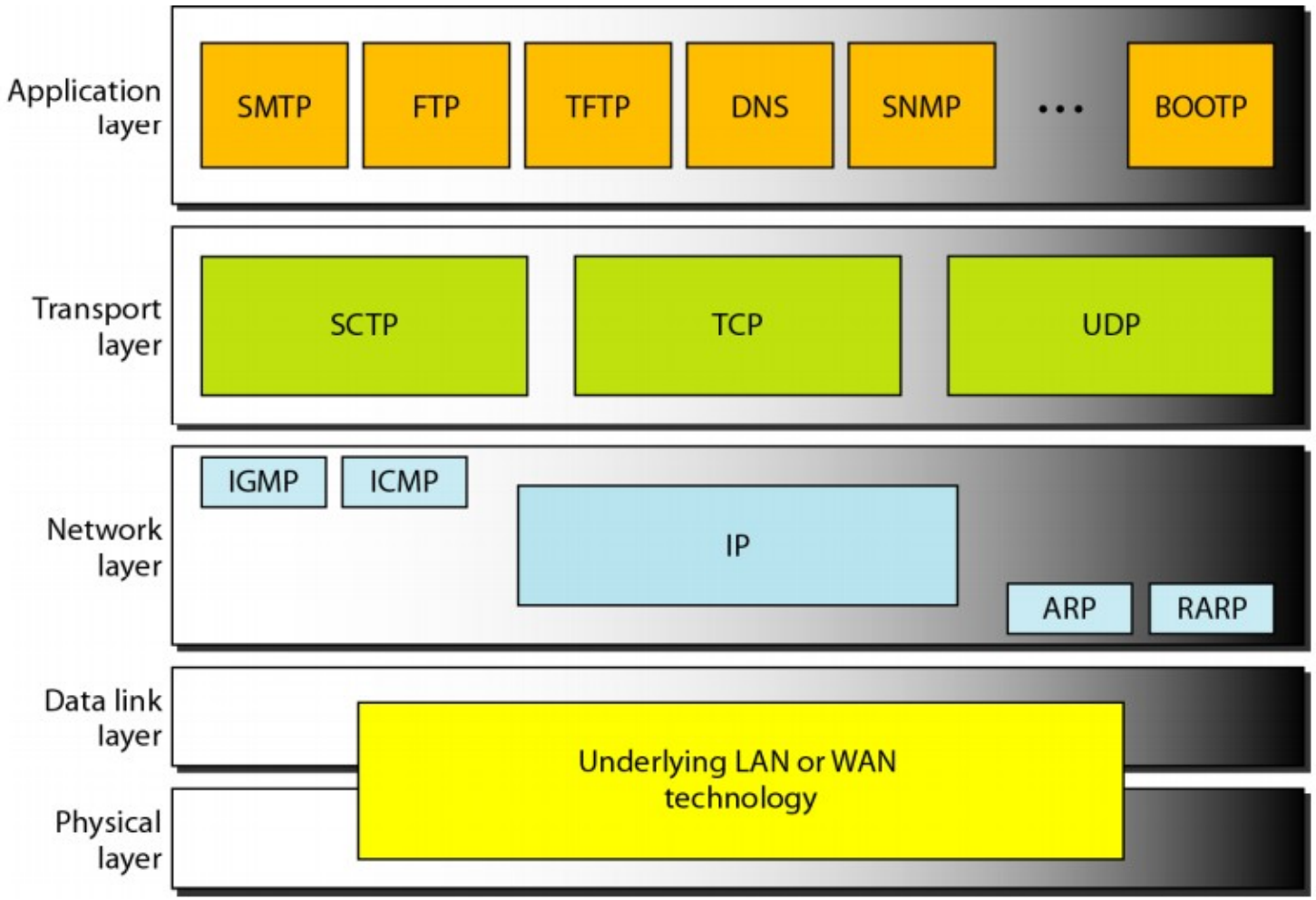
Transport Layer

- In computer networking, a transport layer provides **end-to-end communication services** for applications.
- The transport layer provides **services**
  - connection-control
  - reliability
  - flow control
  - multiplexing.
- The best-known transport protocol is the **Transmission Control Protocol (TCP),** used for connection-oriented transmissions.
- whereas the connectionless **User Datagram Protocol (UDP)** is used for simpler messaging transmissions.


Application layer
SMTP
FTP
TFTP
DNS
SNMP
• • •
BOOTP
Transport layer
SCTP
TCP
UDP
Network layer
IGMP
ICMP
IP
ARP
RARP
Data link layer
Underlying LAN or WAN technology
Physical layer

Internet Transport Protocols: UDP

- **UDP (User Datagram Protocol)**
  - connectionless transport protocol
  - UDP is basically just IP with a short header added
  - provides a way for applications to send encapsulated IP datagrams and send them without having to establish a connection
  - UDP transmits **segments** consisting of an 8-byte header followed by the payload

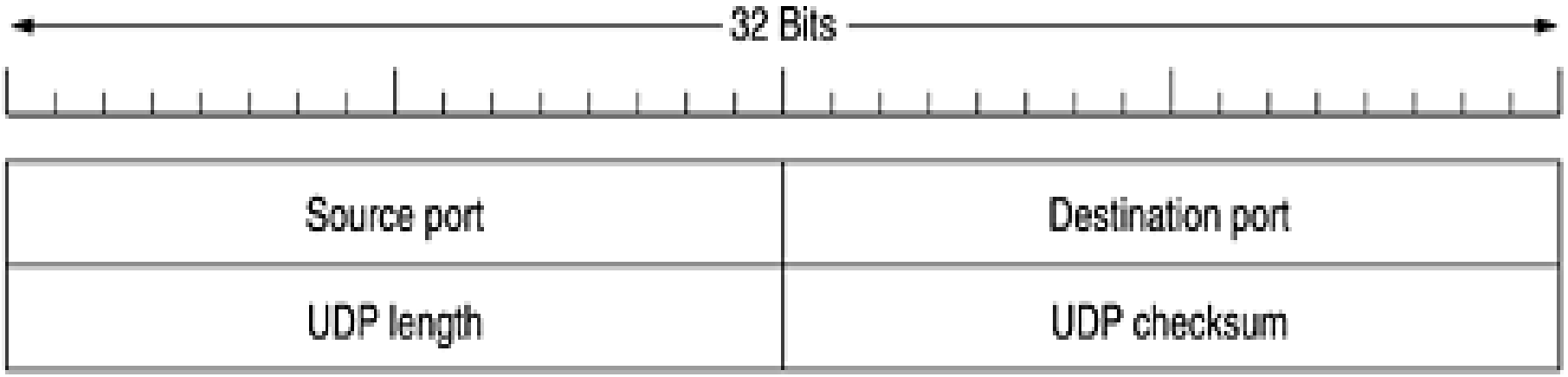


Fig: The UDP header

Internet Transport Protocols: UDP

- Two ports are used to identify the end points within the source and destination machines.
- When a UDP packet arrives, its payload is handed to the process attached to the destination port.
- Without the port fields, the transport layer would not know what to do with the packet.
- With ports, it delivers segments correctly
- source port is primarily needed when a reply must be sent back to the source
- UDP length field includes the 8-byte header and the data
- UDP checksum is optional and stored as 0 if not computed

