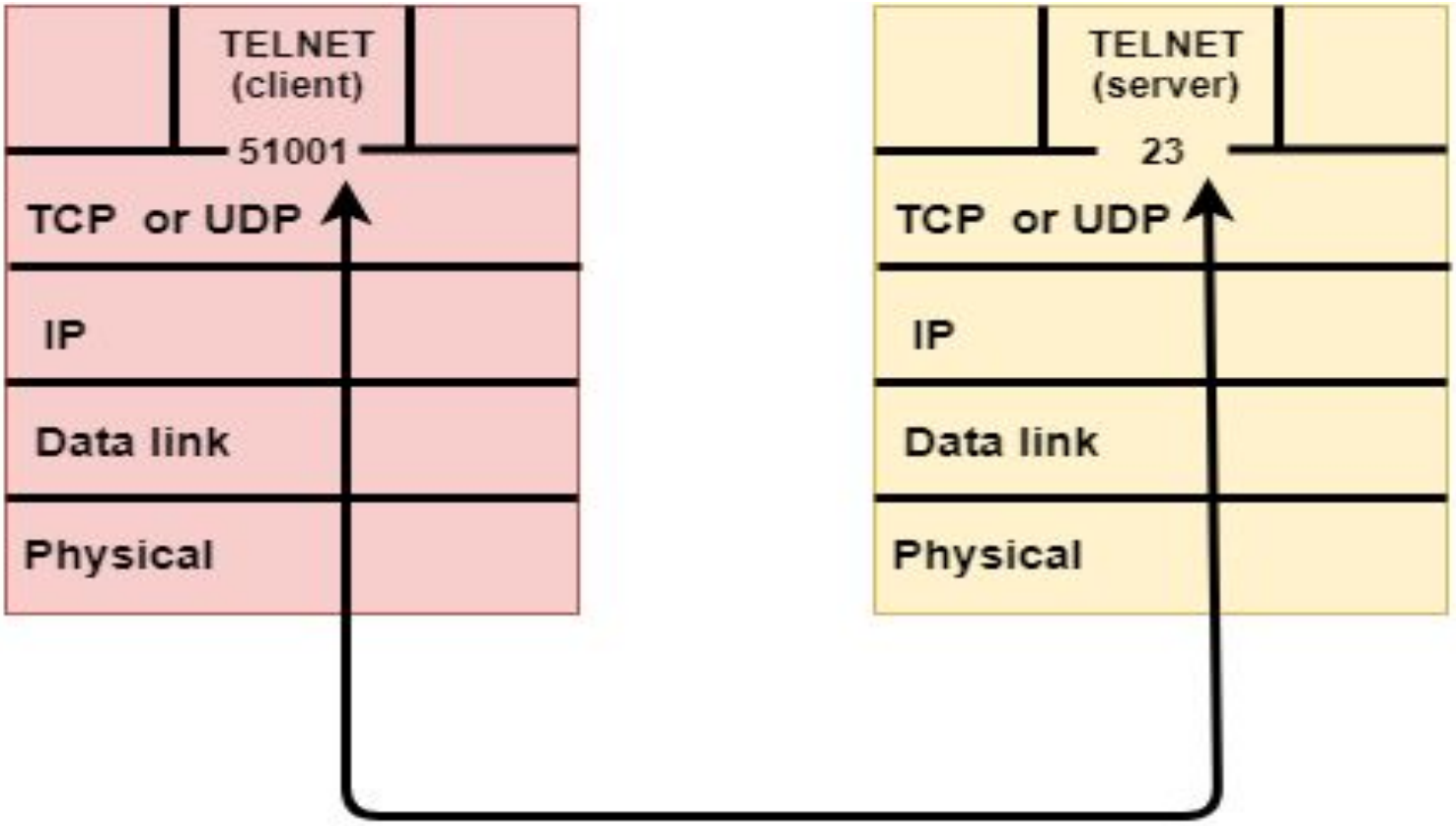
Transport Layer protocols:

- The transport layer is represented by two protocols: TCP and UDP.
- The IP protocol in the network layer delivers a datagram from a source host to the destination host.
- Nowadays, the operating system supports multiuser and multiprocessing environments, an executing program is called a process. When a host sends a message to other host means that source process is sending a process to a destination process. The transport layer protocols define some connections to individual ports known as protocol ports.
- An IP protocol is a host-to-host protocol used to deliver a packet from source host to the destination host while transport layer protocols are port-to-port protocols that work on the top of the IP protocols to deliver the packet from the originating port to the IP services, and from IP services to the destination port.
- Each port is defined by a positive integer address, and it is of 16 bits.

