UNIT-4

Transport Layer

Segmentation
Transport
Reassembly

B SAI BABA**,M.Tech(Ph.D),**VIT,Bhimavaram

Syllabus

★ The Transport Layer

- Transport service
- Elements of transport protocol
- Simple Transport Protocol
- Internet transport layer protocols: UDP and TCP

THE TRANSPORT LAYER SERVICES

1. Services Provided to the Upper Layers

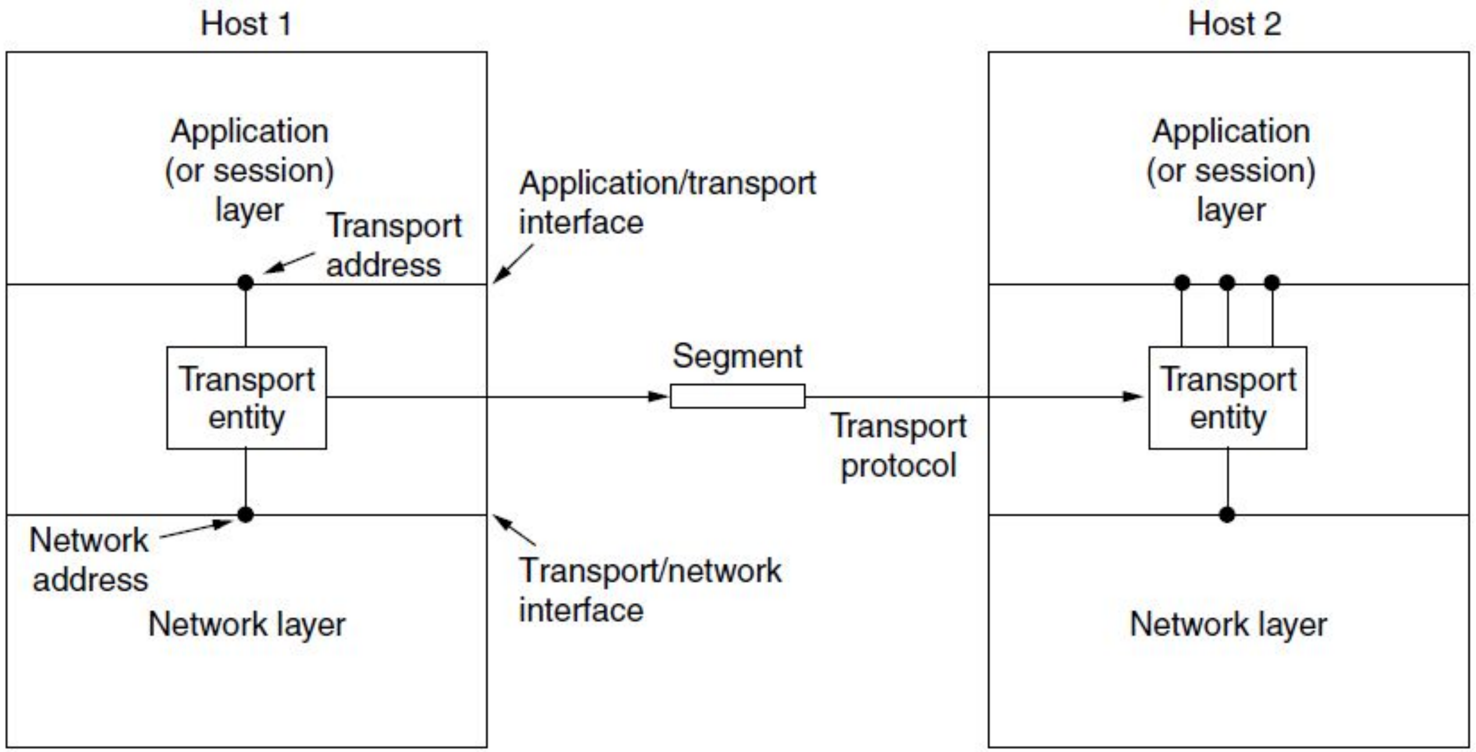


Figure 6-1. The network, transport, and application layers.

- The ultimate goal of the transport layer is to provide **efficient, reliable, and cost-effective data transmission service** to its users, normally processes in the application layer.
- To achieve this, the transport layer makes use of the services provided by the network layer.
- The software and/or hardware within the transport layer that does the work is called **the transport entity**.
- The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card.
- The (logical) relationship of the network, transport, and application layers is illustrated in Fig. 6-1.

Working of Transport Layer:

The transport layer **takes services from the Application layer** and **provides services to the Network layer .**

At the sender's side	At the receiver's side
● The transport layer receives data (message) from the Application layer and then performs **Segmentation**, divides the actual message into **segments**, **adds the source and destination's port numbers** into the header of the segment, and transfers the message to the Network layer.	● The transport layer receives data from the Network layer, reassembles the segmented data, **reads its header, identifies the port number**, and forwards the message to the appropriate port in the Application layer.

Responsibilities of a Transport Layer:

- The Process to Process Delivery
- End-to-End Connection between Hosts
- Multiplexing and Demultiplexing
- Congestion Control
- Error Control
- Flow control

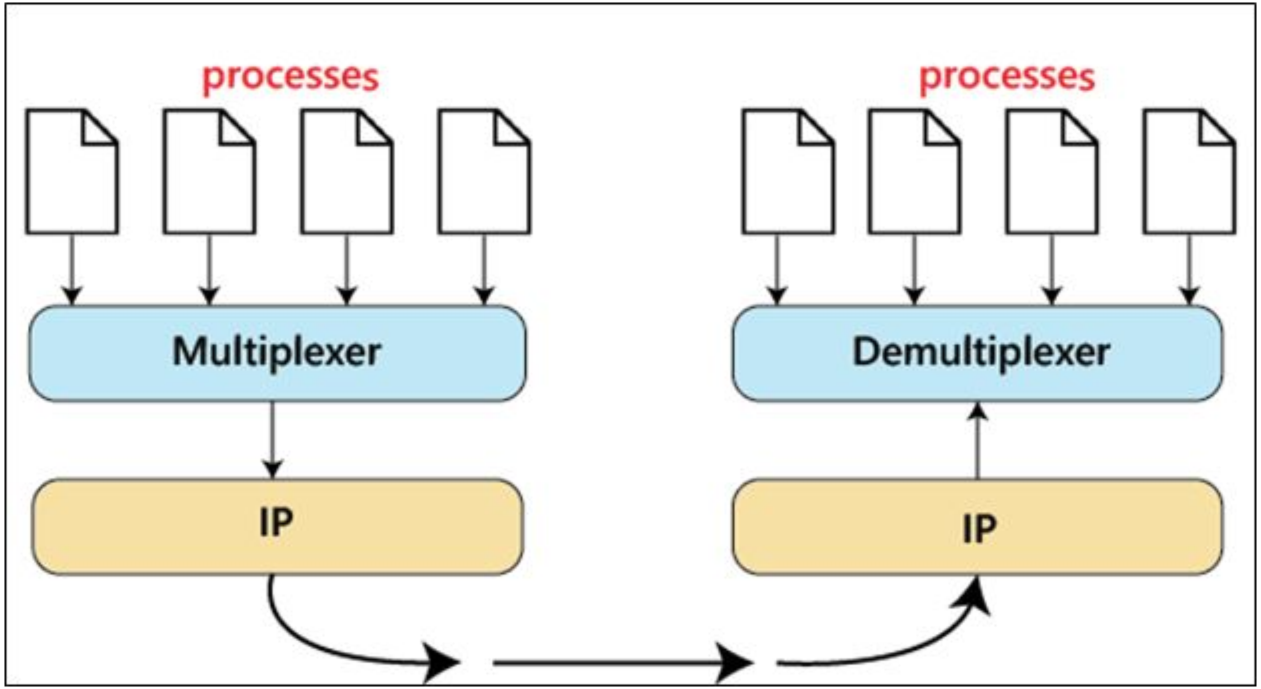
The Process to Process Delivery

Sender
Receiver

1. The Process to Process Delivery

- While
  - **Data Link Layer requires** the **MAC address** of source-destination hosts to correctly deliver **a frame** ,
  - **The Network layer** requires **the IP address** for **appropriate routing of packets**,
  - In a similar way **Transport Layer** requires **a Port number** to **correctly deliver the segments of data to the correct process** amongst the multiple processes running on a particular host.
  - **A port number** is a **16-bit address** used to identify any client-server program uniquely.

Process to Process Delivery



PORT?

- The transport layer is responsible for ensuring the reliable transfer of data between two devices in a network.
- Ports play a crucial role in the transport layer, and **they are used to facilitate communication between applications or services running on different devices.**

Port Definition:

★ A port is **a 16-bit unsigned integer (ranging from 0 to 65535)** that serves as an endpoint for communication.

★ Ports are used to distinguish between different services or applications running on the same device, allowing multiple services to operate concurrently.

★ For example, **a web server** typically listens on **port 80 for HTTP requests**, while an **email server** might use **port 25 for SMTP** (Simple Mail Transfer Protocol) communication.