Internet Transport Protocols: UDP

- UDP does *not do*
  - flow control,
  - error control, or
  - retransmission upon receipt of a bad segment

- UDP does
  - provide an interface to the IP protocol with the added feature of demultiplexing multiple processes using the ports

Internet Transport Protocols: UDP

- Applications:
  - client-server
    - client sends a short request to the server and expects a short reply back
    - Eg: DNS, DHCP,BOOTP,RPC, RIP,OSPF,RTP **(Real-time Transport Protocol)**
- RTP suitable for for real time applications
  - Internet radio,
  - Internet telephony,
  - music-on-demand,
  - videoconferencing,
  - video-on-demand

Internet Transport Protocols: TCP

- for most Internet applications, reliable, sequenced delivery is needed
- UDP cannot provide this, so another protocol is required.
- It is called **TCP** (**Transmission Control Protocol)**
- TCP was designed to
  - provide a reliable end-to-end byte stream over an unreliable internetwork
  - dynamically adapt to properties (different topologies, bandwidths, delays, packet sizes, etc) of the internetwork
  - be robust in the face of many kinds of failures

Internet Transport Protocols: TCP

- **TCP Service Model**
  - TCP service is obtained by both the sender and receiver creating end points called sockets
  - Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port.**
  - A port is the TCP name for a TSAP.
  - For TCP service to be obtained, a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine.
  - A socket may be used for multiple connections at the same time.
  - Connections are identified by the socket identifiers at both ends, that is, (*socket1, socket2)*

Internet Transport Protocols: TCP

- Port numbers below 1024 are called **well-known ports** and are reserved for standard services.

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial file transfer protocol
79	Finger	Lookup information about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

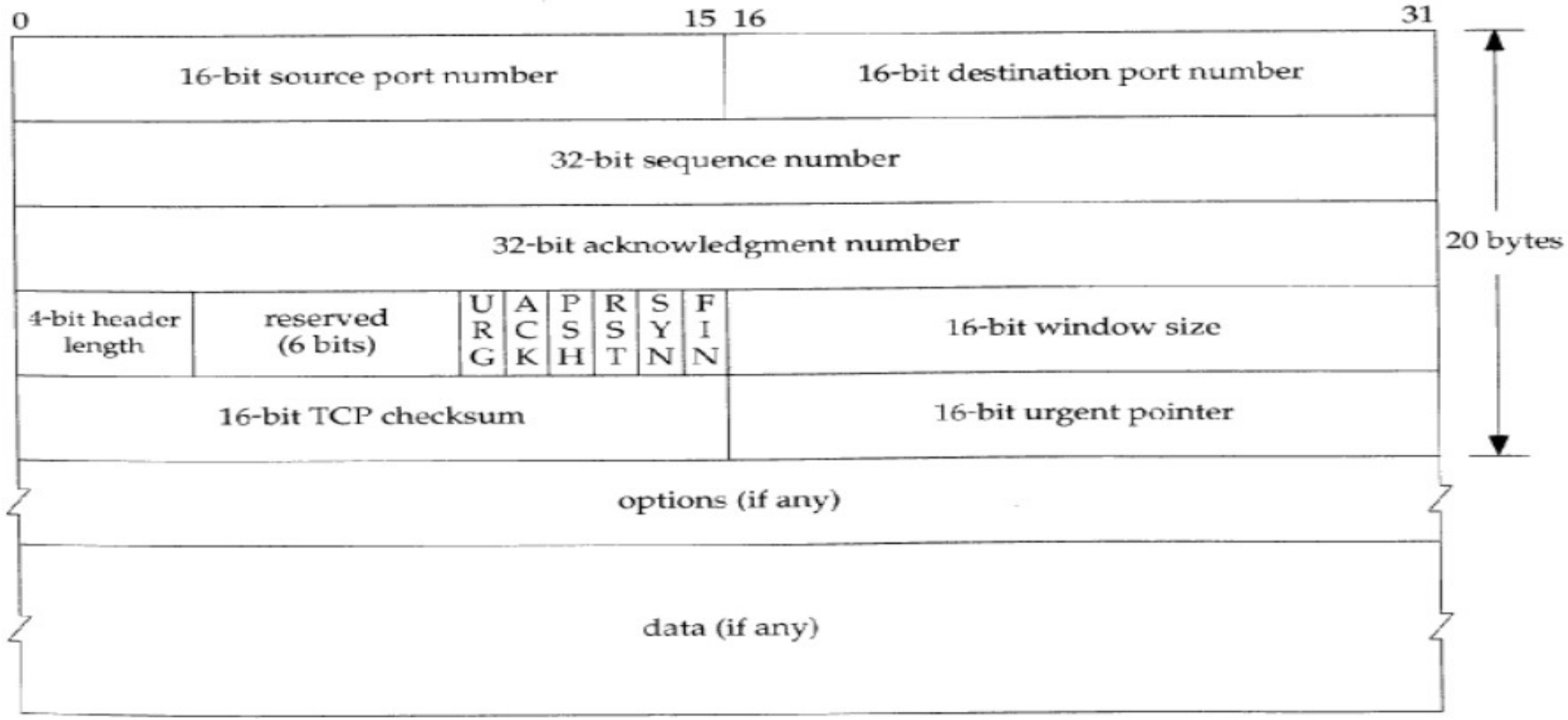
Fig: Some assigned ports.

Internet Transport Protocols: TCP

- All TCP connections are full duplex and point-to-point.
  - Full duplex
    - traffic can go in both directions at the same time.
  - Point-to-point
    - each connection has exactly two end points.
- TCP does not support multicasting or broadcasting.
- TCP connection is a byte stream, not a message stream.
- To force data out immediately, applications can use the PUSH flag, which tells TCP not to delay the transmission
- URGENT flag causes TCP to stop accumulating data and transmit everything it has for that connection immediately

TCP HEADER

- **TCP data** is encapsulated in an IP datagram.
- Its normal size is 20 bytes unless options are present.



