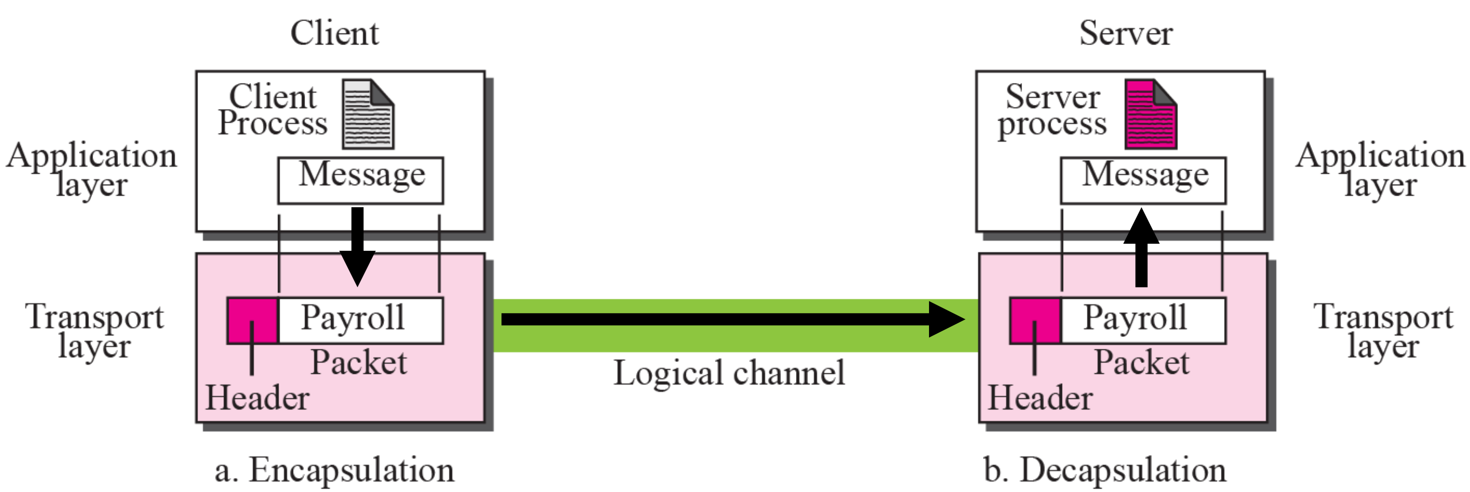
Transport Layer: Packetization

(2) Packetization – application layer produces a stream of data ⇒ transport layer is responsible for:

must feature !!!

1) **breaking large messages** into smaller pieces that network layer can handle
2) **adding/removing header** to/from each packet to enable reassembly at receiver