★ A combination of an **IP address and a port number** is referred to as **a socket.**

Difference between PORT,IP and MAC Address

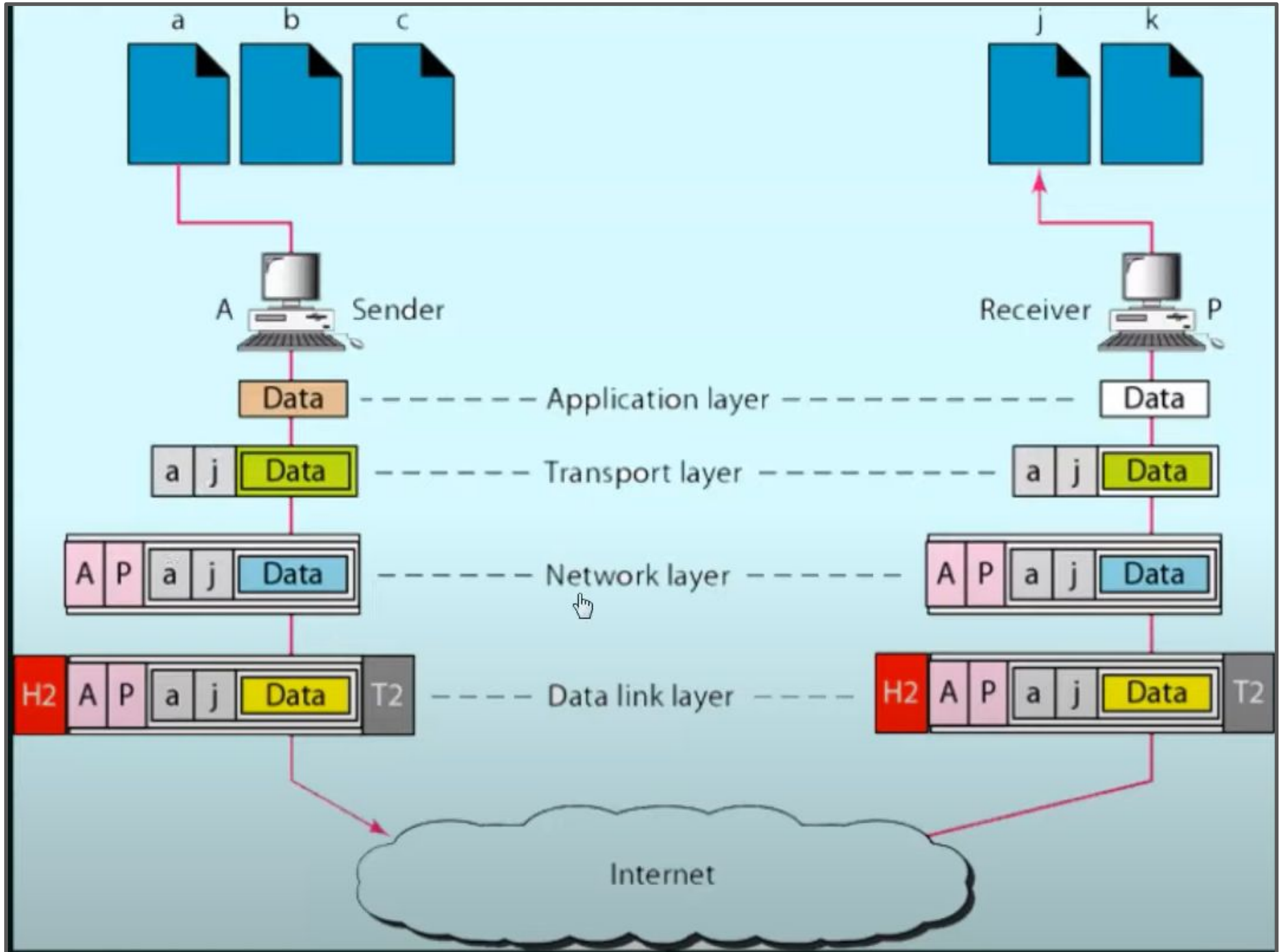
Suppose your friends want to send a parcel and he/she is in **America** and you are in **Bhimavaram**.

★ **Reaching your CITY = Reaching Your NETWORK (IP Address)**

★ **Reaching Your APARTMENT = Reaching the HOST (MAC Address)**

★ **Reaching the RIGHT PERSON = Reaching the right PROCESS (PORT Address)**

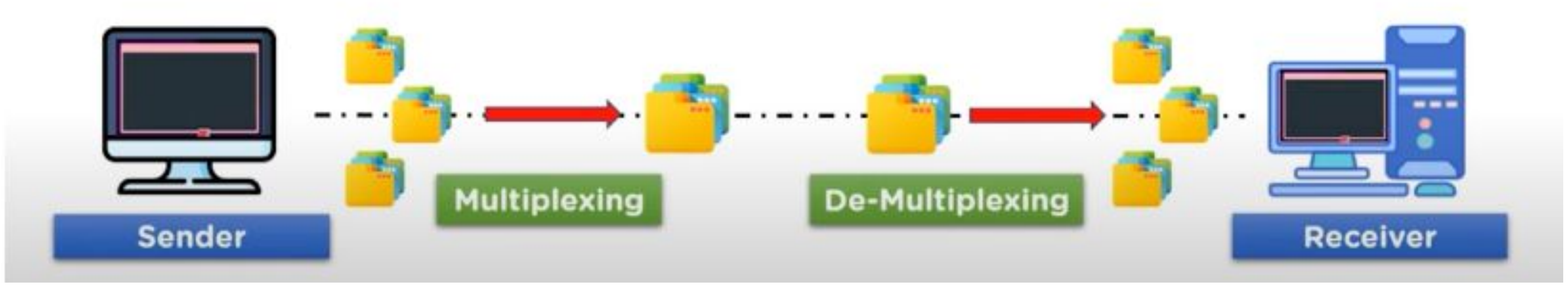
Relation between PORT, IP, MAC Address



2. End-to-end Connection between Hosts

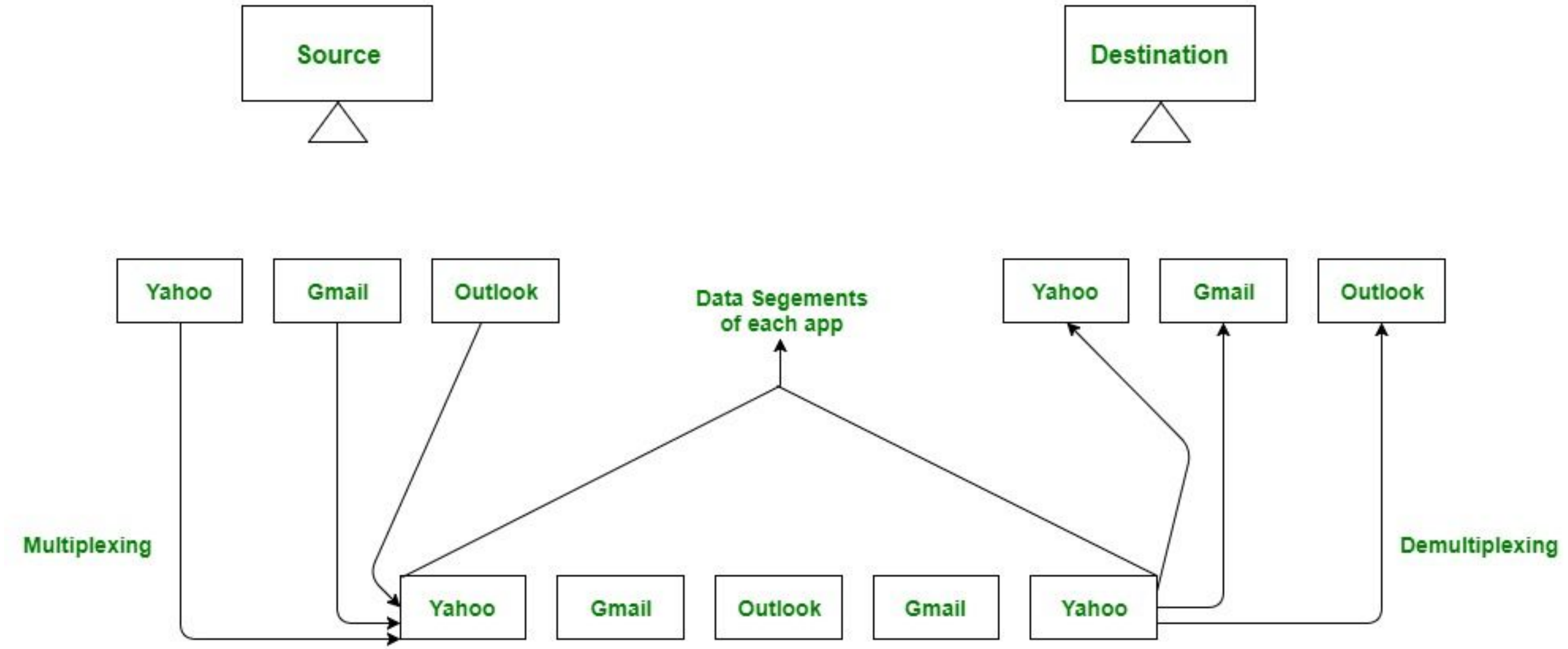
- The transport layer is also responsible for **creating the end-to-end Connection between hosts** for which it mainly uses **TCP and UDP**.
- **TCP** is a secure, connection-oriented protocol that uses a handshake protocol to establish a robust connection between two end hosts.
  - TCP ensures **the reliable** delivery of messages and is used in various applications.
- **UDP**, on the other hand, is a stateless and **unreliable protocol** that ensures best-effort delivery.
  - It is suitable for applications that have little concern with flow or error control and requires sending the bulk of data like video conferencing.
  - It is often used in multicasting protocols.

3. Multiplexing and Demultiplexing



"Multiplexing and Demultiplexing, enable the transport layer ***to manage multiple communication streams over a single network connection****, ensuring that data from different applications or sources can be sent and received correctly."*

Multiplexing / Demultiplexing



1. Multiplexing (Muxing):

Definition: Multiplexing is the process of combining multiple data streams or communication channels into a single data stream for transmission over a network.

Purpose: It allows multiple applications or communication sources to share the same network connection efficiently.

How it works: When data is sent from different applications or sources on a device, the transport layer adds headers or tags to each data segment. These headers contain information that helps identify the source or destination of the data. The transport layer then combines these data segments into a single stream, often referred to as a "multiplexed stream" or "multiplexed connection," before transmitting it over the network.

2. Demultiplexing (Demuxing):

Definition: Demultiplexing is the process of separating the combined multiplexed data stream into individual data streams or communication channels upon reception.

Purpose: It ensures that data from different sources can be correctly routed to their respective applications or destinations.

How it works: When data is received over the network, the transport layer examines the headers or tags in the incoming data to determine which application or source each segment belongs to. Based on this information, it separates the multiplexed data stream into individual data streams, each destined for a specific application or source. These individual data streams are then delivered to the appropriate higher-layer protocols or applications.