TCP Header

1. Source port number (16 bits)

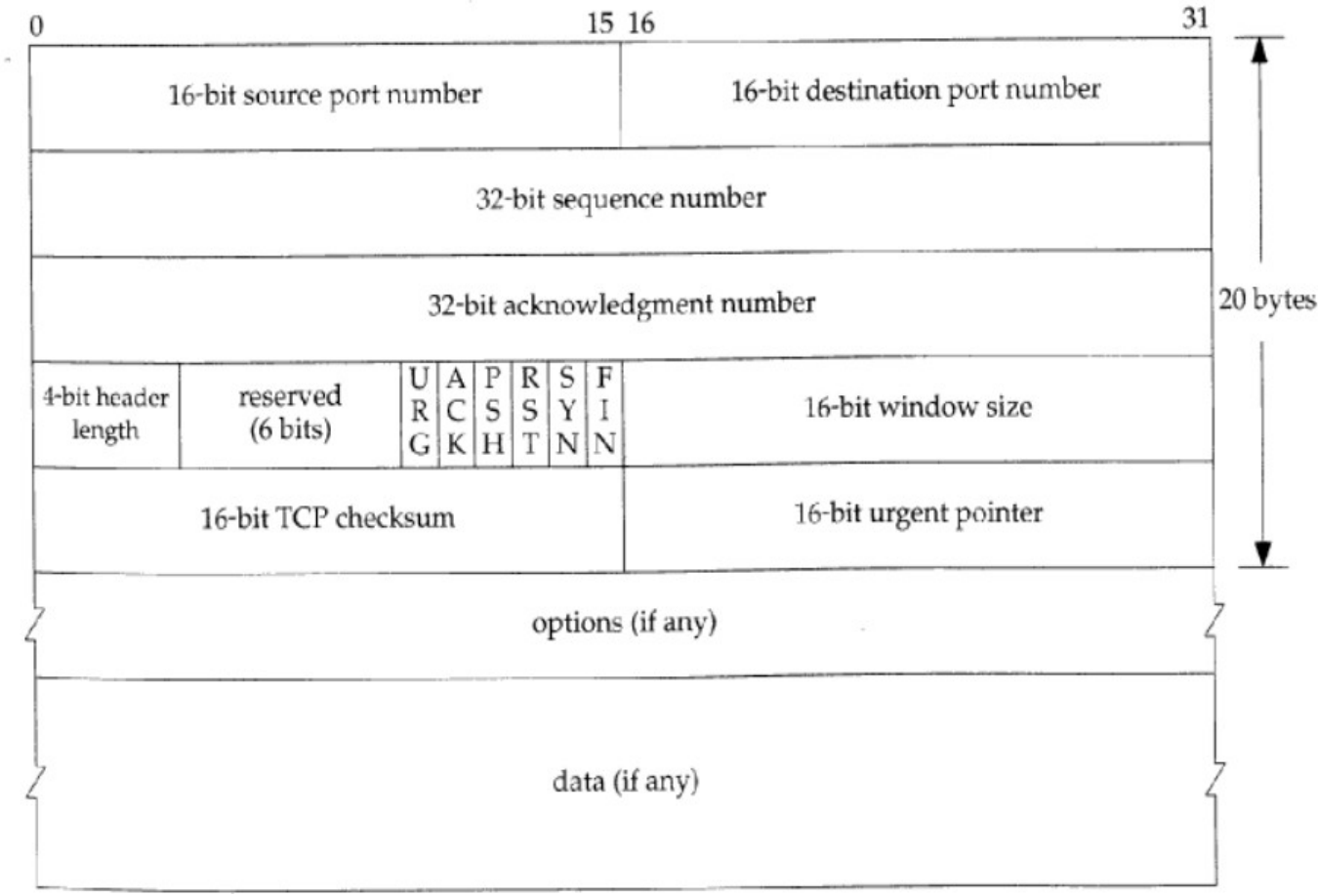
- identifies the **TCP process** which sent the datagram.
- 20=FTP,23=Telnet,53=DNS,80=HTTP

2. Destination port number (16 bits)

- identifies the **TCP process** which is receiving the datagram.

3. Sequence number (32 bits)

- identifies the **first byte** of the outgoing data.
- The **receiver** uses this
  - to re-order segments arriving



TCP Header

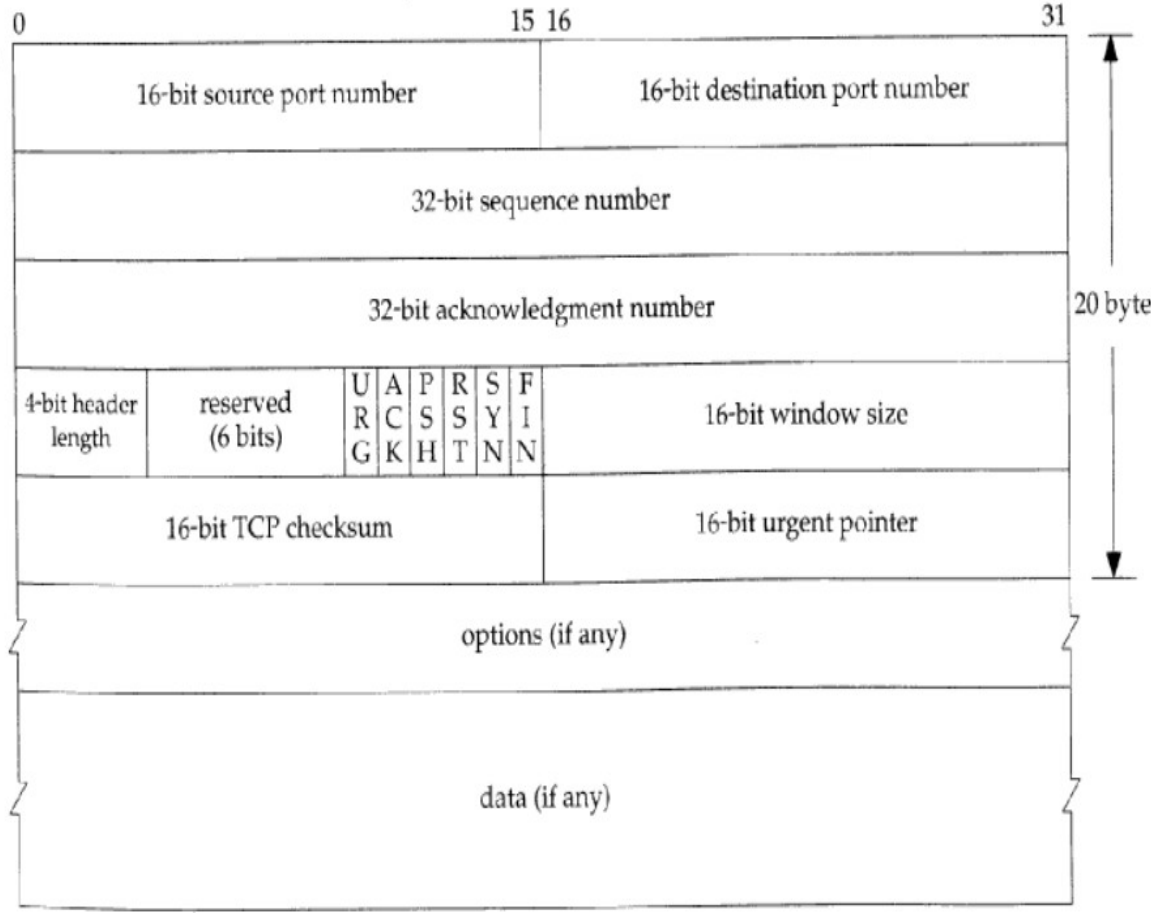
4. Acknowledgement number (32 bits)

- Contains the next sequence number that the sender of the acknowledgement expects to receive .
  - which is the **sequence number plus 1** (plus the number of bytes received in the last message).
- This number is used only if the ACK flag is on.

