UDP and TCP

Dr. G. Omprakash

Assistant Professor, ECE, KLEF

Transport-Layer protocols

- UDP: User Datagram Protocol
  - Unreliable connectionless transport-layer protocol
- TCP: Transmission Control Protocol
  - Reliable connection-oriented protocol
- SCTP: Stream Control Transmission Protocol
  - New transport-layer protocol: combines the features of UDP and TCP

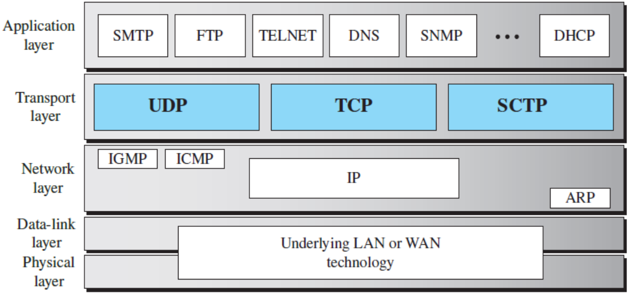
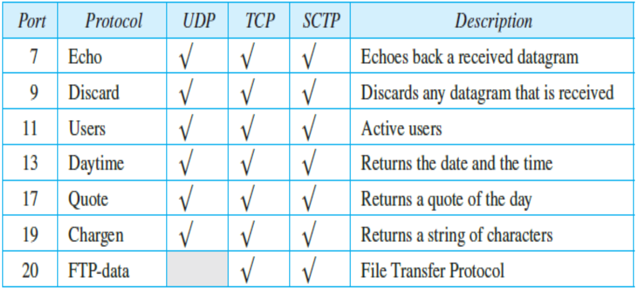


Figure: Position of transport-layer protocols

Port Numbers



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

Figure: Commonly used ports used with the three transport protocols

User Datagram Protocol

User Datagram Protocol

- It is a connectionless, unreliable transport protocol
- Provides process-to-process communication
- Advantages:
  - Simple protocol using a minimum of overhead
- If a process wants to send a small message without caring much about reliability, it can use UDP
- UDP packets $\Longrightarrow$ called user datagrams

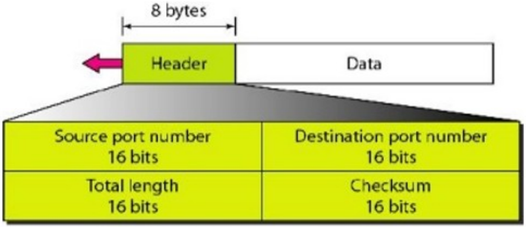


Figure: User datagram packet format

UDP Services

- **Process-to-Process Communication**
    - UDP provides this service using socket addresses
        - socket addresses: combination of IP addresses and port numbers.
- **Connectionless Services**:
    - Each user datagram sent by UDP is an independent datagram
    - No relationship between the different user datagrams (coming from the same source)
    - user datagrams are not numbered
    - There is no connection establishment and no connection termination (as in TCP)
- There is no flow control
    - Process using UDP should provide for this service
- There is no error-control (Except for optional checksum)
- It does not provide congestion control (connectionless protocol)
- UDP encapsulates and decapsulates messages

UDP Usecase

- A client/server application such as **DNS** uses the services of UDP
  - Send a short request to a server and to receive a quick response from it.
  - The request and response can each fit in one user datagram.
- Used in real-time interactive application, such as Skype.
  - Audio and video are divided into frames and sent one after another.
  - Each small part of the screen is sent using one single user datagram
  - Receiving UDP can easily ignore the corrupted or lost packet
    - Viewers do not even notice the part of the screen which is blank for a very short period of time.
- A client/server application such as **SMTP**-used in e-mail cannot use the services of UDP
  - Long e-mail message (video,image) doesn't fit in one user datagram

UDP Uses

- UDP is suitable for a process that requires simple request-response communication
- UDP is a suitable transport protocol for multicasting
  - Multicasting possible only in UDP but not in TCP
- UDP is used for some route updating protocols such as Routing Information Protocol (RIP)
- UDP is used for management processes such as SNMP (Simple Network Management Protocol)