Elements of Transport Protocol

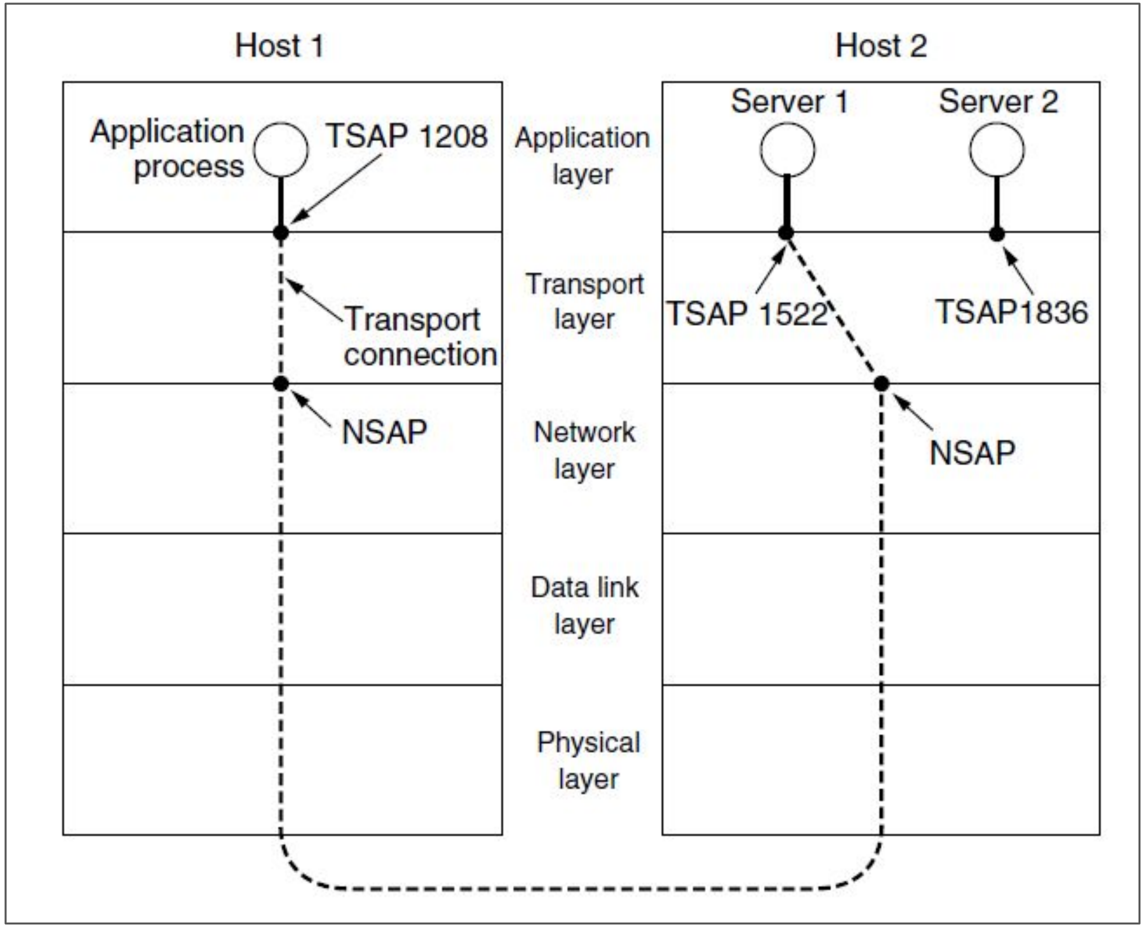
Elements of Transport Protocol

- **Addressing**
- **Connection Establishment**
- **Connection Release**
- **Flow Control and Buffering**
- **Multiplexing**

Addressing

- When an application (e.g., a user) process wishes to set up a connection to a remote application process, **it must specify which one to connect to**.
- The method normally used is **to define transport addresses to which processes can listen for connection requests**. In the Internet, these endpoints are called **ports**.
- We will use the generic term **TSAP** (Transport Service Access Point) to mean a specific endpoint in the transport layer.
- The analogous endpoints in the network layer (i.e., network layer addresses) are called NSAPs (Network Service Access Points). **IP addresses** are examples of **NSAP**s.
- Figure 6-8 illustrates the relationship between the NSAPs, the TSAPs, and a transport connection.
- Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP.
- These connections run through NSAPs on each host, as shown. The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport endpoints that share that NSAP.

The Relationship Between TSAPs, NSAPs and Transport Connections



A possible scenario for a transport connection is as follows:

1. **A mail server** process attaches itself to **TSAP 1522** on host 2 to wait for an incoming call.

2. **An application process** on host 1 wants to send an email message, so it attaches itself to **TSAP 1208** and issues **a CONNECT request**. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.

3. The application process sends over the mail message.

4. The mail server responds to say that it will deliver the message.

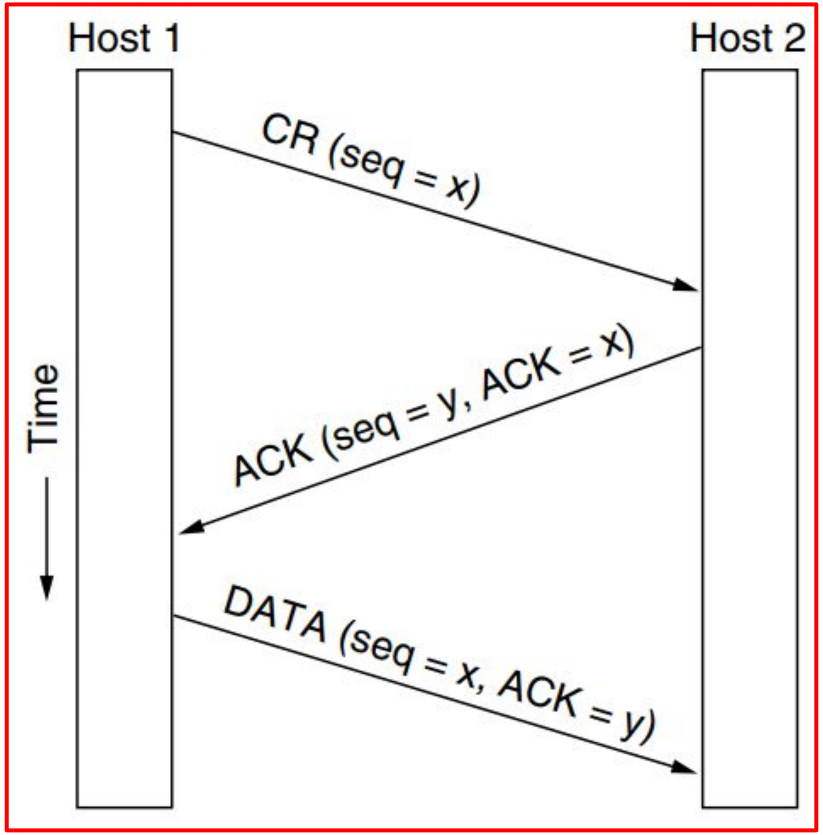
5. The transport connection is released.

Connection Establishment

- Establishing a connection sounds easy, but it is actually surprisingly tricky.
- At first glance, it would seem sufficient for one transport entity to just send a **CONNECTION REQUEST** segment to **the destination** and wait for a **CONNECTION ACCEPTED reply**.
- **The problem occurs** when the network can lose, delay, corrupt,and duplicate packets. This behavior causes serious complications.
- Imagine a subnet that is so congested that acknowledgements hardly ever get back in time and each packet times out is retransmitted two or more times
- **Tomlinson (1975) introduced the three-way handshake,** to handle **duplications** in transmission process.

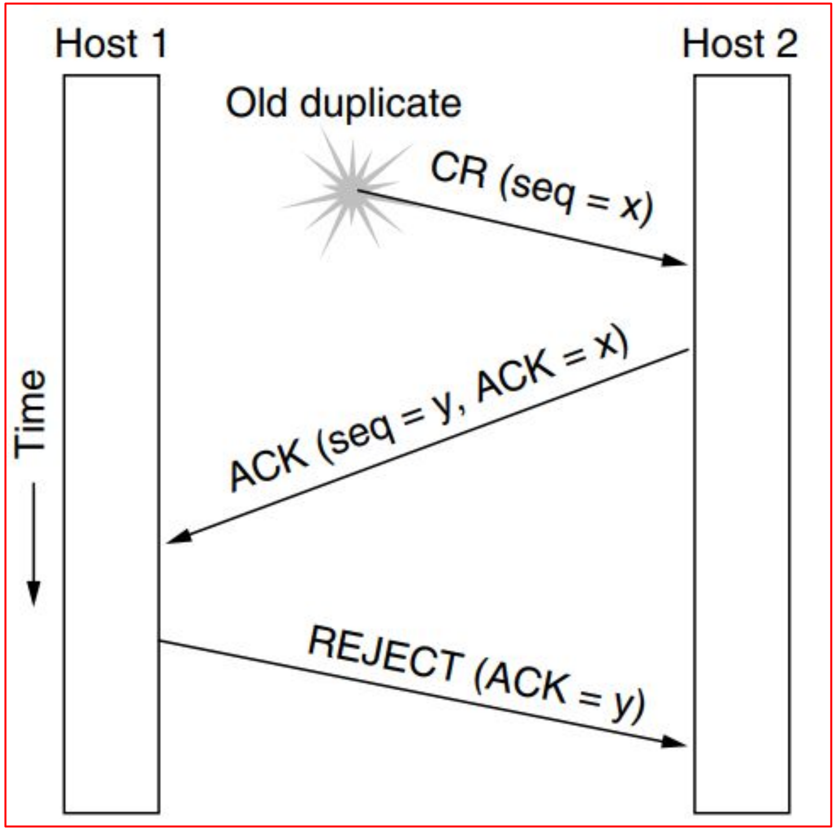
Normal Procedure

- Host 1 chooses a sequence number x, and sends a CONNECTION REQUEST segment containing it to host 2.
- Host 2 replies with an ACK segment 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 segment that it sends.



Abnormal situation: Old duplicate CONNECTION REQUEST appearing out of nowhere.

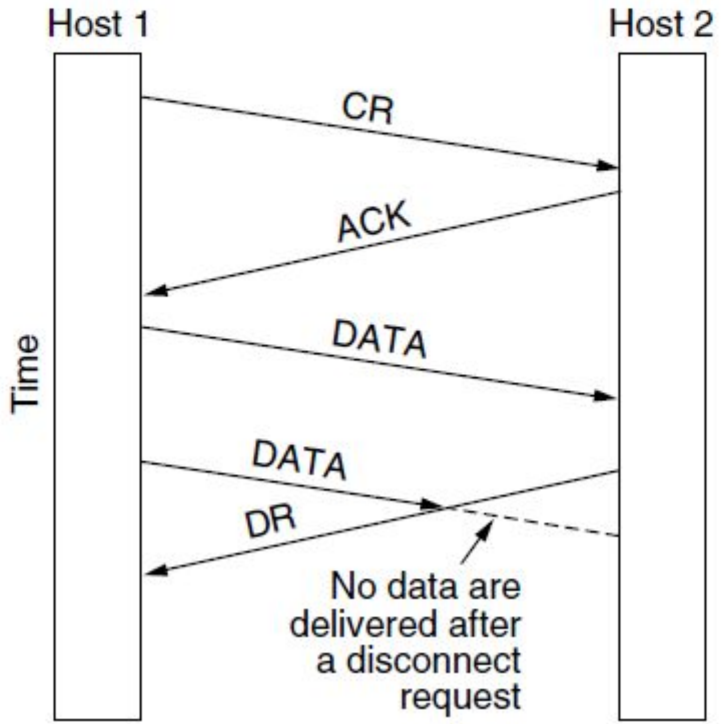
- The first segment is **a delayed duplicate CONNECTION REQUEST** from an old connection.
- This segment arrives at host 2 without host 1's knowledge.
- Host 2 reacts to this segment by sending host 1 an ACK segment, in effect asking for verification that host 1 was indeed trying to set up a new connection.
- 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.