5. Header Length

- The length of the header can be between 20 and 60 bytes.

6. Reserved – This is a 6-bit field reserved for future use



TCP Header

The six one-bit flags

- used to relay control information between TCP peers.
- The possible flags include **SYN, FIN, RESET, PUSH, URG, and ACK**.

7. URG

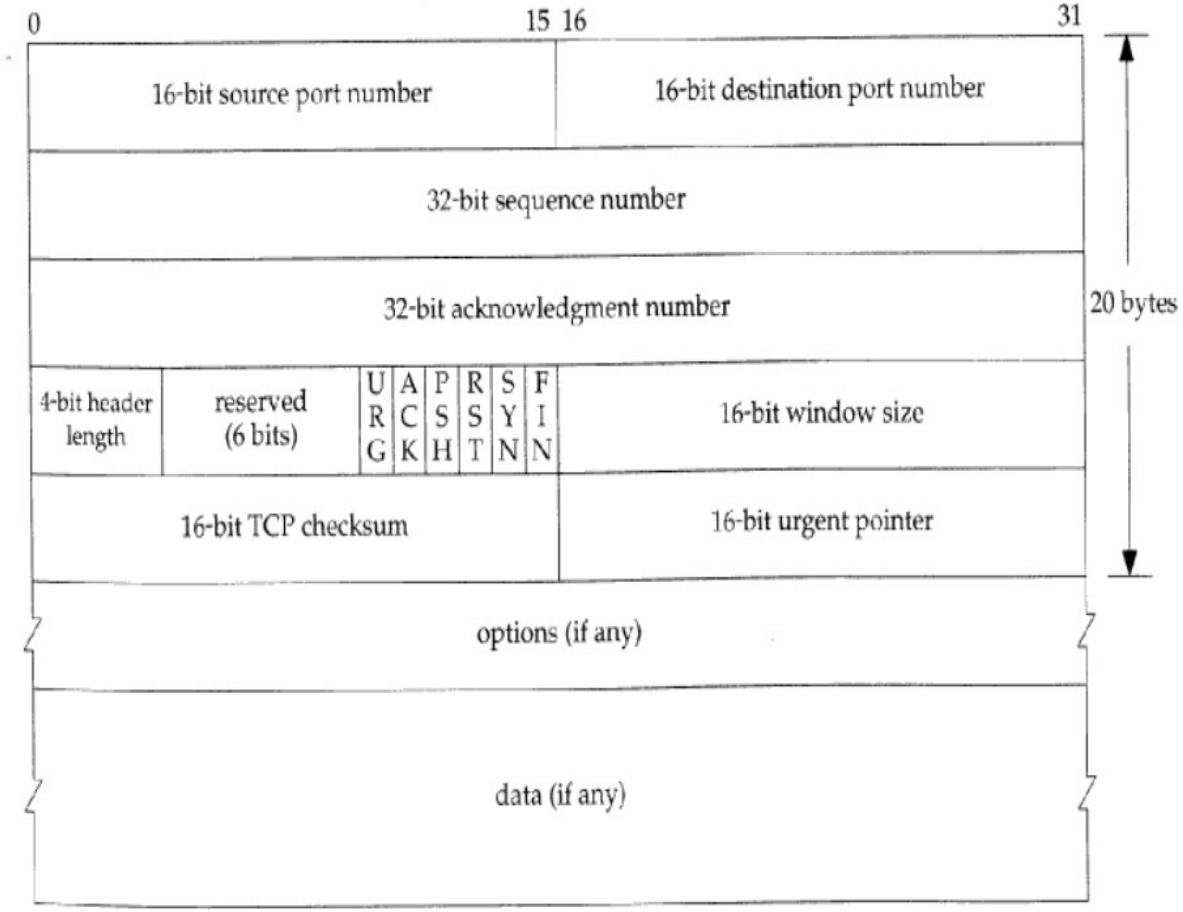
- This bit indicates whether the urgent pointer field in this packet is being used.

8. ACK

- This bit is set to indicate the ACK number field in this packet is valid.

9. PSH

- This bit indicates **PUSHed data**.
- The **receiver** is requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received.



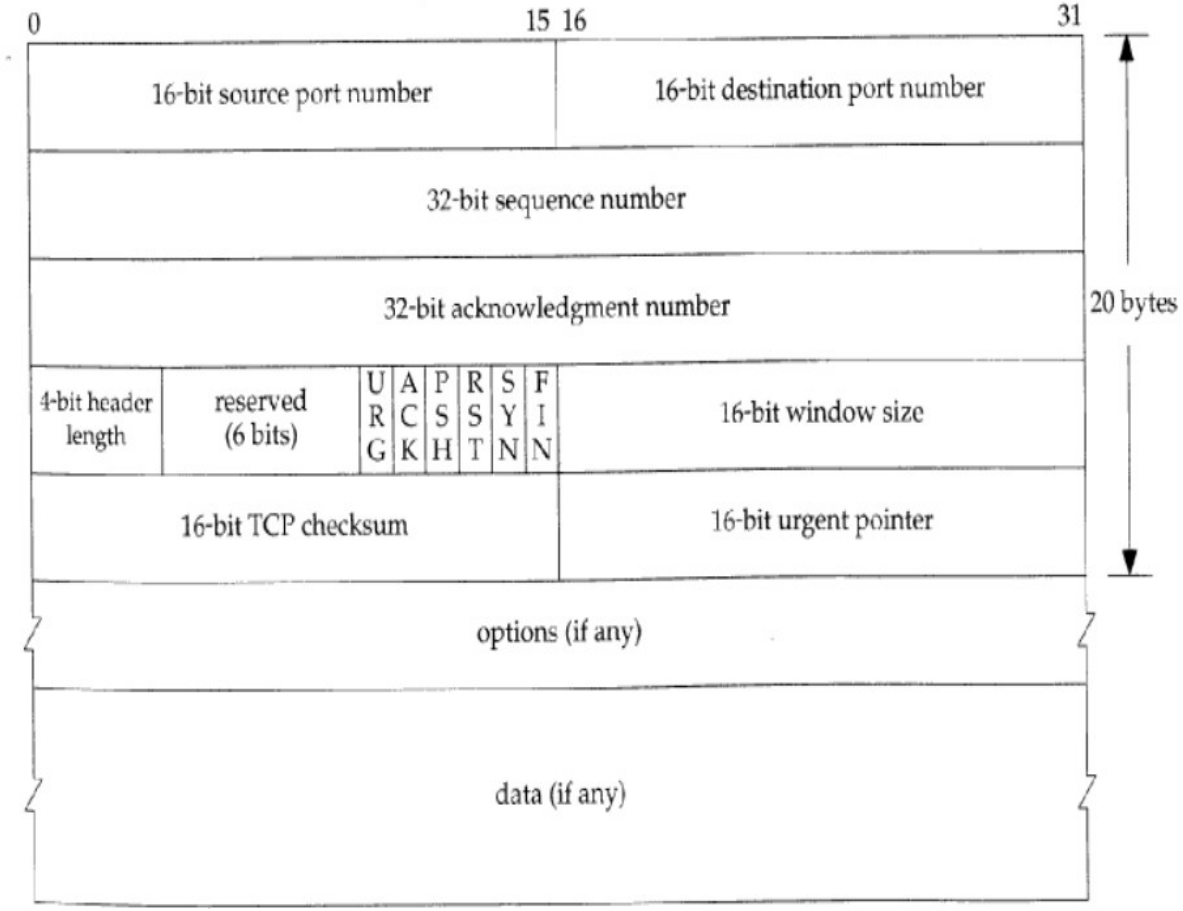
TCP Header

10. RST

- This flag is used to reset a connection that has become confused due to a host crash or some other reason.