Transmission Control Protocol (TCP)

Transmission Control Protocol

- TCP is a connection-oriented, reliable protocol
- TCP explicitly defines
  - Connection establishment
  - Data transfer
  - Connection teardown phases
- To provide reliability
  - TCP uses a combination of the Go-Back-N and Selective-Repeat protocols
  - Uses checksum for error detection
  - Uses retransmission of lost or corrupted packets, selective acknowledgments

TCP Services

- **Process-to-Process Communication**
  - TCP provides process-to-process communication using port numbers
- **Stream Delivery Service**
  - TCP allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes
- **Sending and Receiving Buffers**
  - Why TCP needs Buffer?
    - Sending and the receiving processes may not necessarily write or read data at the same rate

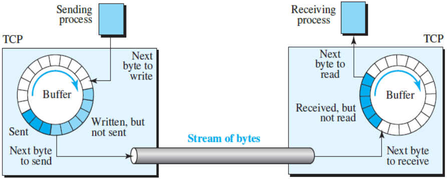


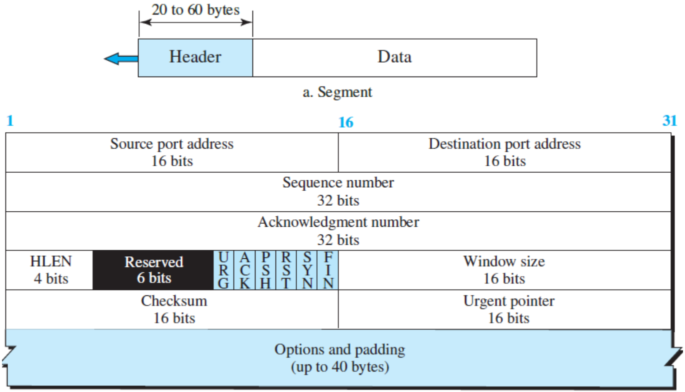
Figure: Sending and receiving buffers

- Normally the buffers are hundreds or thousands of bytes
- Buffer size can be different on both end

TCP Services

- **Segments**
  - TCP groups a number of bytes together into a packet called a *segment*
  - TCP adds a header to each segment and delivers the segment to the network layer for transmission
  - The segments are encapsulated in an IP datagram and transmitted.
  - Segments may be received out of order, lost, or corrupted and re-sent
    - segments are not necessarily all the same size
- **Full-Duplex Communication**
  - *Full-Duplex service*: Data can flow in both directions at the same time
- **Multiplexing and Demultiplexing**
  - TCP performs multiplexing at the sender and demultiplexing at the receiver.
- **Connection-Oriented Service**: Three phases occur
  - The two TCP's establish a logical connection between them
  - Data are exchanged in both directions.
  - The connection is terminated.

TCP Segment



20 to 60 bytes

Header

Data

a. Segment

1

16

31

Source port address
16 bits

Destination port address
16 bits

Sequence number
32 bits

Acknowledgment number
32 bits

HLEN
4 bits

Reserved
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)

b. Header

Figure: TCP segment format

Header Fields

- **Source port address**: 16-bit field that defines the port number of the application program (process) in the source host
- **Destination port address**:16-bit field that defines the port number of the application program in the receiving host.
- **Sequence number**: 32-bit field defines the number assigned to the first byte of data contained in this segment
    - During connection establishment each party uses a random number generator to create an **initial sequence number (ISN)**
- **Acknowledgment number**: 32-bit field defines the byte number that the receiver of the segment is expecting
    - x received $\implies$ $x + 1$ is the acknowledgment number
- **Header Length**: 4-bit field indicates the number of 4-byte words in the TCP header. (value=5 for 20 bytes, 15 for 60 bytes)

Header Fields

- **Control:**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
- **Window Size**: Defines the window size of the sending TCP in bytes
  - 16 bits $\Longrightarrow$ maximum size of the window is 65,535 bytes
- **Checksum**: 16-bit field contains the checksum.
- **Urgent Pointer**: valid only if the urgent flag is set
  - used when the segment contains urgent data

