IT 4505
Section 3.1

Transport Layer Protocols

Transport Layer protocols

Introduction

- These are designed to allow peer entities on the source and destination hosts to carry on a conversation.

- Two end-to-end transport layer protocols have been defined here.

TCP(Transmission Control Protocol)

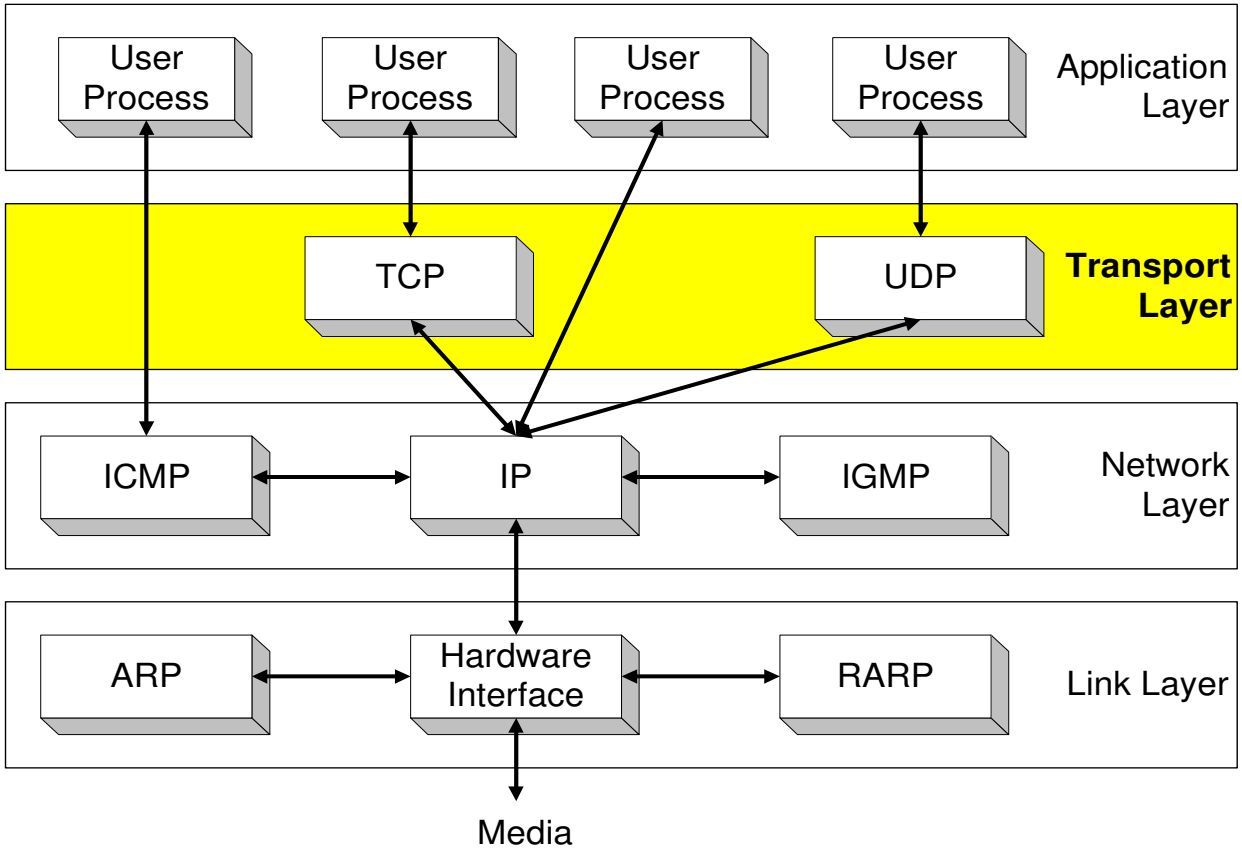
reliable, connection-oriented protocol.