11.SYN

- This bit is used to **establish connections**.
- The connection request(1st packet in 3-way handshake) has SYN=1 and ACK=0.
- The connection reply (2nd packet in 3-way handshake) has SYN=1 and ACK=1.



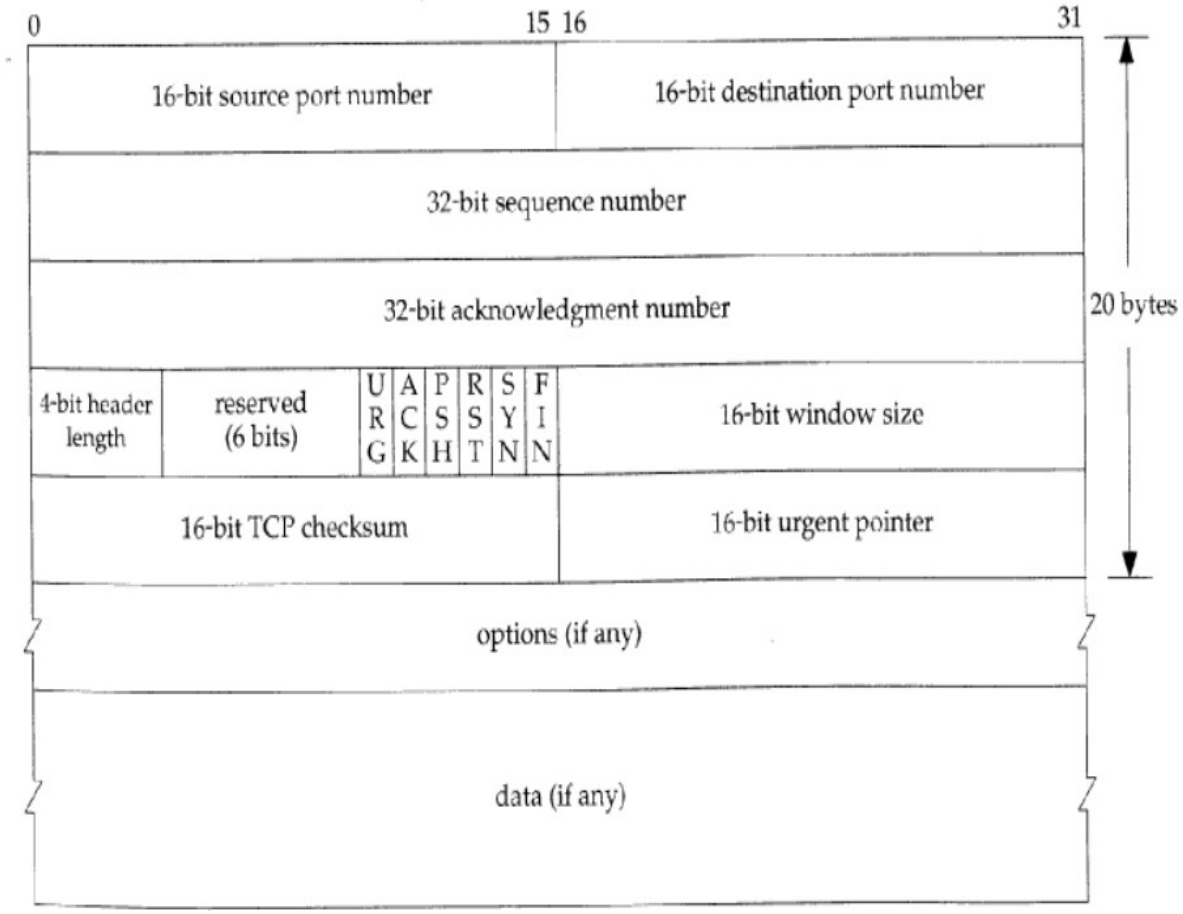
TCP Header

12. FIN

- This bit is used to release a connection.
- It specifies that the **sender** has no more fresh data to transmit.
- However, it will retransmit any lost or delayed packet.
- Also, it will continue to receive data from other side.

13. Window Size(16 bit)

- identifies how much buffer space is available for incoming data.