Connection Release

- Releasing a connection is easier than establishing one.
- There are two styles of terminating a connection:
    - **Asymmetric release**
    - **Symmetric release**
- **Asymmetric release** is a method of terminating a network connection **where one side of the connection initiates the release without waiting for the other side to confirm or acknowledge the release. may result,** It may result **data loss**.
- **Symmetric release**, on the other hand, is a method **where both endpoints of a network connection agree to release the connection simultaneously.**

Asymmetric Release



Abrupt disconnection with loss of data

Symmetric Release

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

Eg:

- Host 1: I am done, are you done too?
- Host 2: I am done too, goodbye

Figures:Illustrates four scenarios of releasing using a three-way handshake.

(a) Normal case of three-way handshake

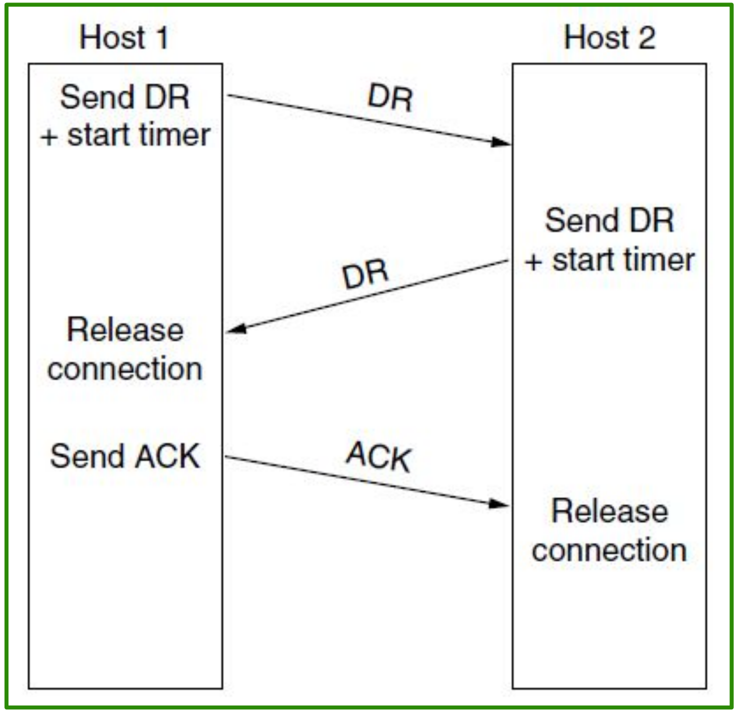
(b) Final ACK lost

(c) Response lost

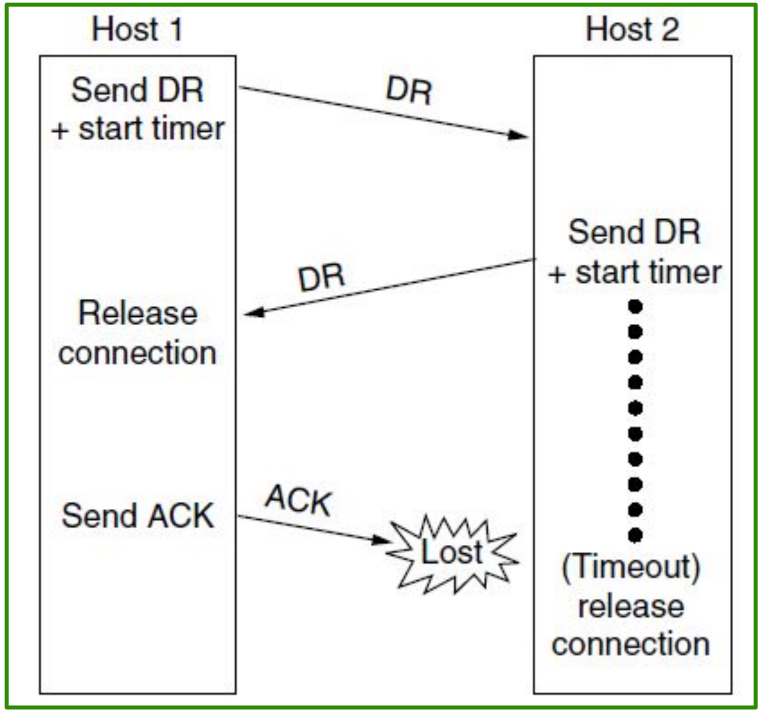
(d) Response lost and subsequent DRs lost

Normal case

Final ACK lost



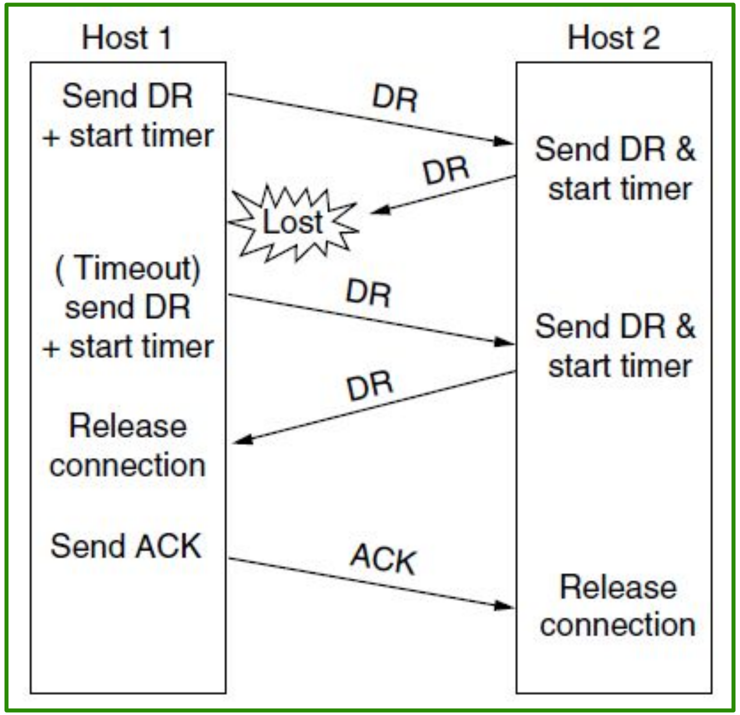
[a]



[b]

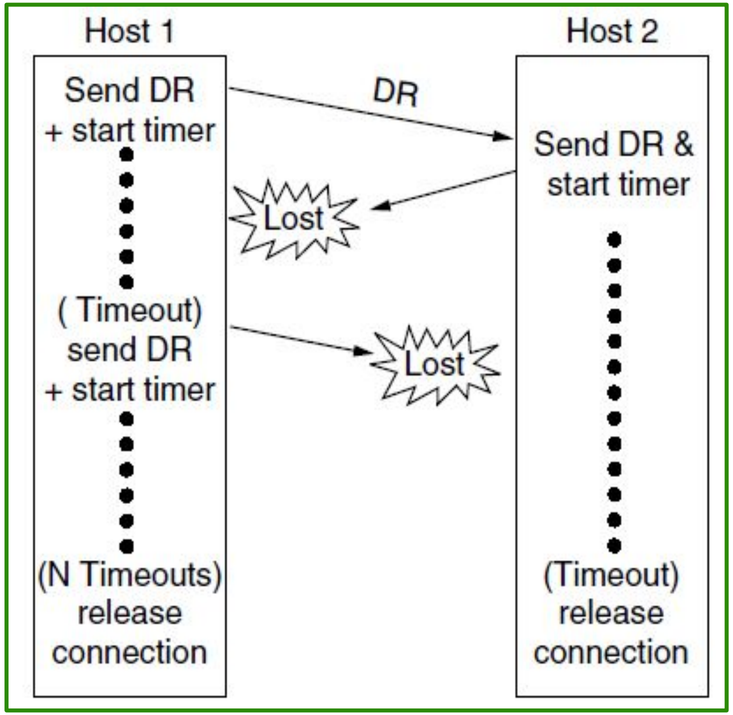
DR-DISCONNECTION REQUEST

Response lost



[c]

Response lost and subsequent DRs lost

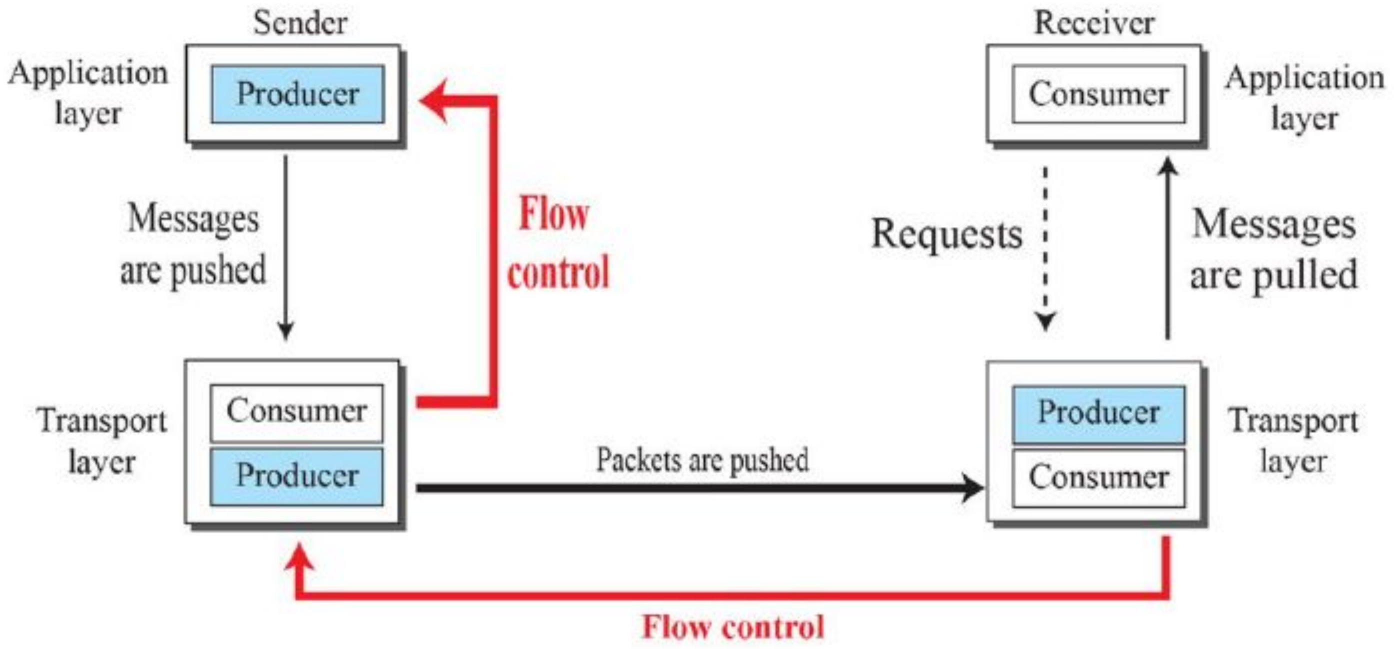


[d]