UDP (User Datagram Protocol

unreliable, connectionless protocol

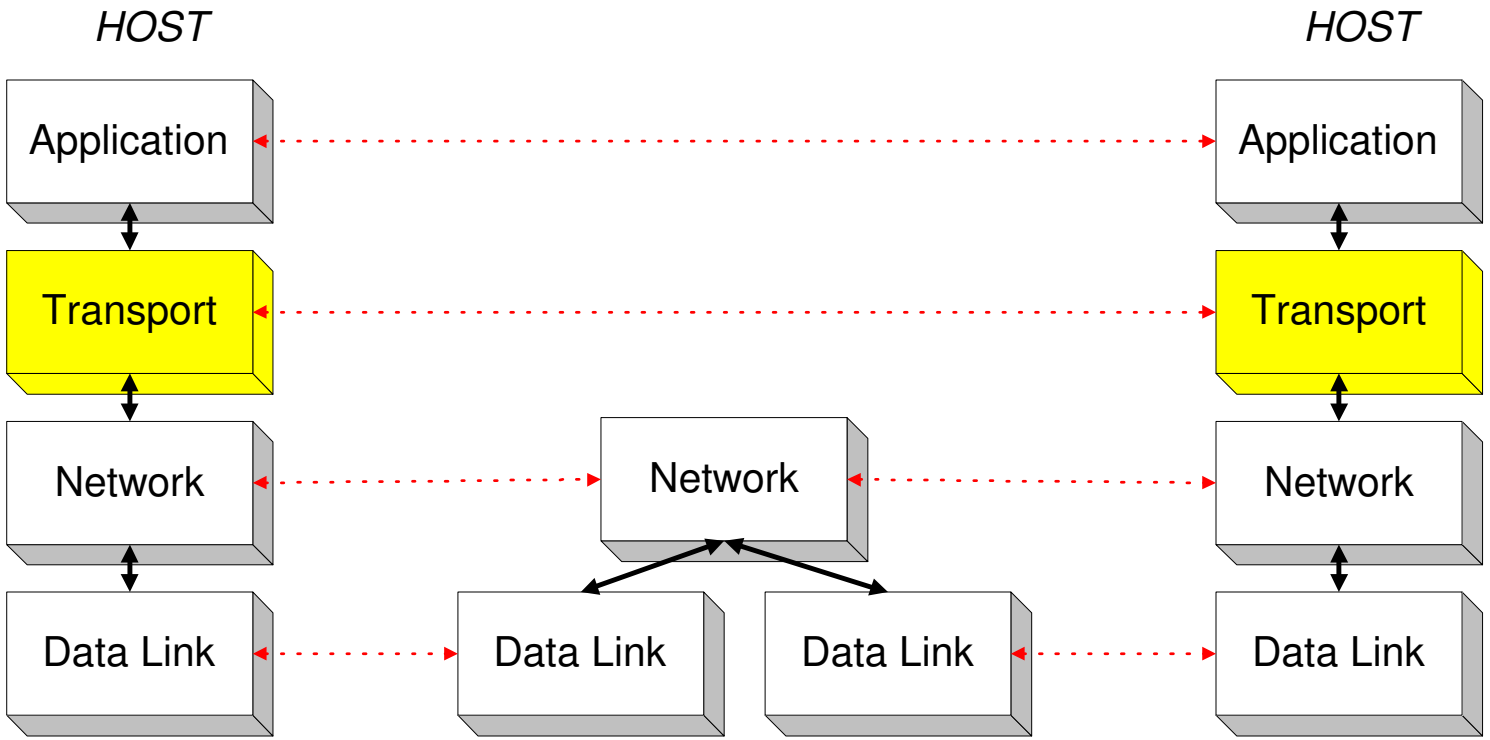
Orientation

- We move one layer up and look at the transport layer.



Orientation Cont.