Transport Layer: Other Functions

(3) Error Control
(4) Flow Control
(5) Connection Setup
(6) Congestion Control

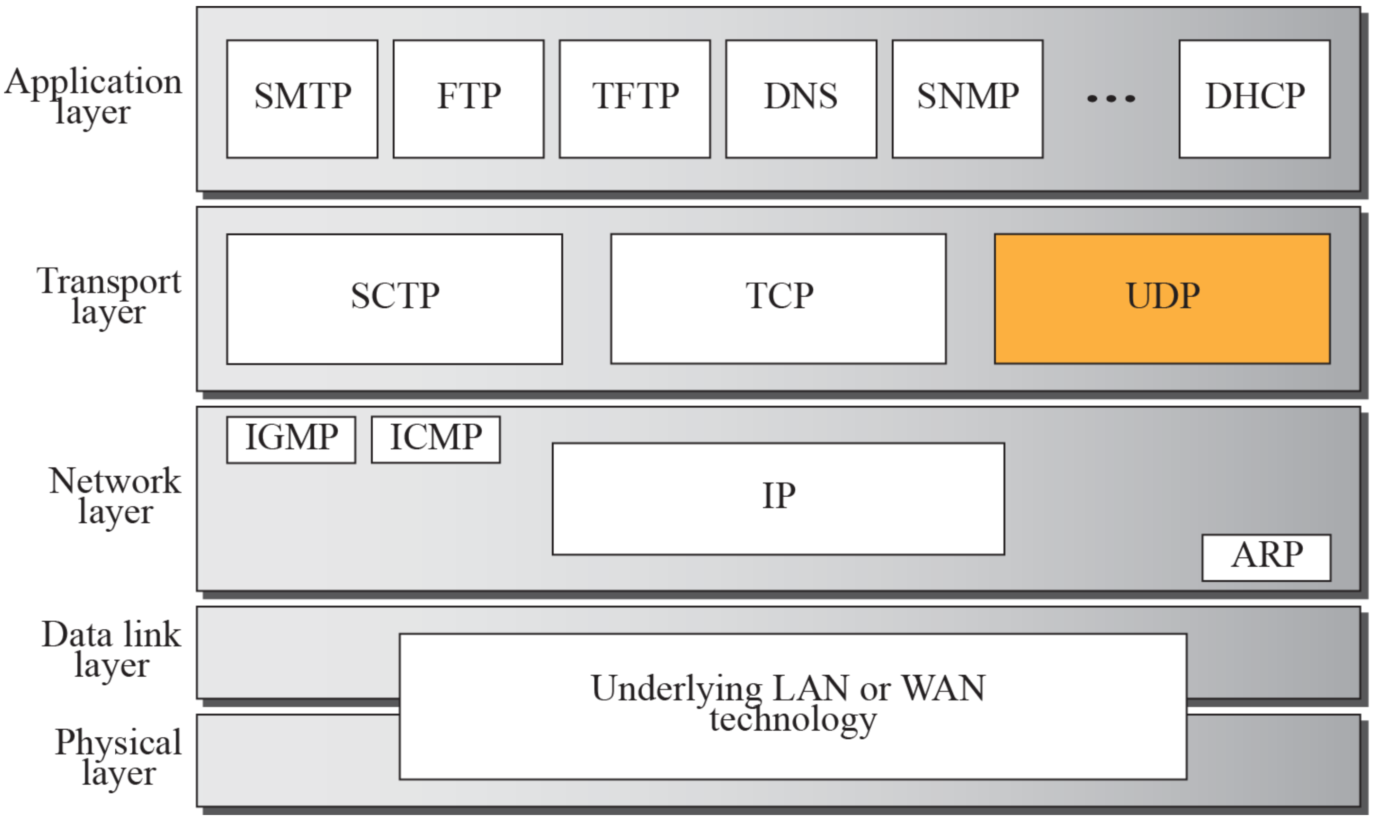
– **transport layer in TCP/IP networks provide two types of process-to-process packet delivery:**

1) **connectionless (UDP-protocol based)**
   - **2 basic func. + limited error control** (optional)
   - **results in unreliable + unordered delivery**

2) **connection oriented (TCP-protocol based)**
   - **2 basic func. + …**
   - **error control**
   - **flow control**
   - **connection setup**
   - **congestion control**
   - **results in reliable + in-order packet delivery**

By providing 2 minimal transport layer services (+ limited error checking)
UDP is no-frills extension of "best-effort" IP.

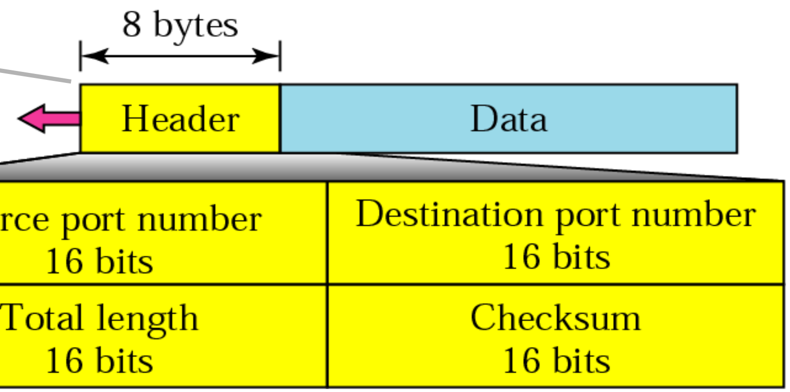
Connectionless Transport: UDP



Connectionless Transport: UDP (cont.)

UDP Datagram – **8 byte header + application data**

header = 4 fields only each consisting of 2 bytes



Total Length – **defines total length of UDP datagram, <u>header + data</u>**

- **16 bits ⇒ theoretical size 0 – 65,535 bytes; however many OSs limit TCP packet size to 512 to 1024 bytes**

Checksum – **used to detect errors over <u>entire</u> UDP datagram (header + data)**

- **calculation of checksum is optional! - <u>if it is not calculated, the field is filled with 0s</u>**
- **if an error is detected, the segment is discarded, and no further action is taken**
- **why UDP checksum if data-link layers provide error checking?!** **there is no guarantee that all links provide error checking**

Connectionless Transport: UDP (cont.)

User Datagram Protocol (UDP) – **connectionless**, **unreliable** (i.e. no flow control and very limited error checking) protocol – "**best effort**" service

- **UDP does not add anything to the service of IP except enabling process-to-process multiplexing / demultiplexing**

If UDP is so powerless – cannot guarantee packet delivery + packets are delivered out of order – why would a process want to use it?!