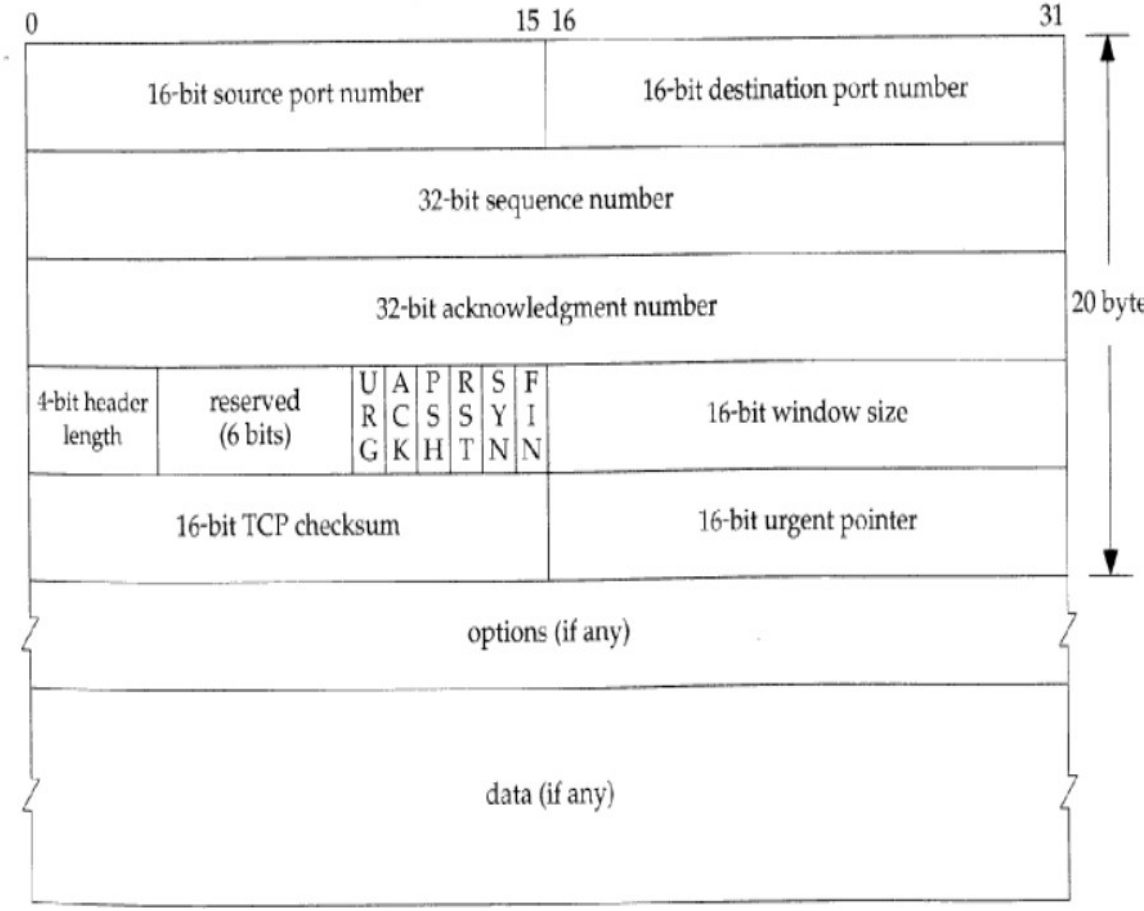
TCP Header

14.Checksum(16 bit)

- field contains a simple checksum over the TCP segment header and data.

15.Urgent Pointer (16 bit)

- valid only if the urgent flag is set, is used when the segment contains urgent data.
- It **defines the number** that must be added to the sequence number to obtain the number of the last **urgent byte** in the data section of the segment.



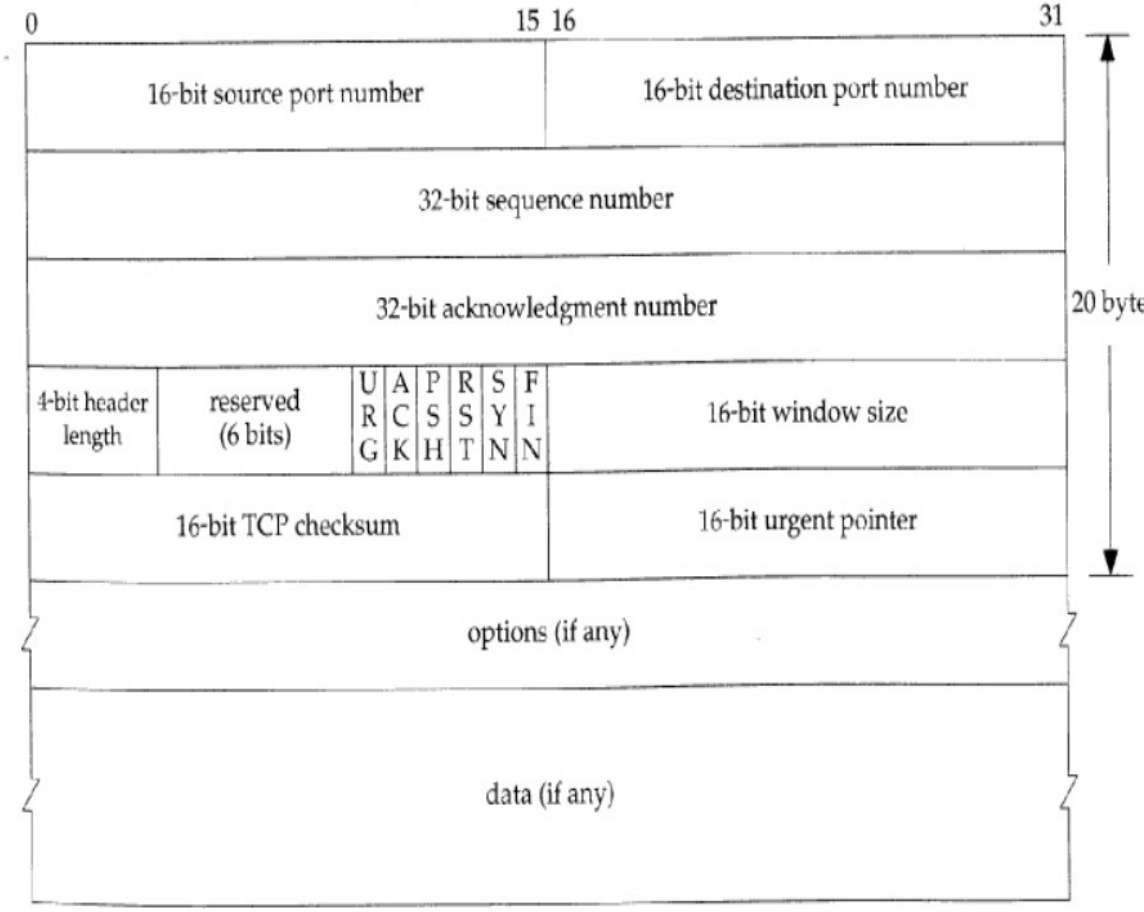
TCP Header

16.Options

- There can be up to **40 bytes** of optional information in the TCP header.

17.Data

- This can be of variable size.
- which can be up to 65535 – 20 = 65515 bytes.



A TCP CONNECTION

TCP is connection-oriented. It establishes a virtual path between the source and destination. All of the segments belonging to a message are then sent over this virtual path. You may wonder how TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented. The point is that a TCP connection is virtual, not physical. TCP operates at a higher level. TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself. If a segment is lost

In TCP connection-oriented transmission requires two phases:

- Connection establishment and Data transfer
- Connection termination

Connection establishment:

- TCP transmits data in full-duplex mode.
- When two TCP's in two machines are connected, they are able to send segments to each other simultaneously.