MLR Institute of Technology

Internet transport layer protocols: UDP and TCP

Transport Layer protocols:

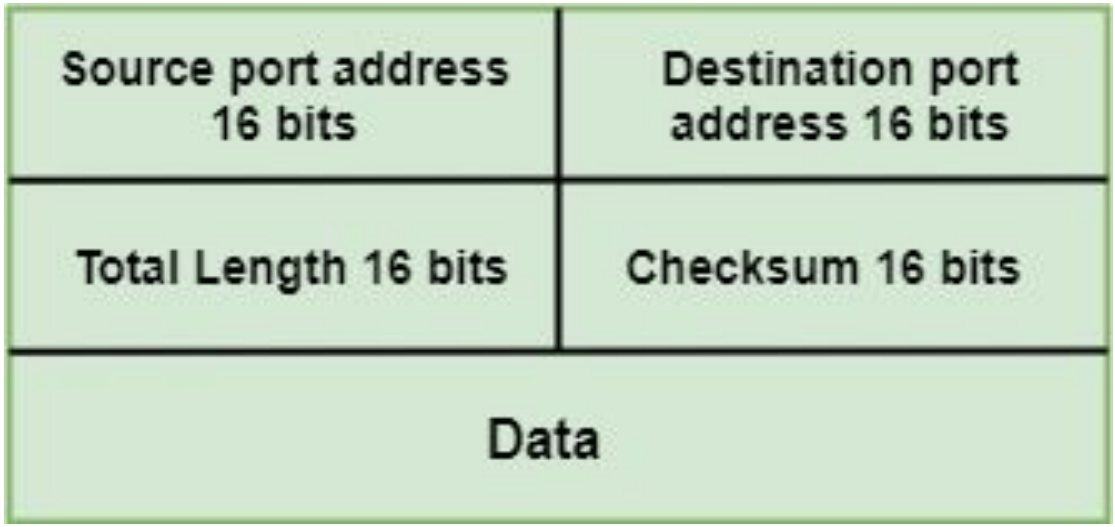


Transport Layer protocols: UDP

- UDP stands for **User Datagram Protocol**.
- UDP is a simple protocol and it provides non-sequenced transport functionality.
- UDP is a connectionless protocol.
- This type of protocol is used when reliability and security are less important than speed and size.
- UDP is an end-to-end transport level protocol that adds transport-level addresses, checksum error control, and length information to the data from the upper layer.
- The packet produced by the UDP protocol is known as a user datagram.

Transport Layer protocols: UDP

User Datagram Format



Where,

Source port address: It defines the address of the application process that has delivered a message. The source port address is of 16 bits address.

Destination port address: It defines the address of the application process that will receive the message. The destination port address is of a 16-bit address.

Total length: It defines the total length of the user datagram in bytes. It is a 16-bit field.

Checksum: The checksum is a 16-bit field which is used in error detection.

Transport Layer protocols: UDP

User Datagram Format

- The *UDP length* field includes the 8-byte header and the data. The *UDP checksum* is optional and stored as 0 if not computed (a true computed 0 is stored as all 1s).

- **UDP** (**User Datagram Protocol**). UDP provides a way for applications to send encapsulated IP datagrams and send them without having to establish a connection.
- UDP is described in RFC 768.
- UDP transmits **segments** consisting of an 8-byte header followed by the payload.
- When a UDP packet arrives, its payload is handed to the process attached to the destination port.
- The source port is primarily needed when a reply must be sent back to the source.
- UDP does not do flow control, error control, or retransmission upon receipt of a bad segment.

Transport Layer protocols: UDP

- One area where UDP is especially useful is in client-server situations.
- Often, the client sends a short request to the server and expects a short reply back. If either the request or reply is lost, the client can just time out and try again.
- An application that uses UDP this way is DNS (the Domain Name System).

Transport Layer protocols: UDP

Disadvantages of UDP protocol:

- UDP provides basic functions needed for the end-to-end delivery of a transmission.
- It does not provide any sequencing or reordering functions and does not specify the damaged packet when reporting an error.
- UDP can discover that an error has occurred, but it does not specify which packet has been lost as it does not contain an ID or sequencing number of a particular data segment.