UDP Advantages

1) **no connection establishment** which adds delay
   - TCP performs 3-way handshake before sending any data
2) **no connection state** – less processing overhead
   - TCP maintains connection state in the end systems, e.g receive and send buffers, sequence and ACK numbers, …
3) **small packet header** – only 8-byte overhead
   - TCP requires 20-bytes of header
4) **no flow/congestion control** – UDP can send packets as fast as desired

Connectionless Transport: UDP (cont.)

UDP Application

1) **UDP is used to carry network management packets, e.g. SNMP and DNS**
   - **network management applications often run when the network is in a stressed state, i.e. when it is difficult to achieve <u>reliable</u> data transfer**

2) **UDP is suitable for applications that require simple <u>timely request-response</u> communication with little concern for flow and <u>error control</u>**
   - **e.g. Echo, Daytime, Users (**e.g. no point to resend old Daytime until correctly received**)**

3) **UDP is often used for real-time multimedia applications**
   - **multimedia applications are often rate sensitive, yet they can tolerate small amount of <u>packet loss</u> or (e.g.) increase compression rate to match available bandwidth in the network**
   - **WARNING! assume everyone is to start streaming high-bit-rate video ⇒ lack of congestion control in UDP can result in high loss rates of UDP packets and crowding out of TCP sessions ☹**

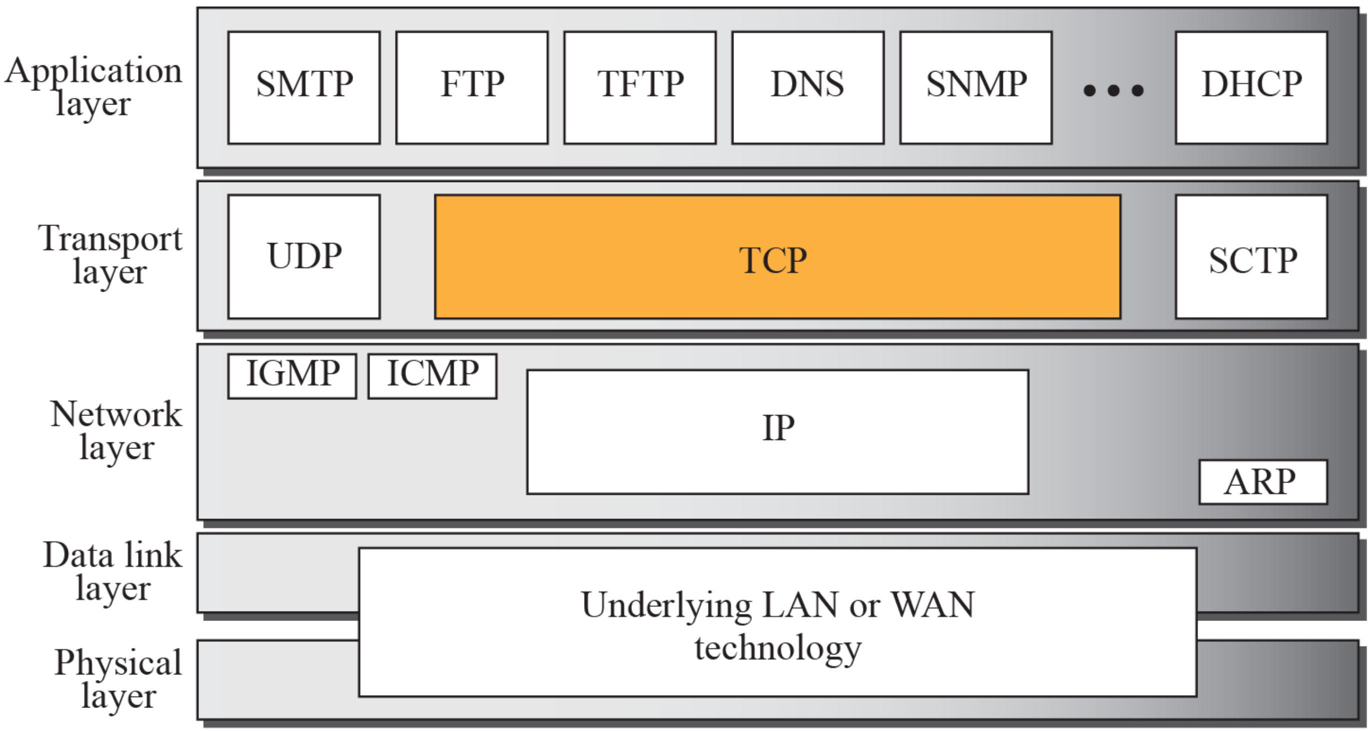
4) **UDP is suitable for multicast and broadcast applications**
   - **<u>TCP supports only point-to-point, i.e. unicast, applications</u>**

Connectionless Transport: UDP (cont.)