- **In Fig. 6-14(a)**, we see the normal case in which one of the users sends a DR (DISCONNECTION REQUEST) segment to initiate the connection release. When it arrives, the recipient sends back a DR segment and starts a timer, just in case its DR is lost. When this DR arrives, the original sender sends back an ACK segment and releases the connection. Finally, when the ACK segment arrives, the receiver also releases the connection. Releasing a connection means that the transport entity removes the information about the connection from its table of currently open connections and signals the connection's owner (the transport user) somehow. This action is different from a transport user issuing a DISCONNECT primitive.
- If the final ACK segment is lost, as shown in **Fig. 6-14(b)**, the situation is saved by the timer. When the timer expires, the connection is released anyway.Now consider the case of the second DR being lost. The user initiating the disconnection will not receive the expected response, will time out, and will start all over again.
- **In Fig. 6-14(c)**, we see how this works, assuming that the second time no segments are lost and all segments are delivered correctly and on time.
- **Our last scenario, Fig. 6-14(d)**, is the same as Fig. 6-14(c) except that now we assume all the repeated attempts to retransmit the DR also fail due to lost segments.After N retries, the sender just gives up and releases the connection.Meanwhile, the receiver times out and also exits.

Flow Control and Buffering

Flow control at the transport layer



Flow Control at Transport Layer:

- In communication at the transport layer, we are dealing with four entities:
  - Sender Process,
  - Sender Transport Layer,
  - Receiver Transport Layer
  - Receiver Process
- The sending process at the application layer is only **a producer.** It produces message chunks and pushes them to the transport layer. The sending transport layer has a double role: it is both **a consumer and a producer.**
- It consumes the messages pushed by the producer. It encapsulates the messages in packets and pushes them to the receiving transport layer.
- The receiving transport layer also has a double role: it is the consumer for the packets received from the sender and the producer that decapsulates the messages and delivers them to the application layer.

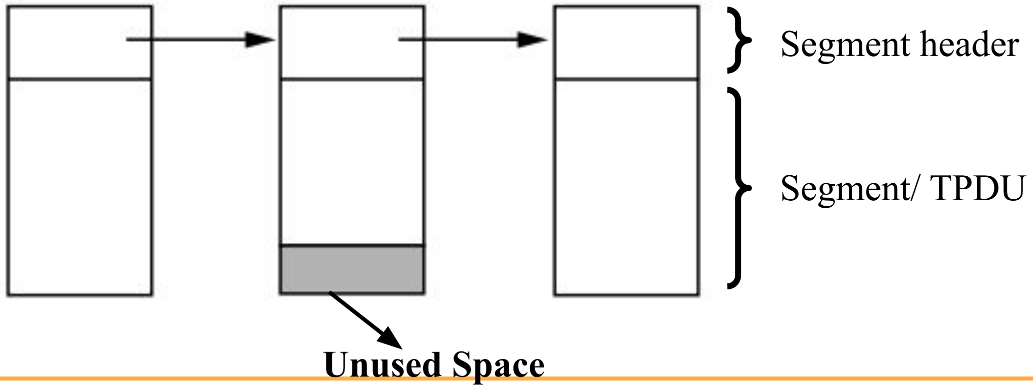
Buffers

- Flow control in the transport layer is a mechanism used to **manage the rate of data transmission** between two devices to ensure that the sender does not overwhelm the receiver with data.
- Buffering is one of the methods employed in flow control to achieve this goal. Buffering involves the use of temporary storage, called buffers, at various points in the communication process to hold data temporarily until it can be transmitted or processed.
- **A buffer** is **a set of memory locations that can hold segments** at the sender side and receiver side.
- **Sender Buffer:** The sender maintains a sender buffer that stores the data it wants to transmit. When the sender application generates data, it places it into the sender buffer.
- **Receiver Buffer:** The receiver maintains a receiver buffer to temporarily hold incoming data. As data arrives at the receiver, it is placed into the receiver buffer. The receiver then processes the data from the buffer at its own pace.

- The sender buffers all the TPDUs sent to the receiver. The buffer size varies for different TPDUs.The buffer size varies for different TPDUs.
- They are:
  - **Chained Fixed-size Buffers**
  - **Chained Variable-size Buffers**
  - **One large Circular Buffer per Connection**

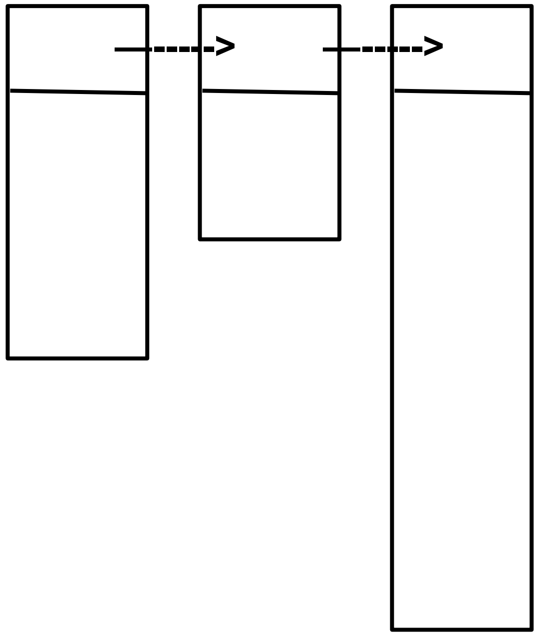
Chained Fixed-size Buffers

- **If most segments are nearly the same size**, it is natural to organize the buffers as a pool of identically sized buffers, with one segment per buffer



Chained Variable-size Buffers:

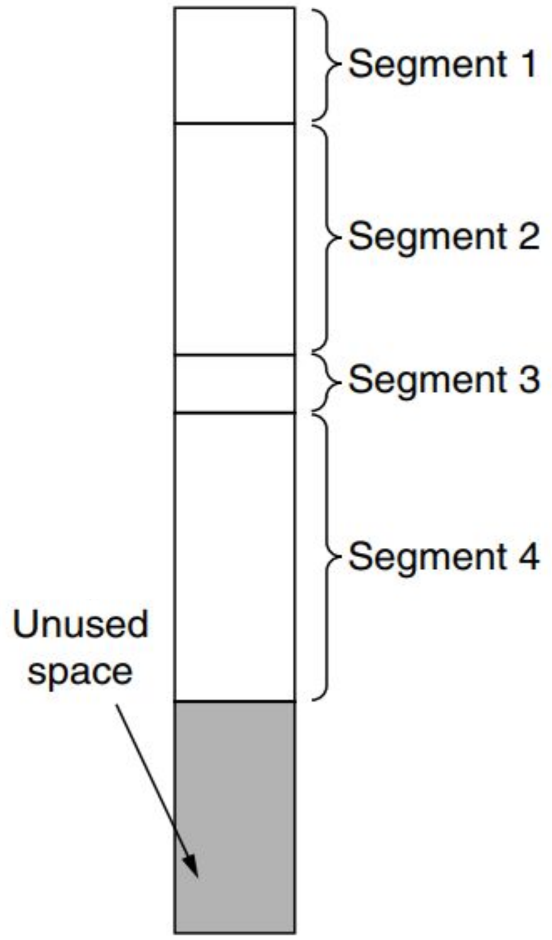
- Chained variable-size buffers extend the concept of chained fixed size buffers but allow for buffers of varying sizes to be linked together.
- Unlike fixed-size buffers, variable-size buffers can adapt to the size of the data being processed.
- This can be more **memory-efficient** when dealing with data of varying sizes.



Chained Variable-size Buffers

One large Circular Buffer per Connection:

- **A single large circular buffer per connection** is dedicated when all connections are heavily loaded.
- This system is simple and elegant and does not depend on segment sizes, but makes good use of memory only when the connections are heavily loaded.