Remote Procedure Call

- In a certain sense, sending a message to a remote host and getting a reply back is a lot like making a function call in a programming language.
- The key work in this area was done by Birrell and Nelson (1984).
- what Birrell and Nelson suggested was allowing programs to call procedures located on remote hosts.
- When a process on machine 1 calls a procedure on machine 2, the calling process on 1 is suspended and execution of the called procedure takes place on 2.
- Information can be transported from the caller to the callee in the parameters and can come back in the procedure result.
- No message passing is visible to the programmer. This technique is known as **RPC** (**Remote Procedure Call**).
- Traditionally, the calling procedure is known as the client and the called procedure is known as the server,

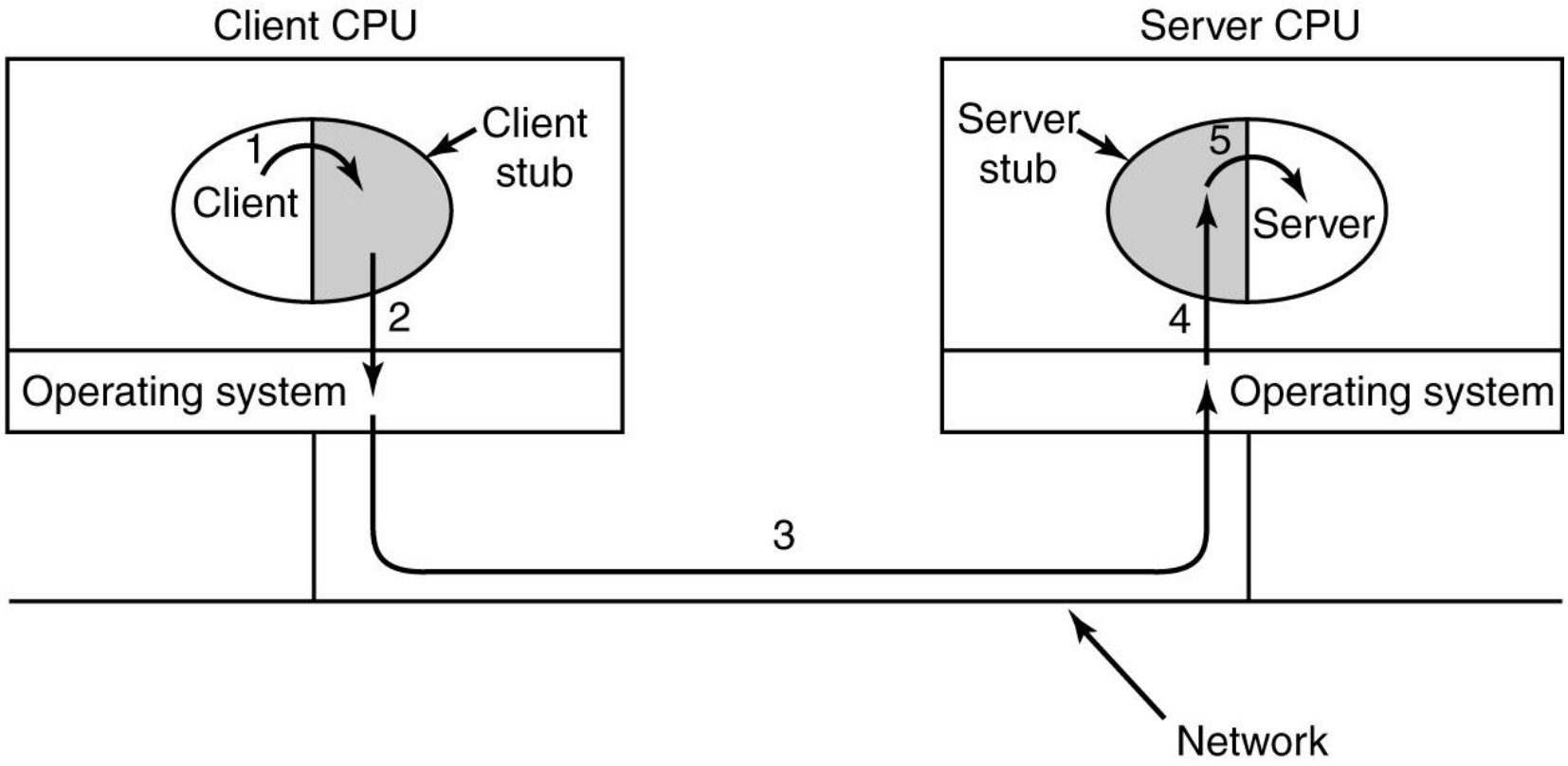
Remote Procedure Call

- The idea behind RPC is to make a remote procedure call look as much as possible like a local one.
- In the simplest form, to call a remote procedure,
- the client program must be bound with a small library procedure, called the **client stub**, that represents the server procedure in the client's address space.
- Similarly, the server is bound with a procedure called the **server stub**. These procedures hide the fact that the procedure call from the client to the server is not local.

Remote Procedure Call

- The actual steps in making an RPC are :
- Step 1 is the client calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.
- Step 2 is the client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called **marshaling**.
- Step 3 is the kernel sending the message from the client machine to the server machine.
- Step 4 is the kernel passing the incoming packet to the server stub. Finally,
- step 5 is the server stub calling the server procedure with the unmarshaled parameters. The reply traces the same path in the other direction.

Remote Procedure Call



Steps in making a remote procedure call. The stubs are shaded.

Remote Procedure Call

- There are few problems in using RPC :
- one is the use of pointer parameters.
- A second problem is that in weakly-typed languages, like C, it is perfectly legal to write a procedure that computes the inner product of two vectors (arrays), without specifying how large either one is. Each could be terminated by a special value known only to the calling and called procedure. Under these circumstances, it is essentially impossible for the client stub to marshal the parameters: it has no way of determining how large they are.
- A third problem is that it is not always possible to deduce the types of the parameters, not even from a formal specification or the code itself.
- A fourth problem relates to the use of global variables.
- These problems are not meant to suggest that RPC is hopeless. In fact, it is widely used,

Transport Layer protocols: TCP

- TCP stands for Transmission Control Protocol.
- It provides full transport layer services to applications.
- It is a connection-oriented protocol means the connection established between both the ends of the transmission.
- For creating the connection, TCP generates a virtual circuit between sender and receiver for the duration of a transmission.