- Transport layer protocols are end-to-end protocols
- They are only implemented at the hosts



Transport Protocols in the Internet

- The Internet supports 2 transport protocols

UDP - User Datagram Protocol

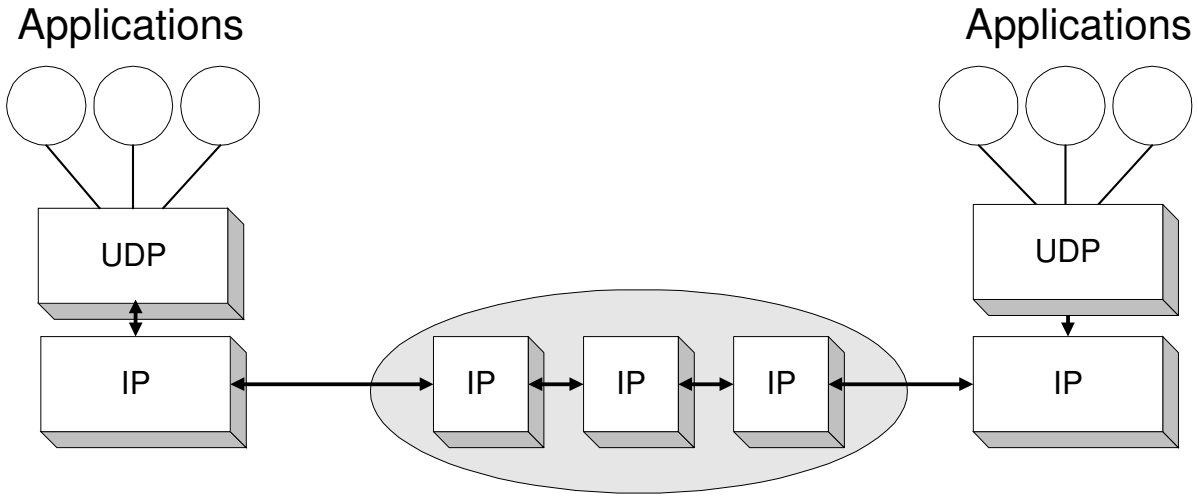
- datagram oriented
- unreliable, connectionless
- simple
- unicast and multicast
- useful only for few applications, e.g., multimedia applications
- used a lot for services
  - network management (SNMP), routing (RIP), naming (DNS), etc.

TCP - Transmission Control Protocol

- stream oriented
- reliable, connection-oriented
- complex
- only unicast
- used for most Internet applications:
  - web (http), email (smtp), file transfer (ftp), terminal (telnet), etc.

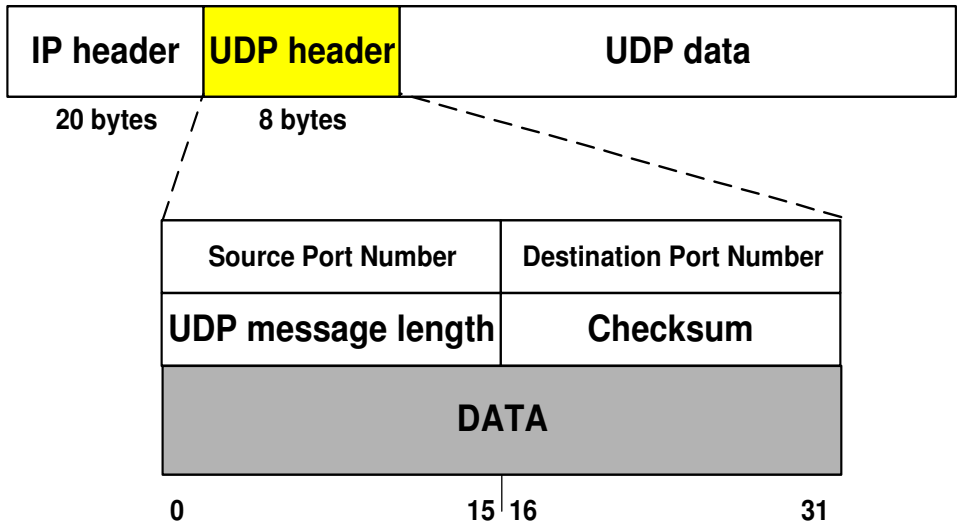
3.2.1 UDP - User Datagram Protocol

- UDP supports unreliable transmissions of datagrams
- UDP merely extends the host-to-to-host delivery service of IP datagram to an application-to-application service
- The only thing that UDP adds is multiplexing and demultiplexing



UDP Cont.