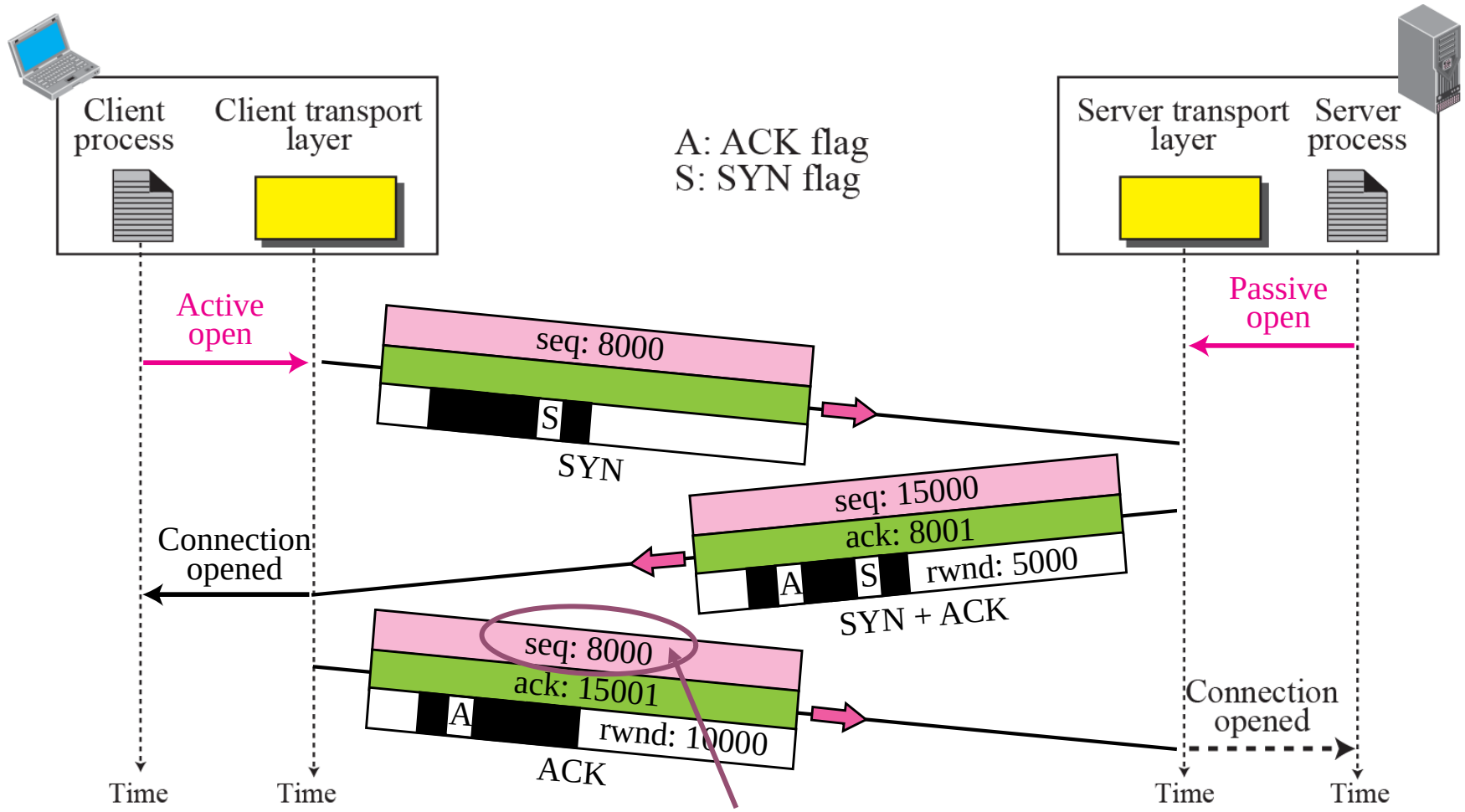
Three way handshake

Step 1: client host sends TCP SYN segment to server
- specifies initial seq #
- no data

Step 2: server host receives SYN, replies with SYNACK segment
- server allocates buffers
- specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data

Connection establishment using three-way handshake



Means "no data" !

seq: 8001 if piggybacking

Note

A SYN segment cannot carry data, but it consumes one sequence number.

Note

A SYN + ACK segment cannot carry data, but does consume one sequence number.

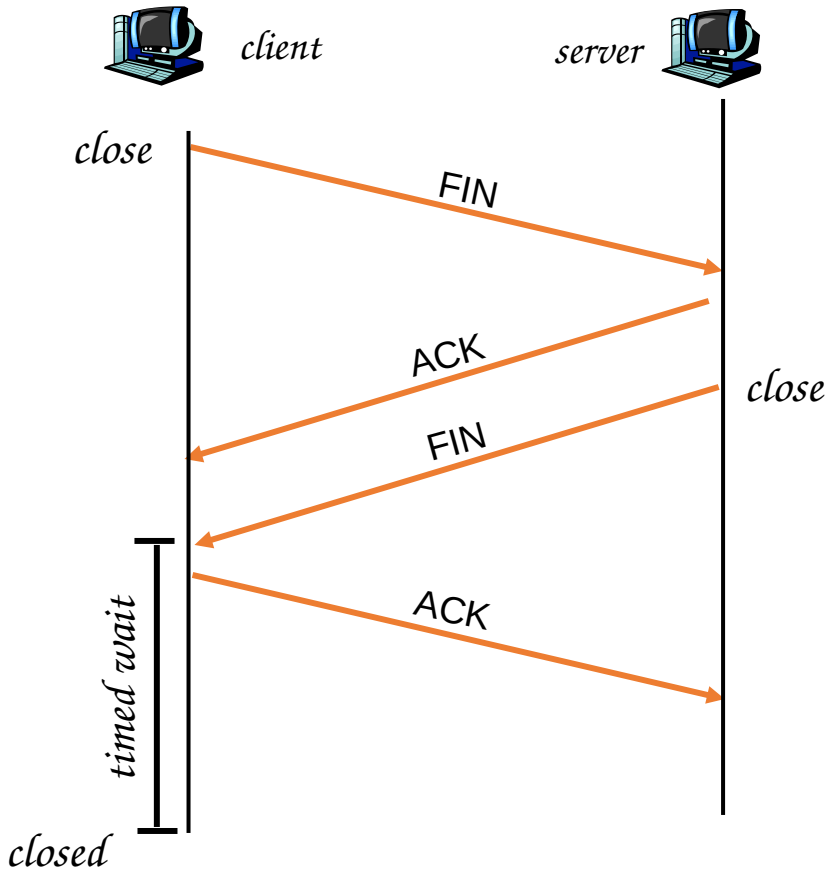
Note

An ACK segment, if carrying no data, consumes no sequence number.

TCP Connection Management (cont.)

Step 1: client end system sends TCP FIN control segment to server

Step 2: server receives FIN, replies with ACK. Closes connection, sends FIN.