Example **[applications that use UDP]**

Port	Application Layer Protocol	Description
7	Echo	Echoes a received datagram back to the sender
11	Users	Active users
13	Daytime	Returns the date and the time
53	DNS	Domain Name Service
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol

Connection-Oriented Transport: TCP



Connection-Oriented Transport: TCP (cont.)

Transmission Control Protocol (TCP) – transport-layer protocol with following properties

- **connection-oriented**: two processes must first "handshake" with each other (agree upon the willingness to communicated) before exchanging any data
- **flow controlled**: sender will not overwhelm receiver
- **reliable = error controlled + ordered**: TCP includes mechanisms for detecting corrupted, lost, out-of-order, and duplicated segments
- **timely**: if TCP fails to deliver data within a certain timeout, it notifies the user of service failure and abruptly terminates connection
- **congestion controlled**: TCP limits amount of data entering the network to amount the network can carry
- **point-to-point**: TCP connection is always between a single sender and a single receiver
- **full-duplex**: each TCP has a sending and receiving buffer; data flows in both directions simultaneously

Connection-Oriented Transport: TCP (cont.)

Example **[applications that use TCP]**

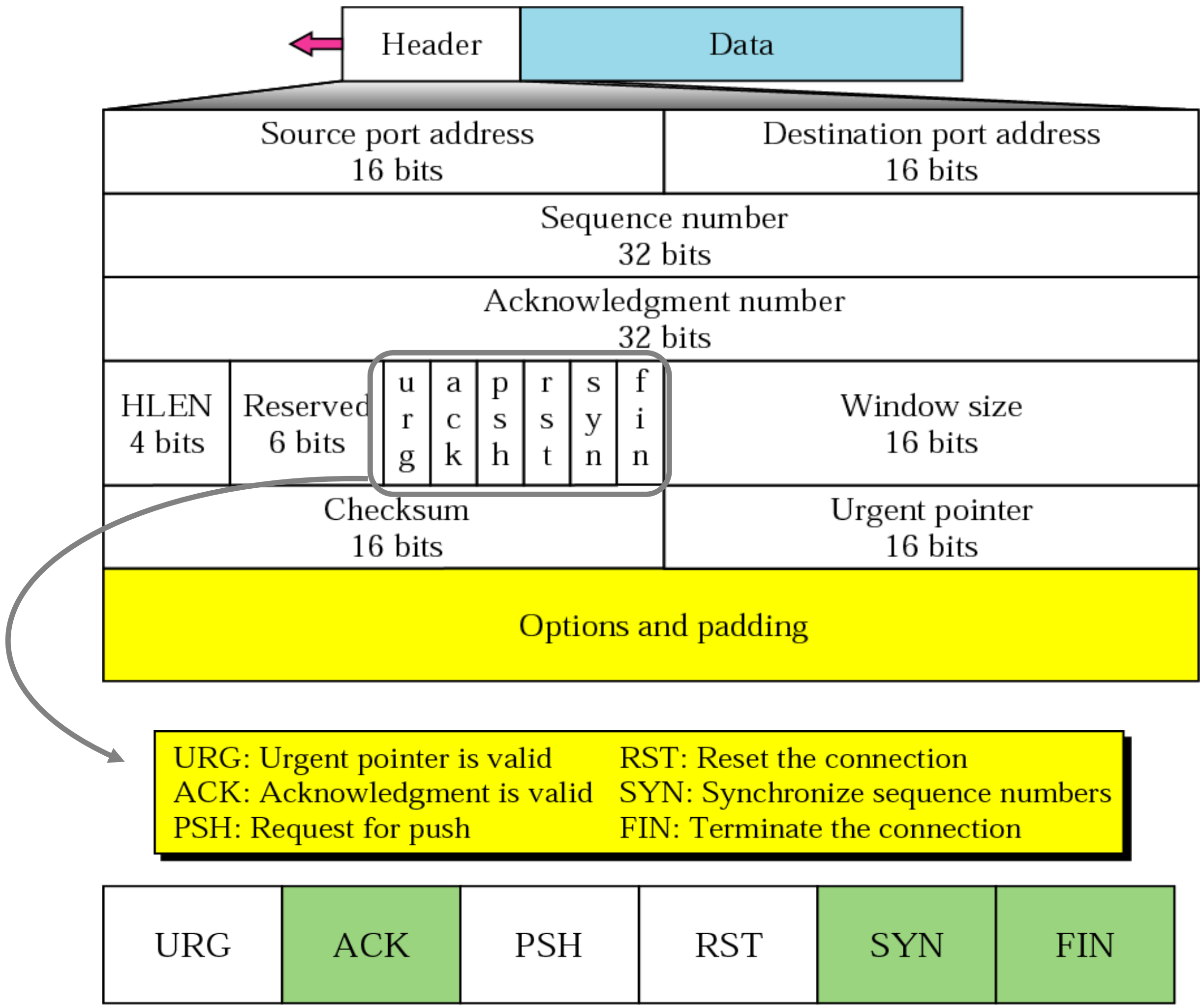
Port	Application Layer Protocol	Description
7	Echo	Echoes a received datagram back to the sender
11	Users	Active users
13	Daytime	Returns the date and the time
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