UDP Format

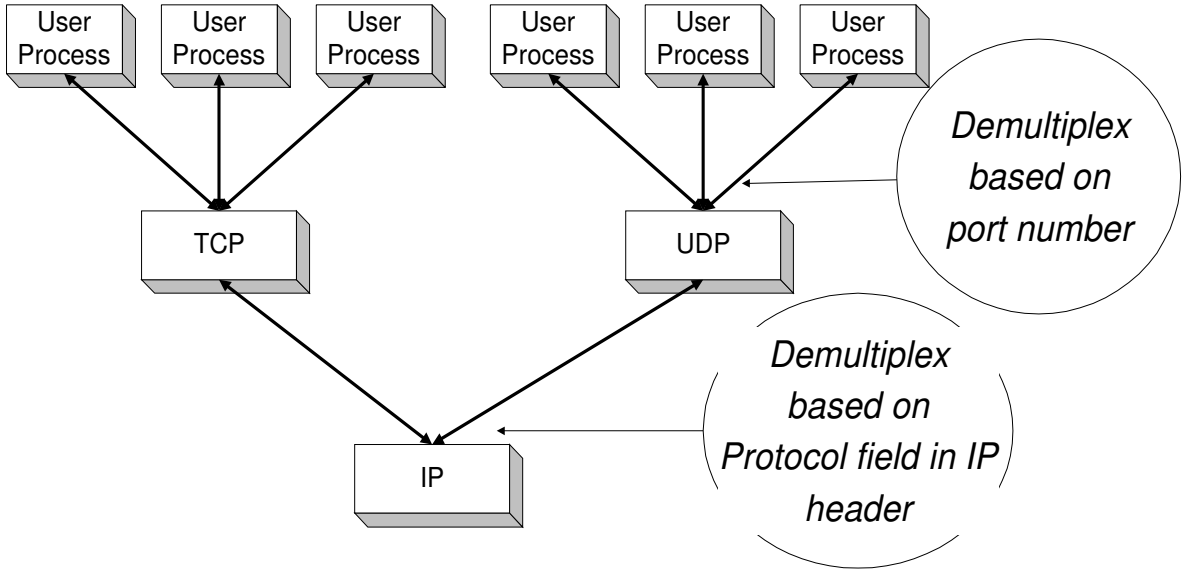


- Port numbers identify sending and receiving applications (processes). Maximum port number is 2^{16}-1= 65,535
- Message Length is at least 8 bytes (I.e., Data field can be empty) and at most 65,535
- Checksum is for header (of UDP and some of the IP header fields)

UDP Cont.