Multiplexing

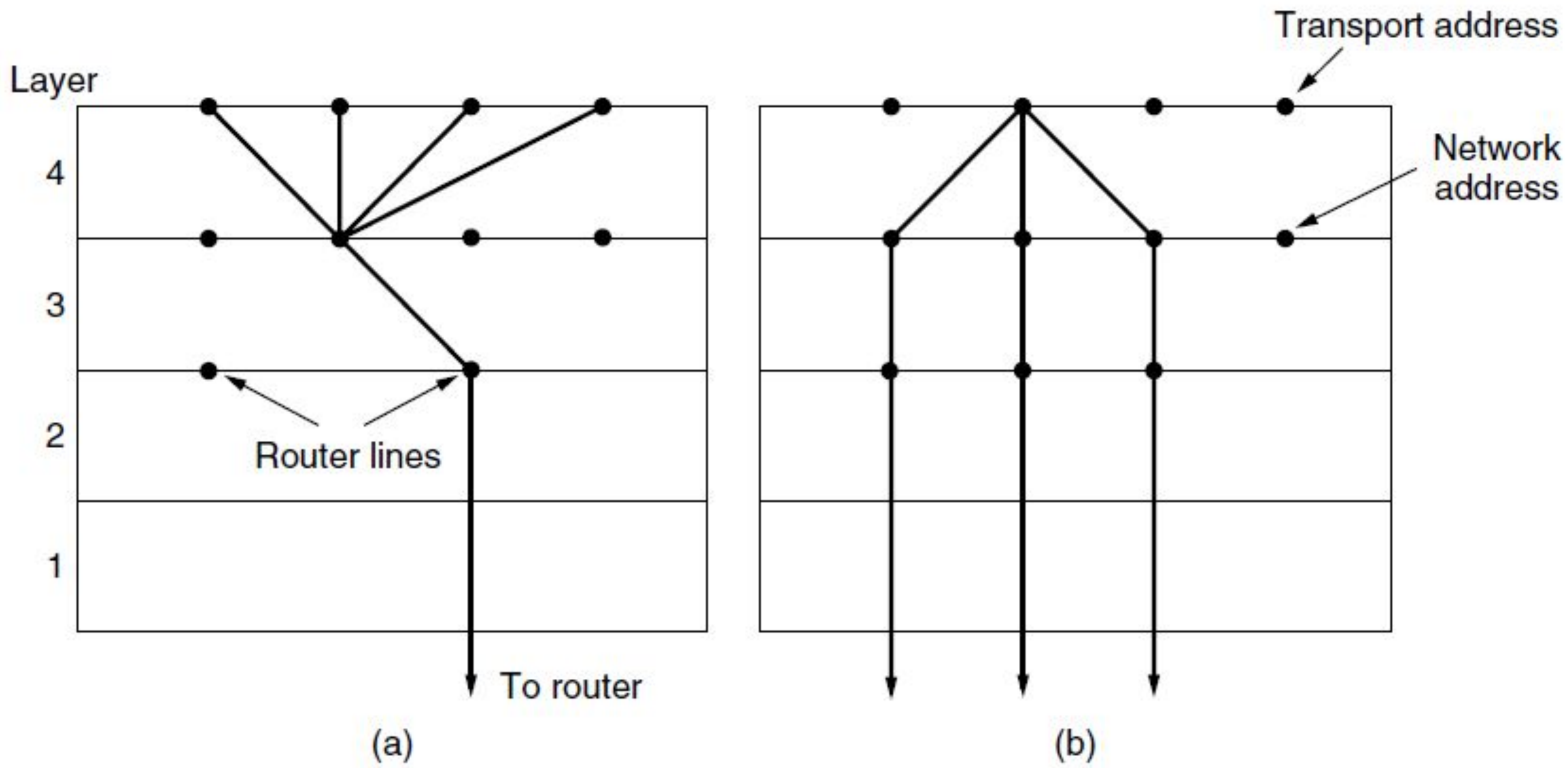


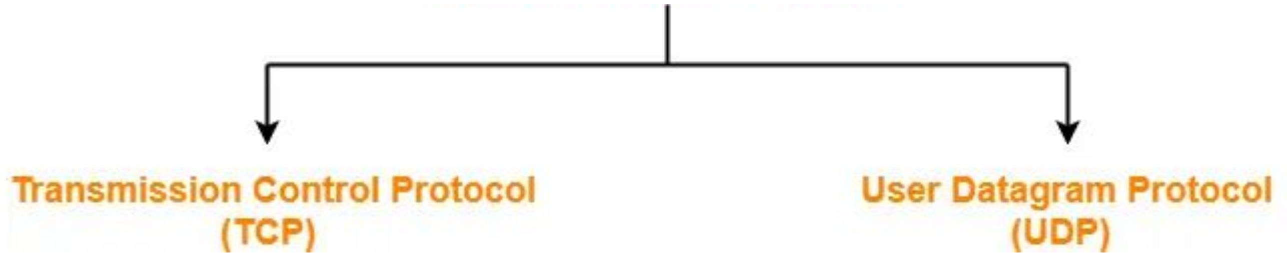
Figure 6-17. (a) Multiplexing. (b) Inverse multiplexing.

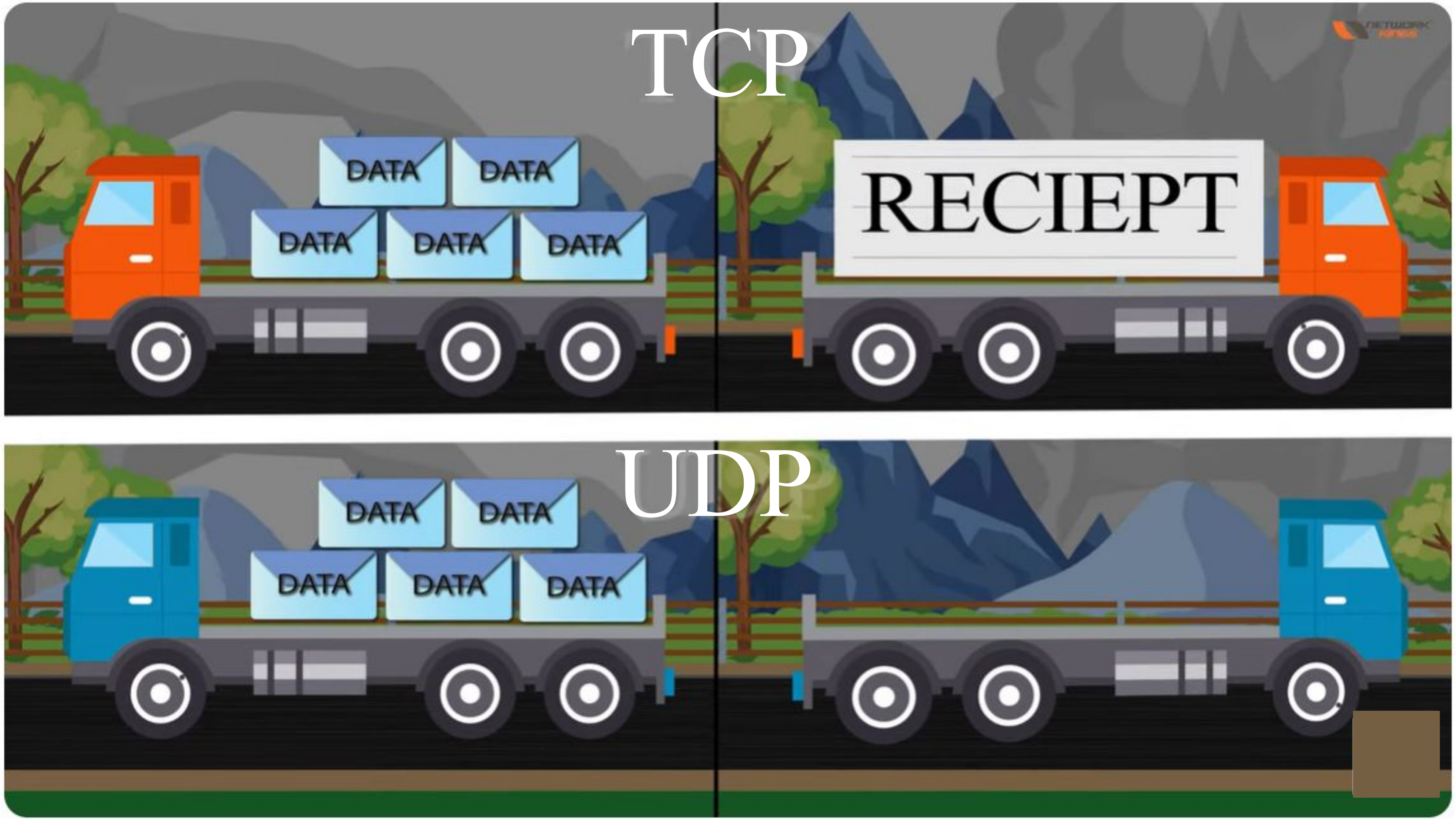
In the transport layer, the need for multiplexing can arise in a number of ways.

- **Case-I** : If only one network address is available on a host, all transport connections on that machine have to use it. When a segment comes in, some way is needed to tell which process to give it to. This situation, called multiplexing, is shown in Fig. 6-17(a).
- In this figure, four distinct transport connections all use the same network connection (e.g., IP address) to the remote host.

- **Case-II** : A host may have access to multiple network paths or links, each with different characteristics, such as bandwidth or reliability. **If a user requires more bandwidth** than a single network path can offer, a technique called inverse multiplexing can be employed.
- Inverse multiplexing involves distributing network traffic among multiple network paths on **a round-robin or load-balancing basis**, effectively combining the resources of these paths.
- Fig. 6-17(b) illustrates **inverse multiplexing**, where multiple network paths are used to enhance the overall performance.

Transport Layer Protocols




TCP
DATA
DATA
DATA
DATA
DATA
RECIEPT
UDP
DATA
DATA
DATA
DATA
DATA

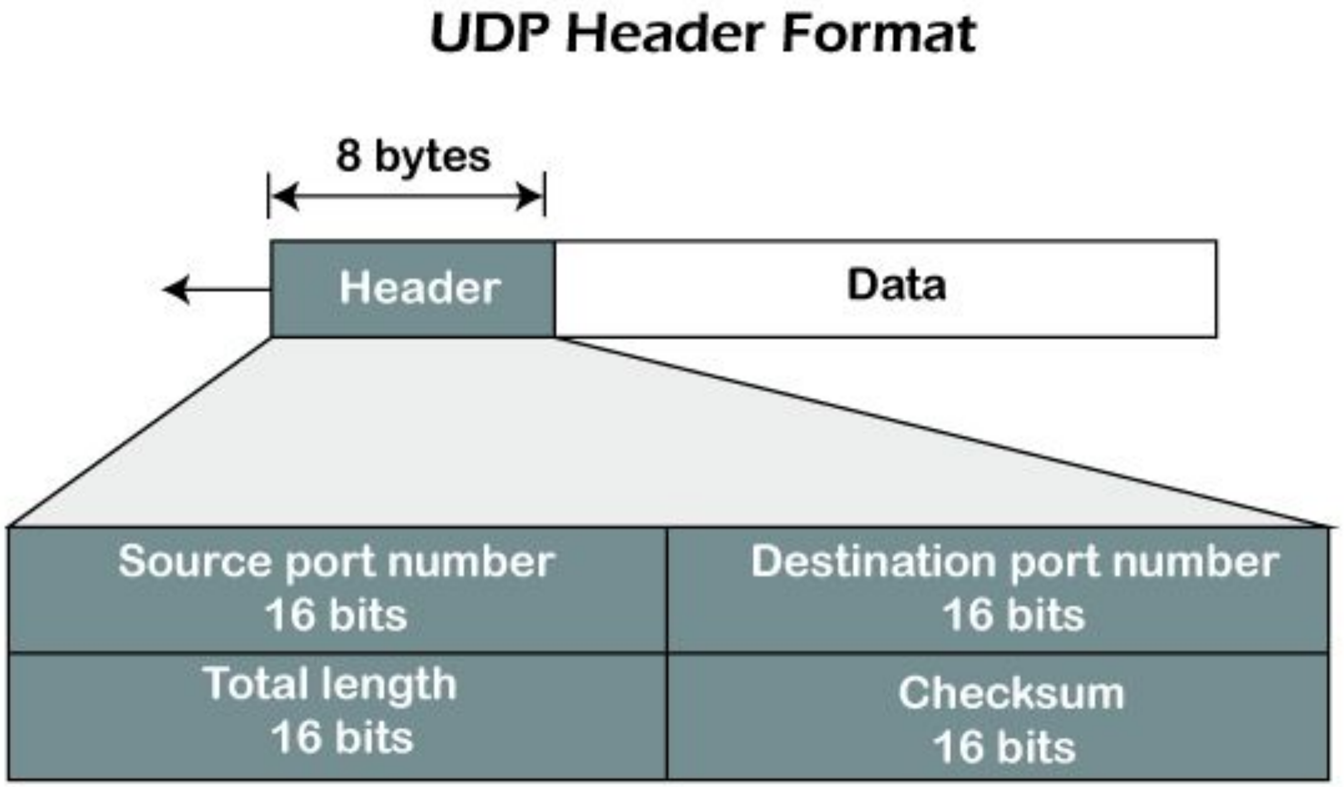
UDP: User Datagram Protocol

- UDP is a part of the **Internet Protocol suite**, referred to as UDP/IP suite.
- Both UDP and TCP are transport layers protocols which are used on the Internet or run on the top of the Internet Protocol (IP) and commonly known as **UDP/IP** and **TCP/IP**, respectively.
- Unlike TCP, it is **an unreliable** and **connectionless protocol**. So, there is **no need to establish a connection** prior to data transfer.
- Compare to TCP,the UDP is the **simplest transport layer protocol** designed to send data over the Internet.
- It picks the datagram from the network layer and attaches the header then forwards it to the user.

