Transport Layer

JAIPUR (H1)

Elder Tripathi Ji House

Jevan	Mamta	Sameer
Arjun	Shweta	Divya

MUMBAI (H2)

Younger Tripathi Ji House

Umesh	Anju	Kriti
Arun	Karishma	Tushar

LNMIIT
The LNM Institute of Information Technology

- Every month, each member writes a letter to each of the members of the other house
- Each house sends 36 letters to the other house every month
- Each letter is delivered by traditional postal service in a separate envelope
- Every month, Arjun visits all the house members (H1), collects the letter and give it to postal service guy who visits H1 every month
- Similarly, when the letters arrive from house (H2), Arjun has a job to distribute the letters among all the family members
- Karishma has a similar job in house 2

Transport Layer

- The postal service provides logical communication between the two houses. The postal service moves letters from house to house, not person to person.
- Arjun and Karishma provide logical communication among the family members
- Arjun and Karishma pick up letters and deliver them to their family members
- From a family member's perspective, Arjun and Karishma provide the letters-delivering services
- Arjun and Karishma are only a part (the end-system part) of the end-to-end delivery process
- This household example is an excellent analogy for explaining how the transport layer relates to the network layer

application messages = letters in envelopes
processes = family members
hosts (end systems) = houses
transport-layer protocol = arjun and karishma
network-layer protocol = postal service

Transport Layer

- Application processes use the logical communication provided by the transport layer to send messages to each other

- Transport layer protocols are implemented in end systems and not in network routers

- The transport layer converts the application-layer messages it receives from a sending application process into transport-layer packets, known as transport-layer ***segments***

- The transport layer then passes the segment to the network layer at the sending end system, where the segment is encapsulated within a network-layer packet (a ***datagram***) and sent to the destination

- The services that a transport protocol can provide are often constrained by the service model of the underlying network-layer protocol

- If the network-layer protocol cannot provide delay or bandwidth guarantees for transport-layer segments sent between hosts, and then the transport-layer protocol cannot provide delay or bandwidth guarantees for application messages sent between processes

Transport Layer

- **Transport layer is responsible for *process-to-process* delivery**

- **At the transport layer, we need a transport layer address, called a *port number*, to choose among multiple processes running on the destination host. The destination port number is necessary for delivery; the source port number is required for the reply**

- **The client program defines itself with a port number chosen randomly by the transport layer software running on the client host. Servers port number cannot be chosen randomly**

lANA Ranges

The lANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well known, registered, and dynamic (or private), as shown in Figure 23.4.

- Well-known ports. The ports ranging from 0 to 1023 are assigned and controlled by lANA. These are the well-known ports.
- Registered ports. The ports ranging from 1024 to 49,151 are not assigned or controlled by lANA. They can only be registered with lANA to prevent duplication.
- Dynamic ports. The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process. These are the ephemeral ports.