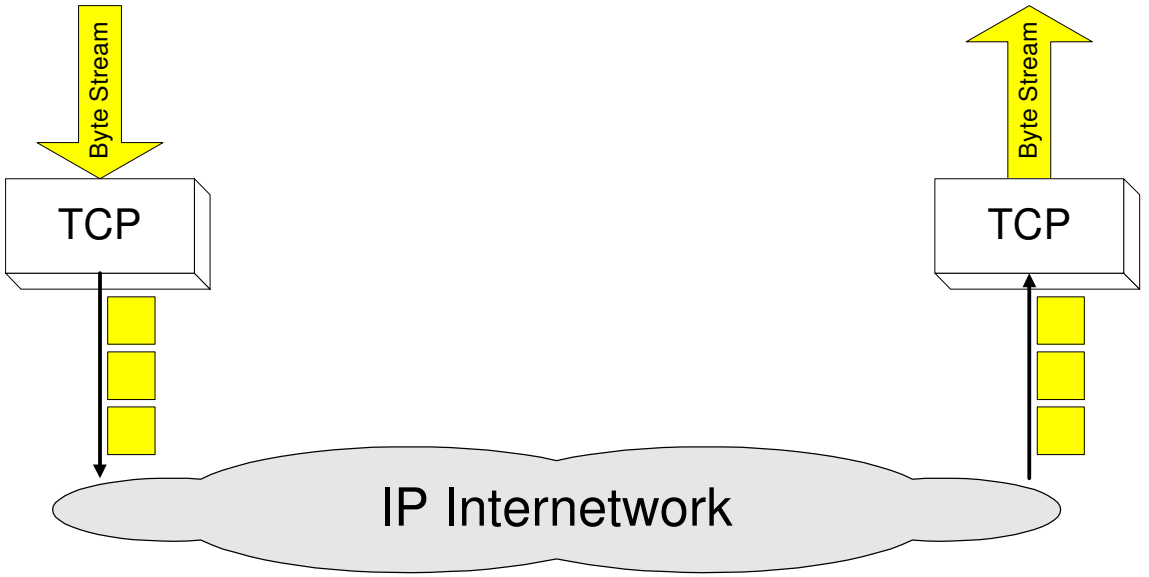
Port Numbers

- UDP (and TCP) use port numbers to identify applications
- A globally unique address at the transport layer (for both UDP and TCP) is a tuple **<IP address, port number>**
- There are 65,535 UDP ports per host.



3.2.2 TCP - Transmission Control Protocol

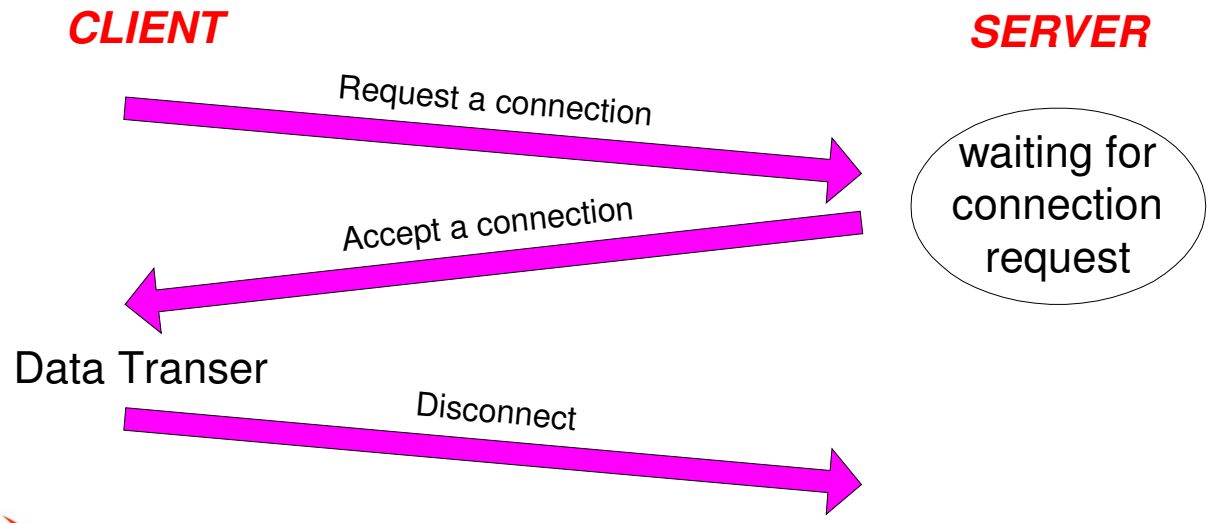
- Connection-oriented protocol
- Provides a reliable unicast end-to-end byte stream over an unreliable internetwork.



TCP Cont.