Characteristics of UDP:

- It is **a fast, unreliable, and stateless protocol** that makes it suitable for use with applications that can tolerate lost data.
- It can be used for transaction-based protocols, such as **DNS ,VoIP**(Voice over Internet Protocol) etc..
- UDP supports **broadcast and multicast communication**, allowing a single packet to be sent to multiple recipients simultaneously.
- For real-time services like **computer gaming, voice or video communication, live conferences we need UDP, Since high performance is needed.**
- There is **no error checking** in UDP.
- It is a connectionless protocol as **it doesn't need a virtual circuit before transferring the data**.

UDP Header Format:



IPv4 Pseudo Header

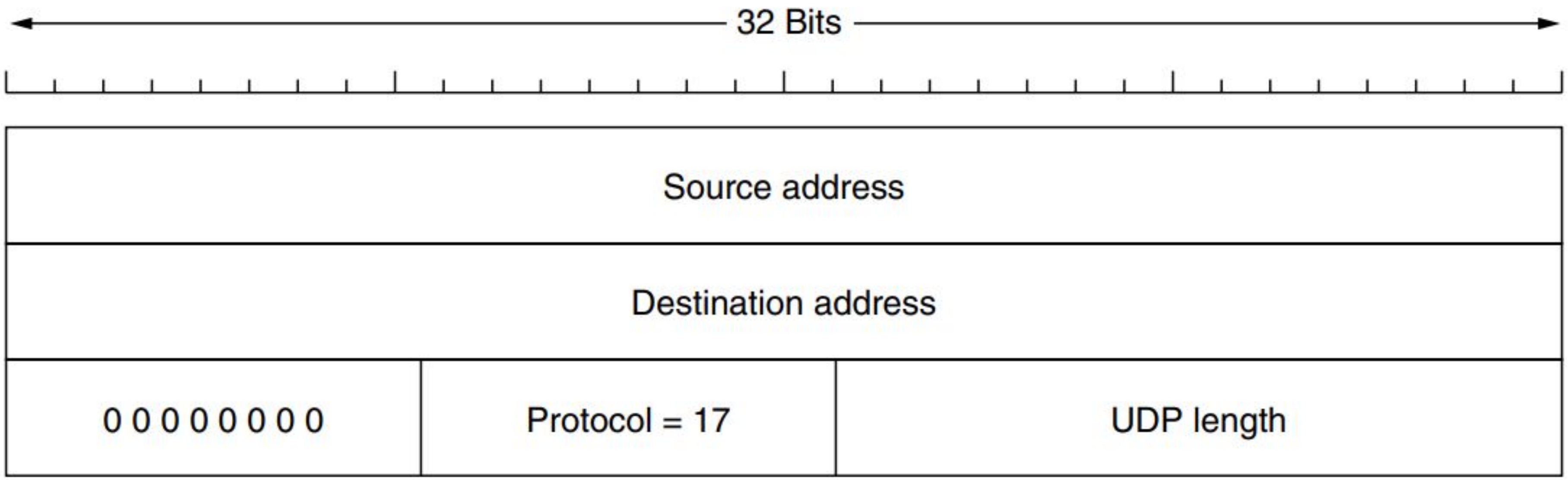


Figure 6-28. The IPv4 pseudoheader included in the UDP checksum.

UDP Header Format:

- **Source Port:** Source Port is a **2 Byte long field** used to identify the port number of the source.
- **Destination Port:** It is a **2 Byte long field**, used to identify the port number of the destination
- **Length:**Length is a **16 bit field/ 2 Bytes.**
  - It identifies the combined length of UDP Header and Encapsulated data.
  - Length = Length of UDP Header + Length of data
- **Checksum:** Checksum is 2 Bytes long field.

Checksum:

The UDP checksum is a 16-bit field that provides a form of error detection for the data contained in the UDP datagram. Here's how the UDP checksum field works:

- **Error Detection:** The primary purpose of the UDP checksum is to detect errors in the UDP datagram during transmission. It helps ensure the integrity of the data being sent.
- **Calculation**: The checksum is calculated based on **the contents of the UDP datagram, including the UDP header, UDP data, and a pseudo-header**. The pseudo-header includes the source and destination IP addresses, the protocol number (which is 17 for UDP), and the UDP length.
- **Algorithm**: The calculation of the checksum uses a simple mathematical algorithm called the one's complement sum. It involves summing up all 16-bit words in the data, taking the one's complement of the sum, and placing it in the checksum field.

- **Verification**: Upon receiving a UDP datagram, the recipient recalculates the checksum based on the received data, including the UDP header, data, and pseudo-header. It then compares the calculated checksum with the checksum value included in the received UDP header.
- **Discard on Error:** If the calculated checksum does not match the value in the checksum field of the received UDP datagram, the datagram is considered to have an error, and it may be discarded. This indicates that the data may have been corrupted during transmission.

TCP(Transmission Control Protocol)

- ★ **TCP (Transmission Control Protocol)** is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing **reliable delivery services**.
- ★ It is **a connection-oriented protocol** that means it establishes the connection prior to the communication that occurs between the computing devices in a network.
- ★ This protocol is used with an IP protocol, so together, they are referred to as a TCP/IP.

Features of TCP protocol:

1. **Reliable :**

   That is, the receiver always sends either **positive or negative acknowledgement** about the data packet to the sender, so that the sender always has bright clue about whether the data packet is reached the destination or it needs to resend it.

2. Order of the data is maintained :

TCP ensures that the data reaches intended destination in the same order it was sent.

3. Connection-oriented:

It is a connection-oriented service that means the data exchange occurs only after the connection establishment. When the data transfer is completed, then the connection will get terminated.

4. Full duplex:

In TCP data can be transmitted from receiver to the sender or vice versa at the same time.

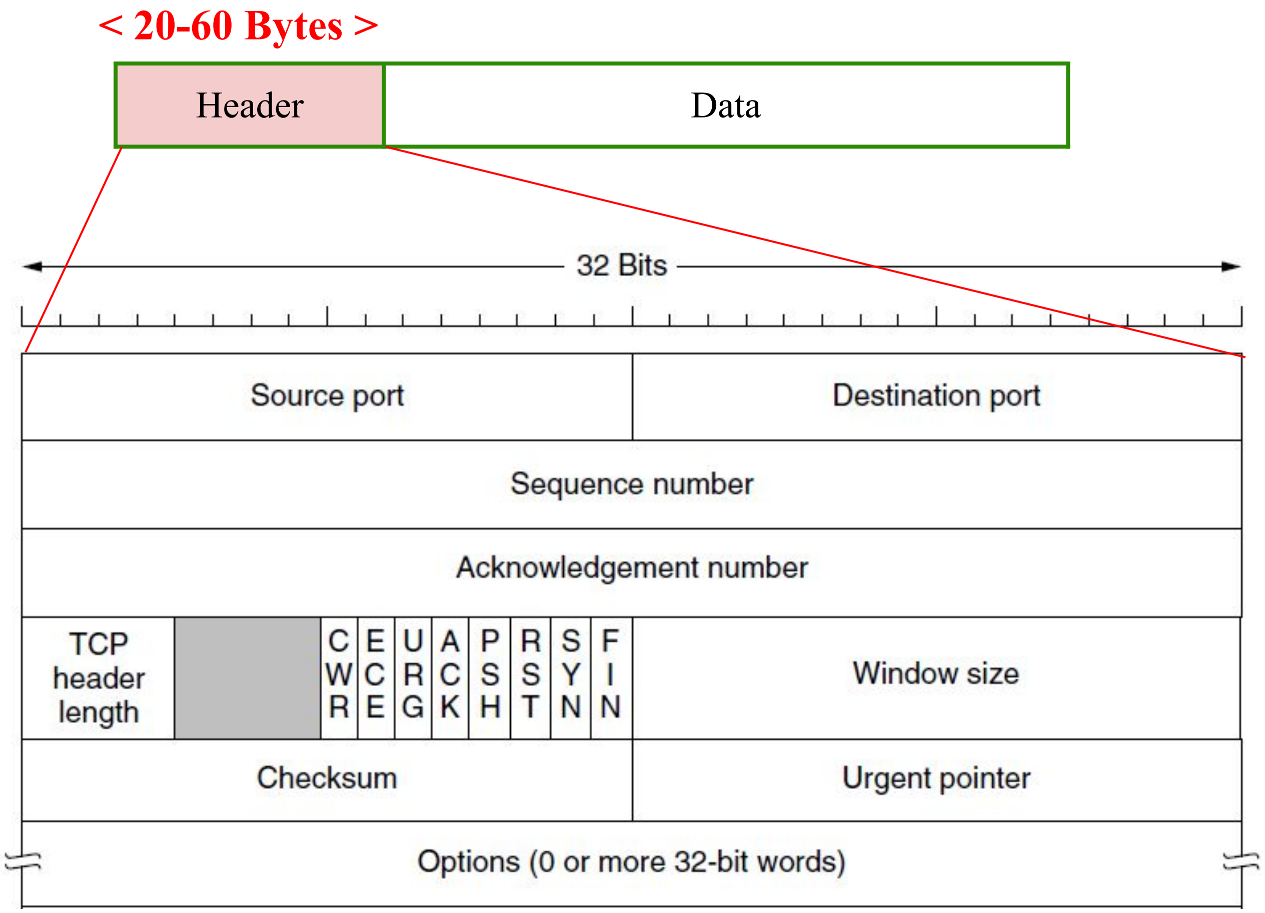
5.TCP provides **error-checking and recovery mechanism**.