Transport Layer

- **Socket address:- Combination of an IP address and Port number**

- **Encapsulation and Decapsulation**

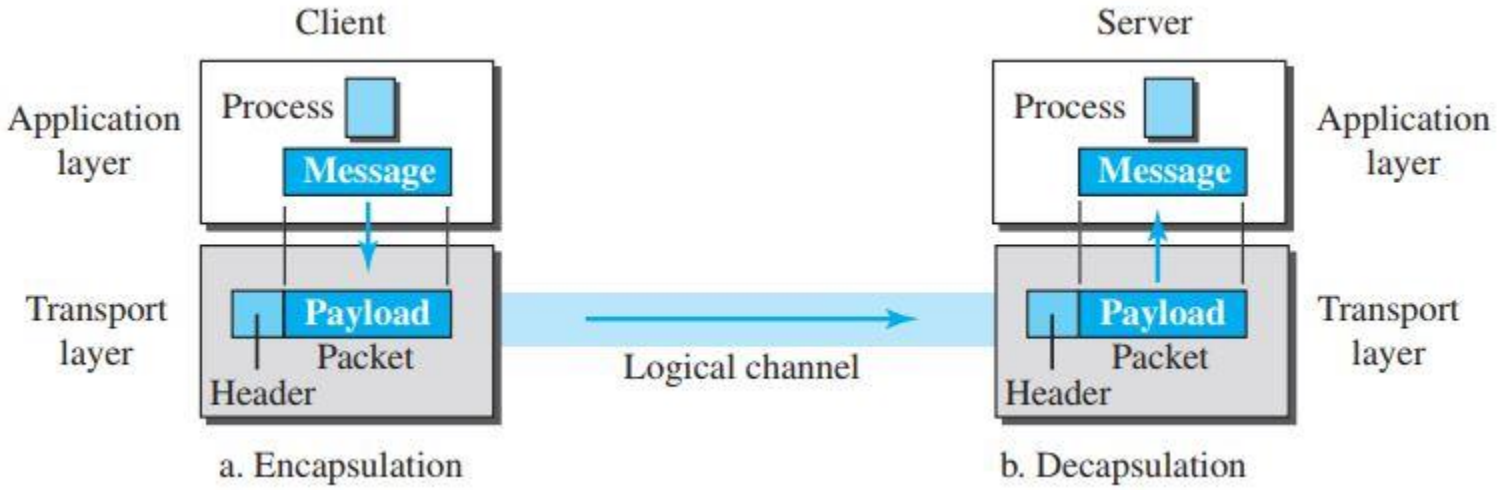

Fig 1. Encapsulation and Decapsulation

- **Multiplexing**

- **Demultiplexing**

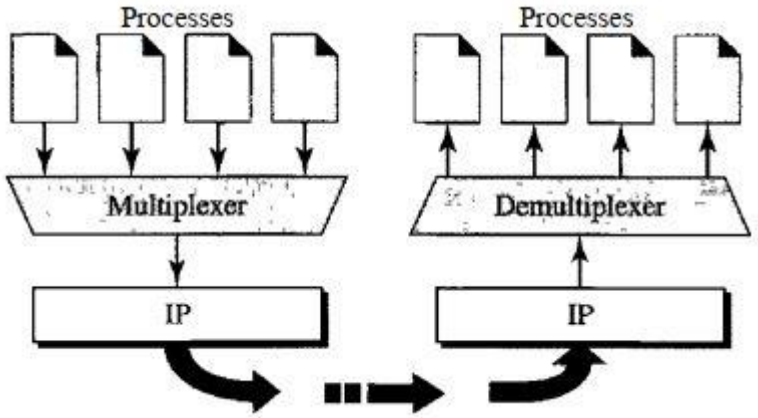

Fig 2. Multiplexing and Demultiplexing

Transport Layer

Port	Protocol	UDP	TCP	SCTP	Description
7	Echo	√	√	√	Echoes back a received datagram
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
21	FTP-21		√	√	File Transfer Protocol
23	TELNET		√	√	Terminal Network
25	SMTP		√	√	Simple Mail Transfer Protocol
53	DNS	√	√	√	Domain Name Service
67	DHCP	√	√	√	Dynamic Host Configuration Protocol
69	TFTP	√	√	√	Trivial File Transfer Protocol
80	HTTP		√	√	HyperText Transfer Protocol
111	RPC	√	√	√	Remote Procedure Call
123	NTP	√	√	√	Network Time Protocol
161	SNMP-server	√			Simple Network Management Protocol
162	SNMP-client	√			Simple Network Management Protocol

Fig 3. Well known port numbers

User Datagram Protocol (UDP)

- **UDP is connectionless and unreliable protocol**
- **UDP is simple protocol using minimum overhead**