MLR Institute of Technology

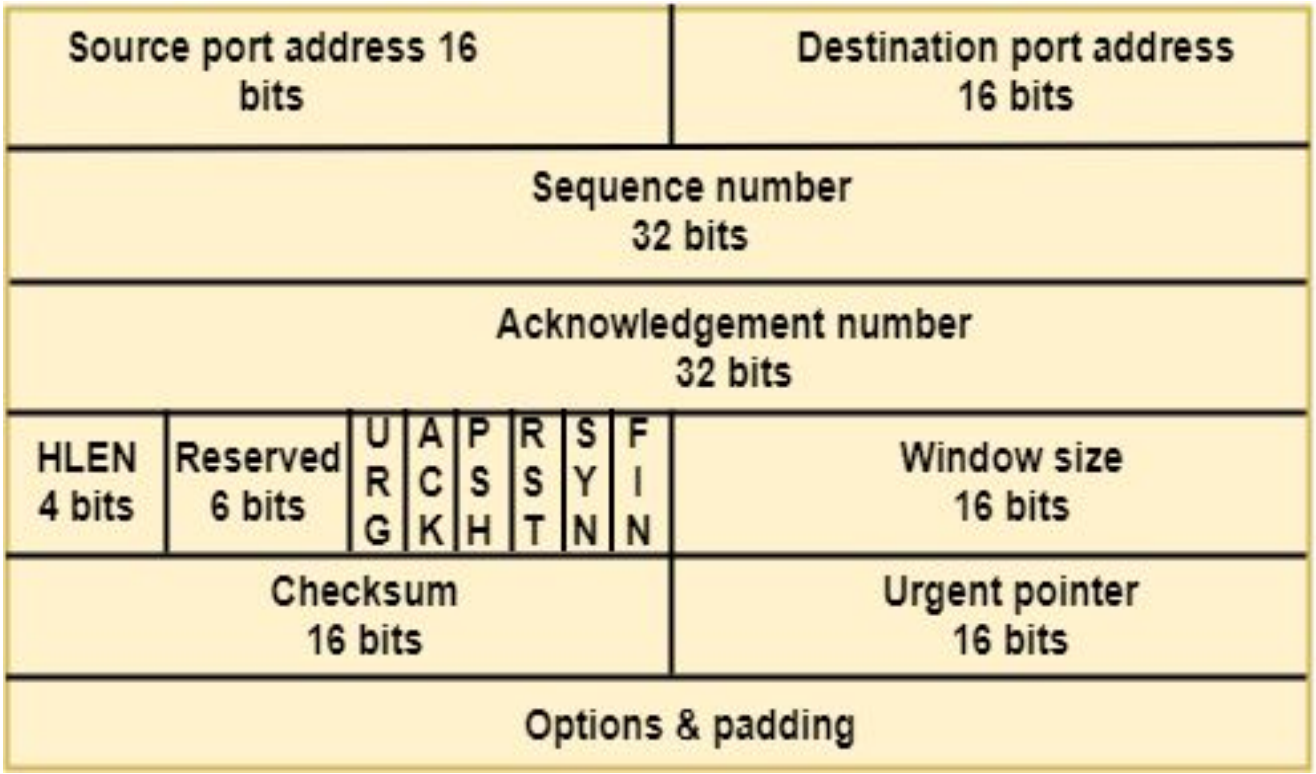
Transport Layer protocols: TCP

Features Of TCP protocol

- Stream data transfer
- Reliability
- Flow Control
- Multiplexing
- Logical connections
- Full Duplex

Transport Layer protocols: TCP

TCP Segment Format



Transport Layer protocols: TCP

Where,

- **Source port address:** It is used to define the address of the application program in a source computer. It is a 16-bit field.

- **Destination port address:** It is used to define the address of the application program in a destination computer. It is a 16-bit field.

- **Sequence number:** A stream of data is divided into two or more TCP segments. The 32-bit sequence number field represents the position of the data in an original data stream.

Transport Layer protocols: TCP

Acknowledgement number: A 32-field acknowledgement number acknowledge the data from other communicating devices. If ACK field is set to 1, then it specifies the sequence number that the receiver is expecting to receive.

Header Length (HLEN): It specifies the size of the TCP header in 32-bit words. The minimum size of the header is 5 words, and the maximum size of the header is 15 words. Therefore, the maximum size of the TCP header is 60 bytes, and the minimum size of the TCP header is 20 bytes.

Reserved: It is a six-bit field which is reserved for future use.

Control bits: Each bit of a control field functions individually and independently. A control bit defines the use of a segment or serves as a validity check for other fields.

Transport Layer protocols: TCP

There are total six types of flags in control field:

URG: The URG field indicates that the data in a segment is urgent.

ACK: When ACK field is set, then it validates the acknowledgement number.

PSH: The PSH field is used to inform the sender that higher throughput is needed so if possible, data must be pushed with higher throughput.

RST: The reset bit is used to reset the TCP connection when there is any confusion occurs in the sequence numbers.

SYN: The SYN field is used to synchronize the sequence numbers in three types of segments: connection request, connection confirmation (with the ACK bit set), and confirmation acknowledgement.

FIN: The FIN field is used to inform the receiving TCP module that the sender has finished sending data. It is used in connection termination in three types of segments: termination request, termination confirmation, and acknowledgement of termination confirmation.

Window Size: The window is a 16-bit field that defines the size of the window.

Checksum: The checksum is a 16-bit field used in error detection.

Urgent pointer: If URG flag is set to 1, then this 16-bit field is an offset from the sequence number indicating that it is a last urgent data byte.

Options and padding: It defines the optional fields that convey the additional information to the receiver.

Transport Layer protocols: TCP and UDP

Differences b/w TCP & UDP

Basis for Comparison	TCP	UDP
Definition	TCP establishes a virtual circuit before transmitting the data.	UDP transmits the data directly to the destination computer without verifying whether the receiver is ready to receive or not.
Connection Type	It is a Connection-Oriented protocol	It is a Connectionless protocol
Speed	slow	high
Reliability	It is a reliable protocol.	It is an unreliable protocol.
Header size	20 bytes	8 bytes
acknowledgement	It waits for the acknowledgement of data and has the ability to resend the lost packets.	It neither takes the acknowledgement, nor it retransmits the damaged frame.