6. TCP provides **congestion control.**

7. TCP provides **flow control** and quality of service.

The TCP Header Format

TCP Header Format :



1. Source Port-

- Source Port is a 16 bit field.
- It identifies the port of the sending application.

2. Destination Port-

- Destination Port is a 16 bit field.
- It identifies the port of the receiving application.

3. Sequence Number-

- Sequence number is a 32 bit field.
- TCP assigns a unique sequence number to each byte of data contained in the TCP segment.
- This field contains the sequence number of the first data byte.
- It is used to reassemble the message at the receiving end of the segments that are received out of order.

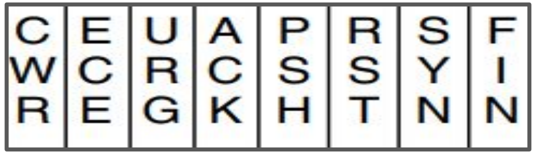
4. Acknowledgement Number-

- Acknowledgment number is a 32 bit field.
- It contains sequence number of the data byte that receiver **expects to receive next from the sender.**
- **It is always sequence number of the last received data byte incremented by 1**.

5.TCP Header Length:

- Header length is a 4 bit field.
- This is a 4-bit field that indicates the length of the TCP header by **a number of 4-byte words** in the header, i.e if the header is 20 bytes(min length of TCP header), then this field will hold 5 (because 5 x 4 = 20) and the maximum length: 60 bytes, then it'll hold the value 15(because 15 x 4 = 60). Hence, the value of this field is always between 5 and 15.

Eight 1-bit flags:(control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc)



1. **ECE(Explicit Congestion Notification)**

- ECE is set to signal an ECN-Echo to a TCP sender to tell it to slow down when the TCP receiver gets a congestion indication from the network.

2. **CWR(Congestion Window Reduced)**

- CWR is set to signal Congestion Window Reduced from the TCP sender to the TCP receiver so that it knows the sender has slowed down and can stop sending the ECN-Echo.

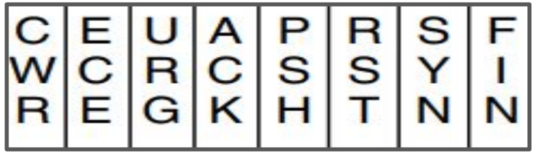
3. **URG(Urgent pointer):**

- URG is set to 1 if the Urgent pointer is in use.
- The Urgent pointer is used to indicate a byte offset from the current sequence number at which urgent data are to be found.

4. ACK(Acknowledgement number)

- The ACK bit is set to 1 to indicate that the Acknowledgement number is valid.

5. PSH(Push)



When PSH bit is set to 1,

- All the segments in the buffer are immediately pushed to the receiving application.
- No wait is done for filling the entire buffer.
- This makes the entire buffer to free up immediately.

6. RST(Reset): RST bit is used to reset the TCP connection.

7. SYN : It synchronizes the sequence number. It's the first bit

8.FIN : FIN bit is used to terminate the TCP connection.

Window Size-

- Window size is a 16 bit field.
- It contains the size of the receiving window of the sender.
- It advertises how much data (in bytes) the sender can receive without acknowledgement.
- Thus, window size is used for Flow Control.

Checksum-

- Checksum is a 16 bit field used for error control.
- It verifies the integrity of data in the TCP payload.
- Sender adds CRC checksum to the checksum field before sending the data.
- Receiver rejects the data that fails the CRC check.

Urgent pointer –

- This field (valid only if the URG control flag is set) is used to point to data that is urgently required that needs to reach the receiving process at the earliest. The value of this field is added to the sequence number to get the byte number of the last urgent byte.

Checksum-

- Checksum is a 16 bit field used for error control.
- It verifies the integrity of data in the TCP payload.
- Sender adds CRC checksum to the checksum field before sending the data.
- Receiver rejects the data that fails the CRC check.

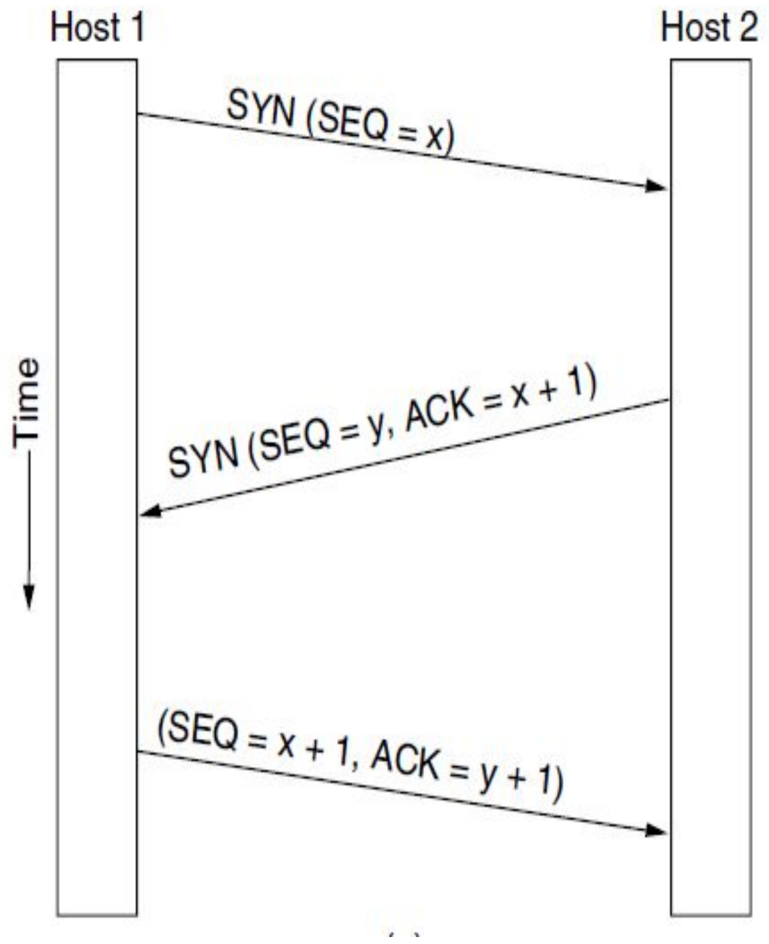
Options - Option field is always described in 32-bit words.

- It facilitates additional options which are not covered by the regular header
- Options field is generally used for the following purposes-
  - Time stamp
  - Window size extension

TCP Connection Establishment

- To make the transport services reliable, TCP hosts must establish a connection-oriented session with one another.
- Connection establishment is performed by using t**he three-way handshake mechanism.**
- A three-way handshake synchronizes both ends of a network by enabling both sides to agree upon original sequence numbers.
- This mechanism also provides that both sides are ready to transmit data and learn that the other side is available to communicate.
- This is essential so that packets are not shared or retransmitted during session establishment or after session termination.
- Each host randomly selects a sequence number used to track bytes within the stream it is sending and receiving.

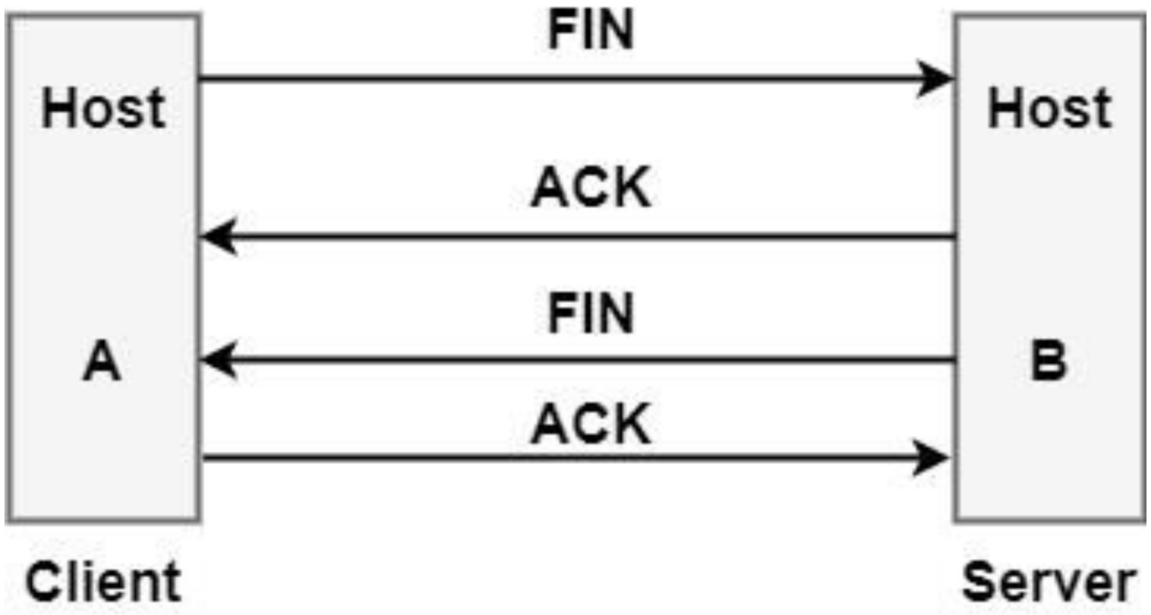
- The first device sends a SYN (synchronize) packet to the second device, indicating its intent to establish a connection.
- The second device receives the SYN packet and sends a SYN-ACK (synchronize-acknowledge) packet back to the first device, indicating its agreement to establish a connection.
- The first device receives the SYN-ACK packet and sends an ACK (acknowledge) packet back to the second device, indicating that it has received the second device's response and is ready to communicate.
- Once the **three-way handshake** is complete, the devices are connected, and data can be exchanged.



TCP connection establishment

TCP Connection Release

- While it creates three segments to establish a connection, it takes **four segments to terminate a connection.**
- During a TCP connection is full-duplex (that is, data flows in each direction independently of the other direction), each direction should be shut down alone.
- The termination procedure for each host is shown in the figure. The rule is that either end can share a FIN when it has finished sending data.
- When a TCP receives a FIN, it should notify the application that the other end has terminated that data flow direction.
- The sending of a FIN is usually the result of the application issuing a close.
- The receipt of a FIN only means that there will be no more data flowing in that direction. A TCP can send data after receiving a FIN.
- The end that first issues the close (example, send the first FIN) executes the active close. The other end (that receives this FIN) manages the passive close.



TCP Termination