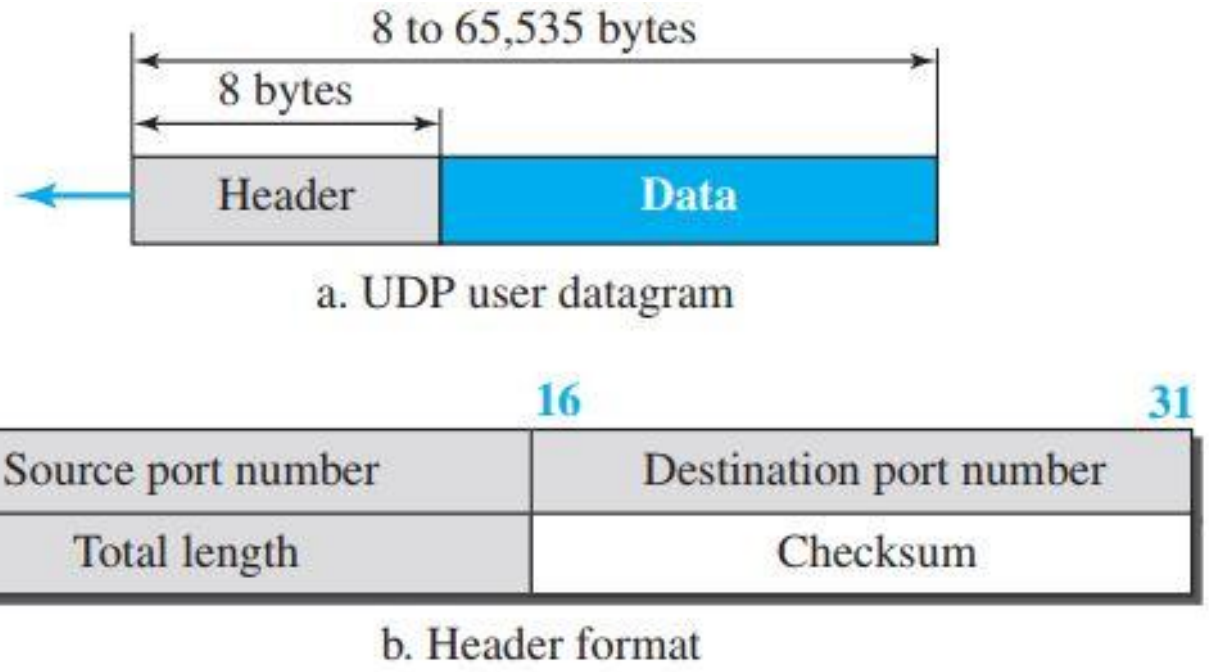

Fig 4. UDP packet format

User Datagram Protocol (UDP)

The following is the content of a UDP header in hexadecimal format.

CB84000D001C001C

a. What is the source port number?

b. What is the destination port number?

c. What is the total length of the user datagram?

d. What is the length of the data?

e. Is the packet directed from a client to a server or vice versa?

f. What is the client process?

Solution

a. The source port number is the first four hexadecimal digits $(CB84)_{16}$, which means that the source port number is 52100.

b. The destination port number is the second four hexadecimal digits $(000D)_{16}$, which means that the destination port number is 13.

c. The third four hexadecimal digits $(001C)_{16}$ define the length of the whole UDP packet as 28 bytes.

d. The length of the data is the length of the whole packet minus the length of the header, or $28 - 8 = 20$ bytes.

e. Since the destination port number is 13 (well-known port), the packet is from the client to the server.

f. The client process is the Daytime (see Table 24.1).

Fig 5. Example

UDP Services

- **Connectionless services**
  - UDP is an independent datagram
  - No connection establishment, no connection termination
  - Processes sending short messages, messages less than 65,507 bytes (65,535 minus 8 bytes for the UDP header and minus 20 bytes for the IP header), can use UDP

- **Flow control**
  - No flow control and hence no window mechanism
  - Receiver may overflow with incoming messages

- **Error control**
  - No error control mechanism except for checksum
  - When error is detected using checksum, the UDP datagram is discarded

UDP Services

- **Congestion control**
  - UDP is connectionless and hence doesn't provide congestion control

- **Encapsulation and Decapsulation**

- **Queueing**