Connection-Oriented

- Before any data transfer, TCP establishes a **connection**:
  - One TCP entity is waiting for a connection ("**server**")
  - The other TCP entity ("**client**") contacts the server
- The actual procedure for setting up connections is more complex.
- Each connection is full duplex



TCP Cont.

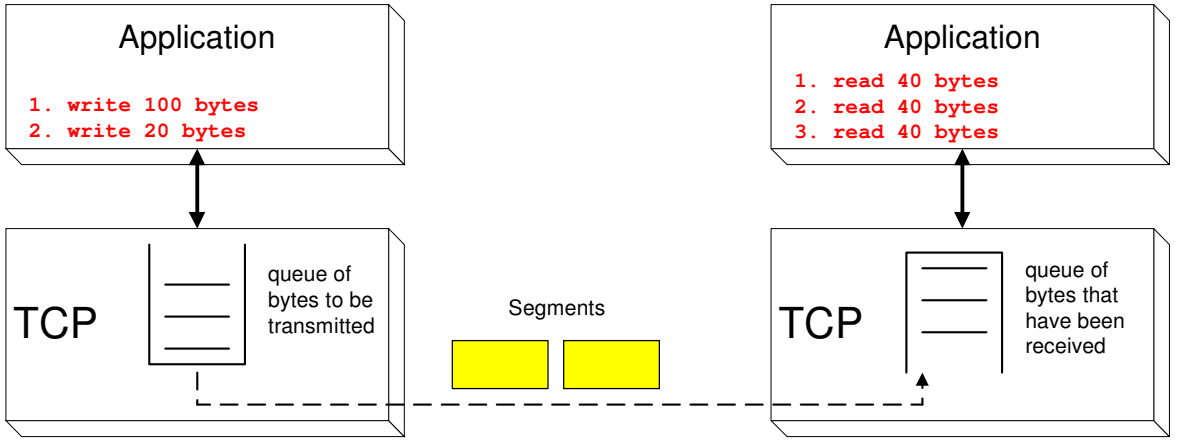
Reliable

- Byte stream is broken up into chunks which are called **segments**
  - Receiver sends acknowledgements (ACKs) for segments
  - TCP maintains a timer. If an ACK is not received in time, the segment is retransmitted
- **Detecting errors:**
  - TCP has checksums for header and data. Segments with invalid checksums are discarded
  - Each byte that is transmitted has a sequence number

TCP Cont.