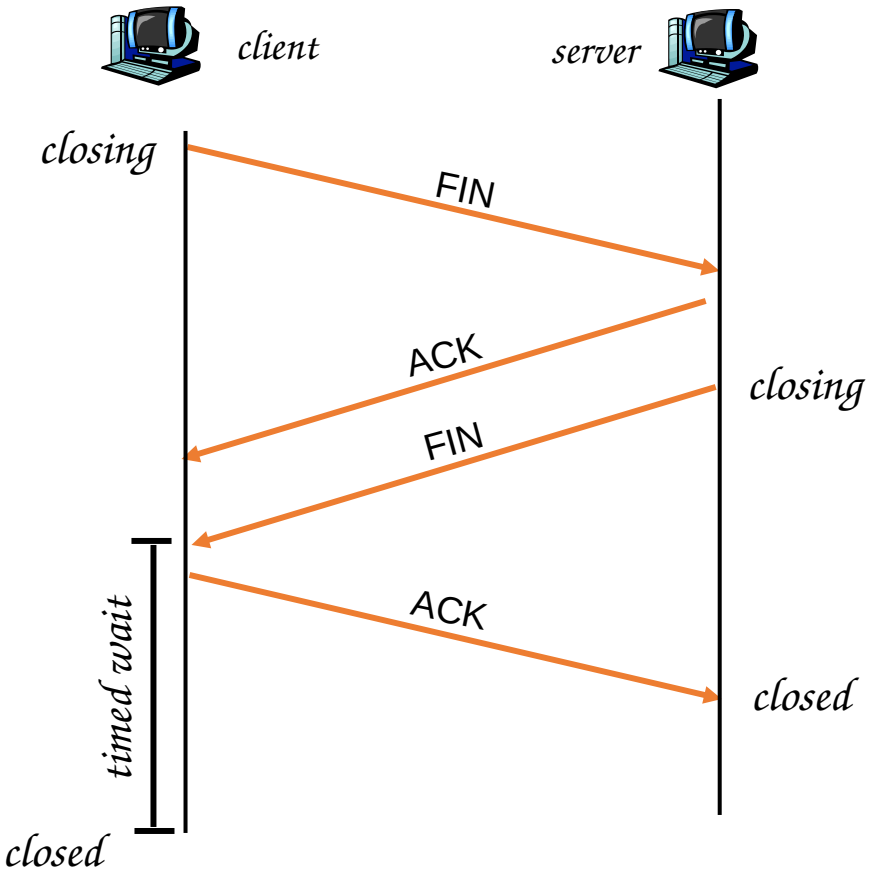


TCP Connection Management (cont.)

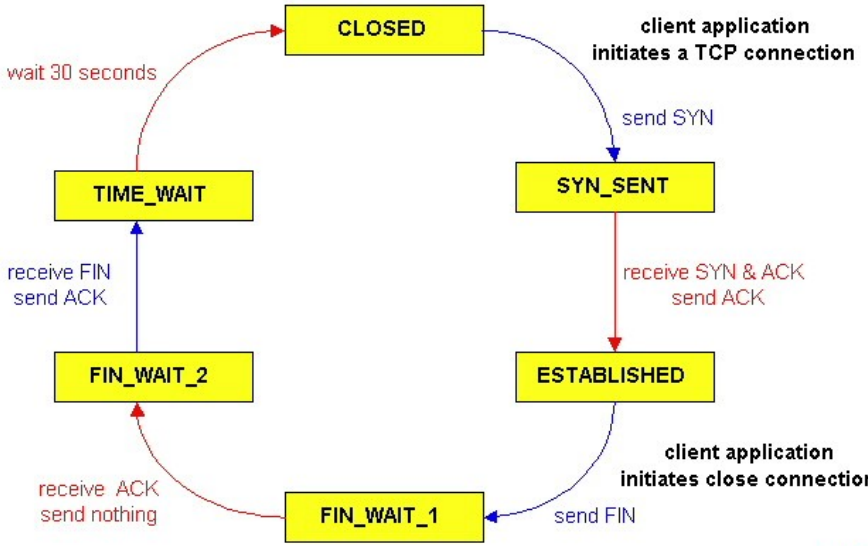
Step 3: client receives FIN, replies with ACK.

- Enters “timed wait” - will respond with ACK to received FINs

Step 4: server, receives ACK. Connection closed.

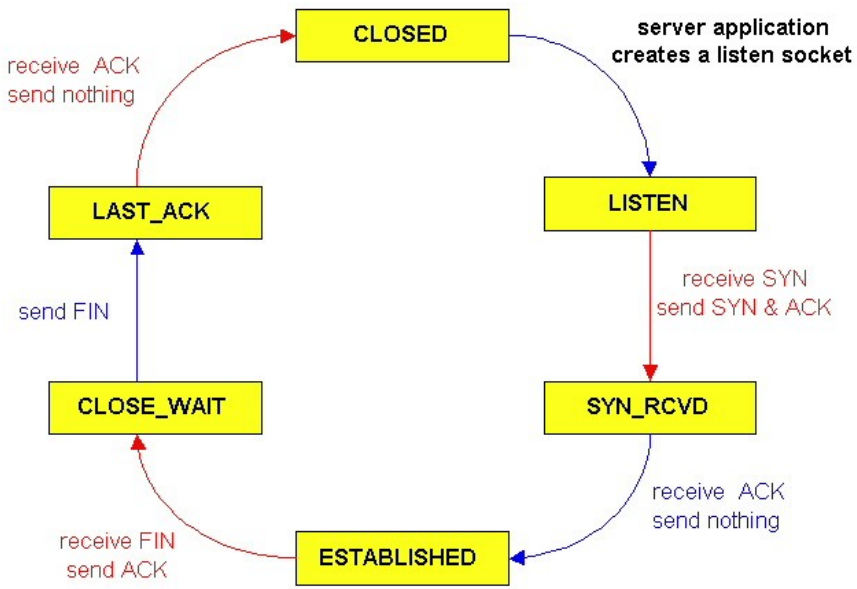


TCP Connection Management (cont)



TCP client lifecycle

TCP server lifecycle



	TCP	UDP
Acronym for	Transmission Control Protocol	User Datagram Protocol or Universal Datagram Protocol
Connection	TCP is a connection-oriented protocol.	UDP is a connectionless protocol
Function	As a message makes its way across the internet from one computer to another. This is connection based.	UDP is also a protocol used in message transport or transfer. This is not connection based which means that one program can send a load of packets to another and that would be the end of the relationship.
Usage .	TCP is suited for applications that require high reliability, and transmission time is relatively less critical.	UDP is suitable for applications that need fast, efficient transmission, such as games. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients
Use by other protocols	HTTP, HTTPs, FTP, SMTP, Telnet	DNS, DHCP, TFTP, SNMP, RIP, VOIP.
Ordering of data packets	Segment sequencing. TCP rearranges data packets in the order specified	No Segment sequencing. UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer.
Speed of transfer	The speed for TCP is slower than UDP.	UDP is faster because error recovery is not attempted. It is a "best effort" protocol.
Reliability	Reliable, There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	Unreliable ,There is no guarantee that the messages or packets sent would reach at all.

Header Size Common	TCP header size is 20 bytes	UDP Header size is 8 bytes
Header Fields	Source port, Destination port, Check Sum	Source port, Destination port, Check Sum
Streaming of data	Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries	Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.
Weight	TCP is heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP is lightweight. There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.
Data Flow Control	TCP does Flow Control. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP does not have an option for flow control
Error Checking .	TCP does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination.	UDP does error checking but simply discards erroneous packets. Error recovery is not attempted
Fields	1. Sequence Number, 2. AcK number, 3. Data offset, 4. Reserved, 5. Control bit, 6. Window, 7. Urgent Pointer 8. Options, 9. Padding, 10. Check Sum, 11. Source port, 12. Destination port	1. Length, 2. Source port, 3. Destination port, 4. Check Sum
Acknowledgement	Acknowledgement segments. Acknowledge sequencing.	No Acknowledgment No Acknowledge sequencing
Handshake	SYN, SYN-ACK, ACK	No handshake (connectionless protocol)