TCP Connection

TCP is connection-oriented transport protocol which establishes a logical path between the source and destination. This connection-oriented transmission requires three phases

- **Connection Establishment**: Sending and Receiving party must initialize communication and get approval from the other party before any data are transferred.
  - Server program tells its TCP that it is ready to accept a connection: *passive open*
  - Client program issues a request to connect to a particular server : *active open*
  - TCP can now start the **three-way Handshaking**

Three-Way Handshaking

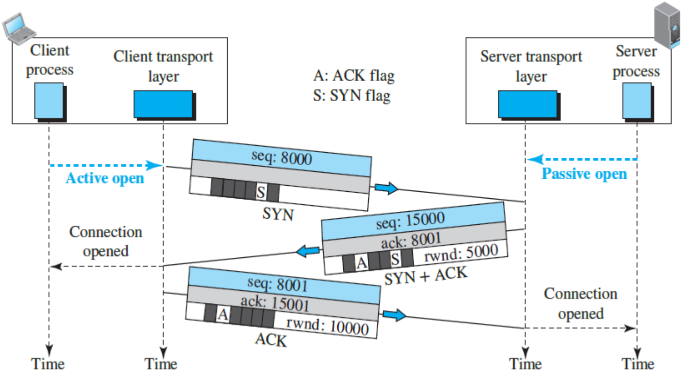


Figure: Connection establishment using three-way handshaking

Three-Way Handshaking

- **First: SYN (Synchronize)**
  - Client sends the first segment-SYN (SYN flag is set)
    - Segment is for synchronization of sequence numbers
    - Random number is chosen and sent to server: Initial Sequence Number (ISN)
    - Segment consumes one sequence number but doesn't carry data
- **Second: SYN-ACK (Synchronize-Acknowledge)**
  - Server sends the second segment (SYN + ACK flags are set)
  - The server uses this segment to initialize a sequence number for numbering the bytes sent from the server to the client
  - ACK flag is set and displaying the next sequence number it expects to receive
  - window size is defined (used by the client)
- **Third: ACK (Acknowledge)**
  - client sends the third segment (only ACK)
  - It acknowledges the receipt of the second segment
    - ACK flag=1, acknowledgment number field is sent
  - If an ACK segment does not carry any data, it does not consume any sequence numbers.

TCP Connection release

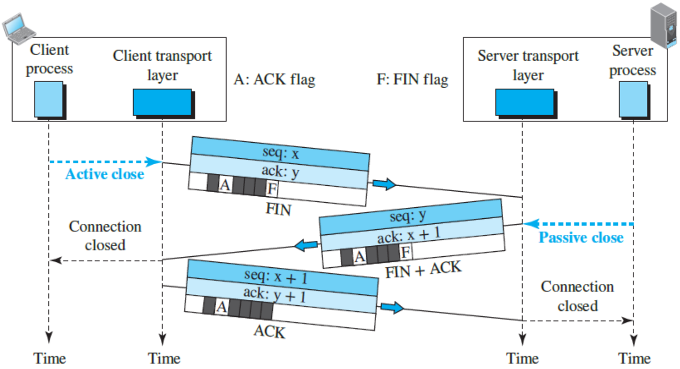


Figure: Connection termination using three-way handshaking

Connection Termination

- **First: FIN (Initiate Termination)**
  - Client TCP send the FIN segment with FIN flag set as 1
  - FIN segment can include the last chunk of data sent by the client
- **Second: FIN-ACK (Terminate-Acknowledge)**
  - After receiving FIN segment, server TCP informs its process and send FIN + ACK segment
  - Server TCP is announcing the closing of the connection in the other direction
  - This segment can also contain the last chunk of data from the server.
- **Third: ACK (Acknowledge)**
  - Client TCP sends the last segment, an ACK segment
  - Confirms the receipt of the FIN segment from the TCP server.
  - This segment cannot carry data

Acknowledge various sources for the images.

Thankyou