Byte Stream Service

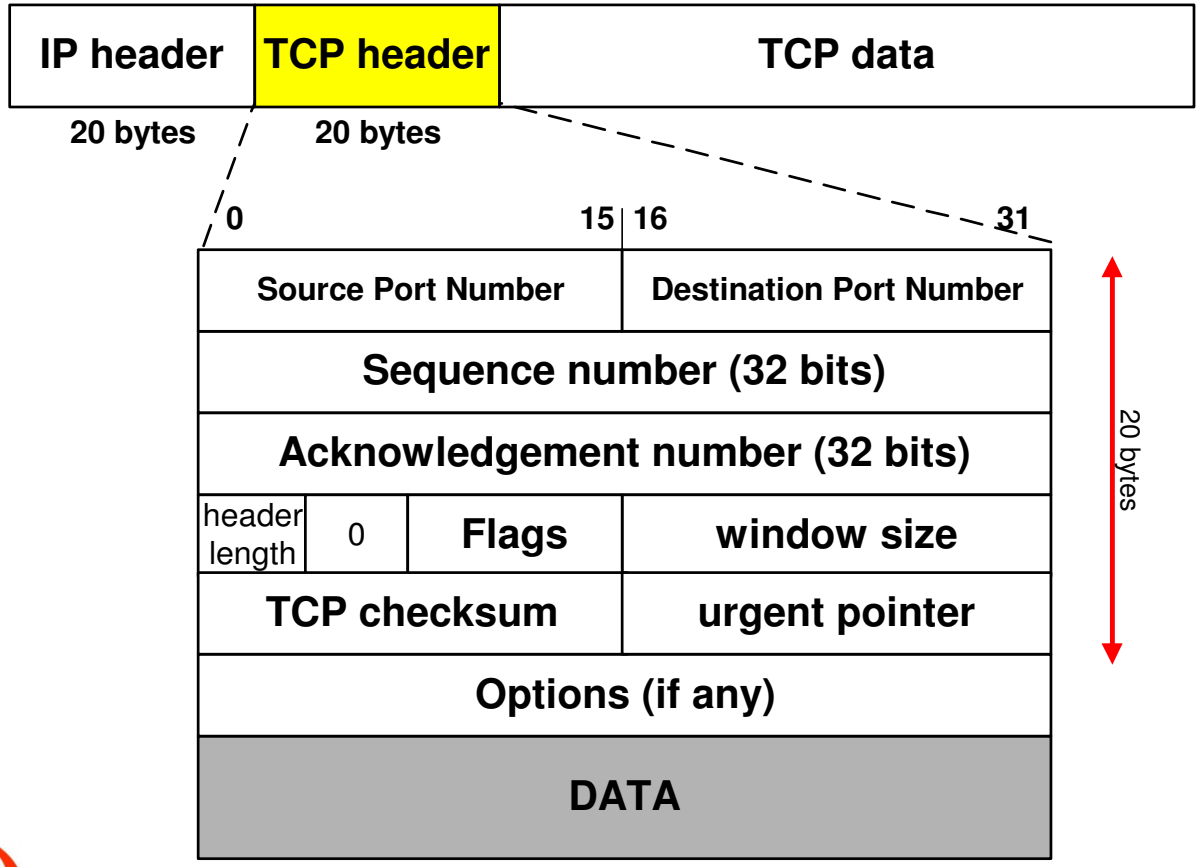
- To the lower layers, TCP handles data in blocks, the segments.
- To the higher layers TCP handles data as a sequence of bytes and does not identify boundaries between bytes
- So: Higher layers do not know about the beginning and end of segments !



TCP Cont.

TCP Format

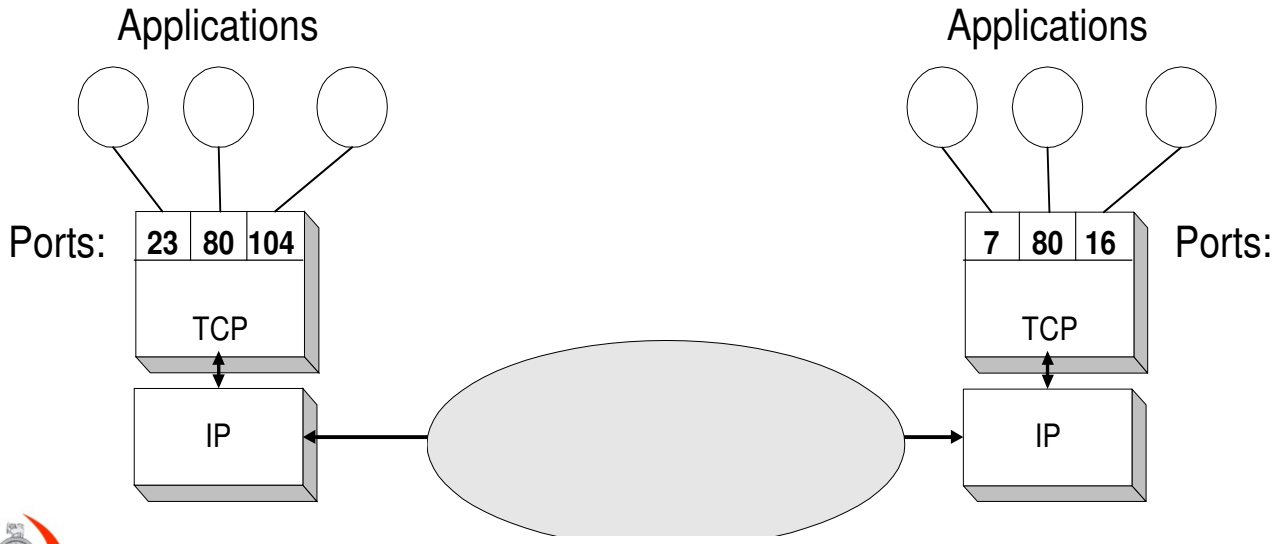
- TCP segments have a 20 byte header with >= 0 bytes of data.



TCP Cont.

TCP header fields - Port Number:

- A port number identifies the endpoint of a connection.
- A pair **<IP address, port number>** identifies one endpoint of a connection.
- Two pairs **<client IP address, server port number>** and **<server IP address, server port number>** identify a TCP connection.



TCP Cont.

TCP header fields - Sequence Number (SeqNo):

- Sequence number is 32 bits long.
- So the range of SeqNo is
    - o $0 <= SeqNo <= 2^{32} - 1 \approx 4.3$ Gbyte
- Each sequence number identifies a byte in the byte stream
- Initial Sequence Number (ISN) of a connection is set during connection establishment

TCP Cont.

TCP header fields - Acknowledgement Number (AckNo):

- Acknowledgements are piggybacked, I.e
  - a segment from A -> B can contain an acknowledgement for a data sent in the B -> A direction
- A hosts uses the AckNo field to send acknowledgements. (If a host sends an AckNo in a segment it sets the "**ACK flag**")
- The AckNo contains the next SeqNo that a hosts wants to receive

  Example: The acknowledgement for a segment with sequence numbers 0-1500 is AckNo=1501

TCP Cont.

TCP header fields - Acknowledge Number (cont'd)

- TCP uses the sliding window flow protocol to regulate the flow of traffic from sender to receiver
- TCP uses the following variation of sliding window:
    - o no NACKs (**N**egative **ACK**nowledgement)
    - o only cumulative ACKs

Header Length (4bits):

- Length of header in 32-bit words
- Note that TCP header has variable length (with minimum 20 bytes)

TCP Cont.

TCP header fields - Flag bits:

- **URG: Urgent pointer is valid**
  - If the bit is set, the following bytes contain an urgent message in the range:
    - **SeqNo <= urgent message <= SeqNo+urgent pointer**
- **ACK: Acknowledgement Number is valid**
- **PSH: PUSH Flag**
  - Notification from sender to the receiver that the receiver should pass all data that it has to the application.
  - Normally set by sender when the sender's buffer is empty

TCP Cont.

TCP header fields - Flag bits:

- **RST: Reset the connection**
  - The flag causes the receiver to reset the connection
  - Receiver of a RST terminates the connection and indicates higher layer application about the reset
- **SYN: Synchronize sequence numbers**
  - Sent in the first packet when initiating a connection
- **FIN: Sender is finished with sending**
  - Used for closing a connection
  - Both sides of a connection must send a **FIN**

TCP Cont.

TCP header fields

- **Window Size:**
  - Each side of the connection advertises the window size
  - Window size is the maximum number of bytes that a receiver can accept.
  - Maximum window size is $2^{16}-1 = 65535$ bytes
- **TCP Checksum:**
  - TCP checksum covers over both TCP header **and** TCP data (also covers some parts of the IP header)
- **Urgent Pointer:**
  - Only valid if **URG** flag is set

TCP Connection Establishment

- TCP uses a **three-way handshake** to open a connection:

(1) ACTIVE OPEN: Client sends a segment with

- SYN bit set
- port number of client
- initial sequence number (ISN) of client

(2) PASSIVE OPEN: Server responds with a segment with

- SYN bit set
- initial sequence number of server
- ACK for ISN of client

(3) Client acknowledges by sending a segment with:

- ACK ISN of server

Three-Way Handshake