TCP provides reliable data delivery to ‘mission-critical’ applications, where delivery must be guaranteed, such as: file transfers, transactions processes, database services.

Connection-Oriented Transport: TCP (cont.)

TCP Datagram – **20- byte header + application data**

- **header can be extended to 60-bytes with "options"**



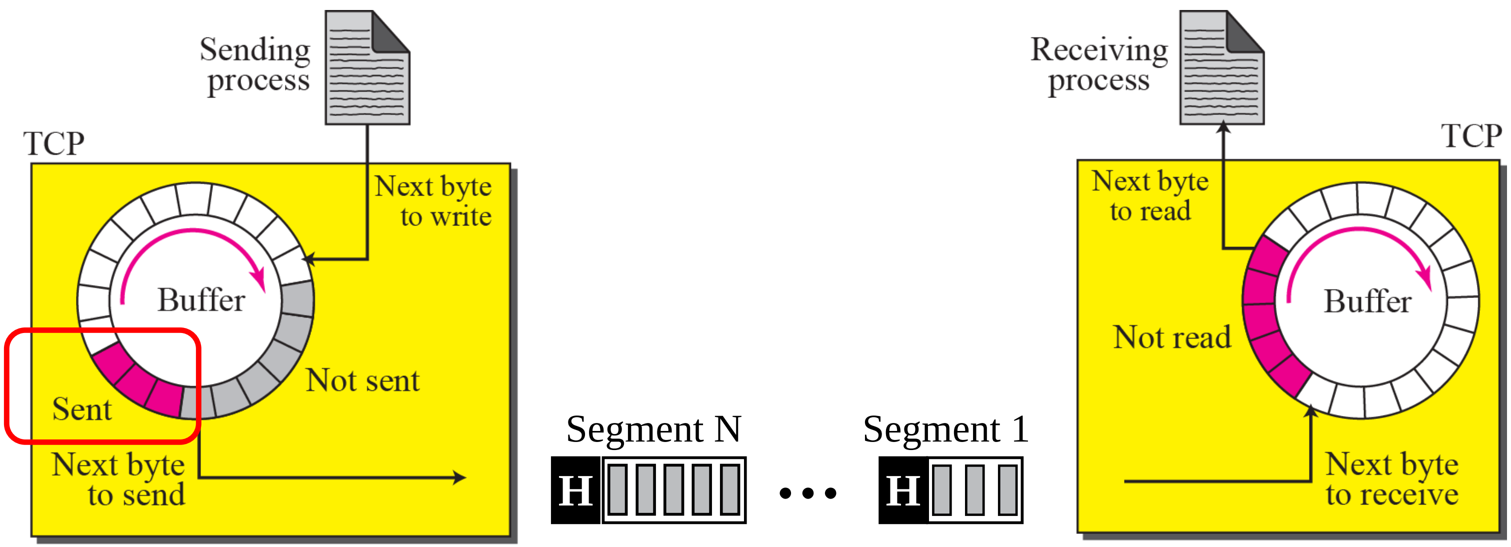
Connection-Oriented Transport: TCP (cont.)

Sequence # – **32-bit field – represents the byte-stream # of the first byte in the segment**

Acknowledgment # – **32-bit field – represents the byte-stream # of the next byte that host is expecting to receive from the other party – cumulative ACKs!**

- **if the byte numbered x has been successfully received, x+1 is the acknowledgment number**
- **pure acknowledgment = TCP segment with no data; otherwise acknowledgment is said to be piggybacked**

why are sent packets still kept in the buffer



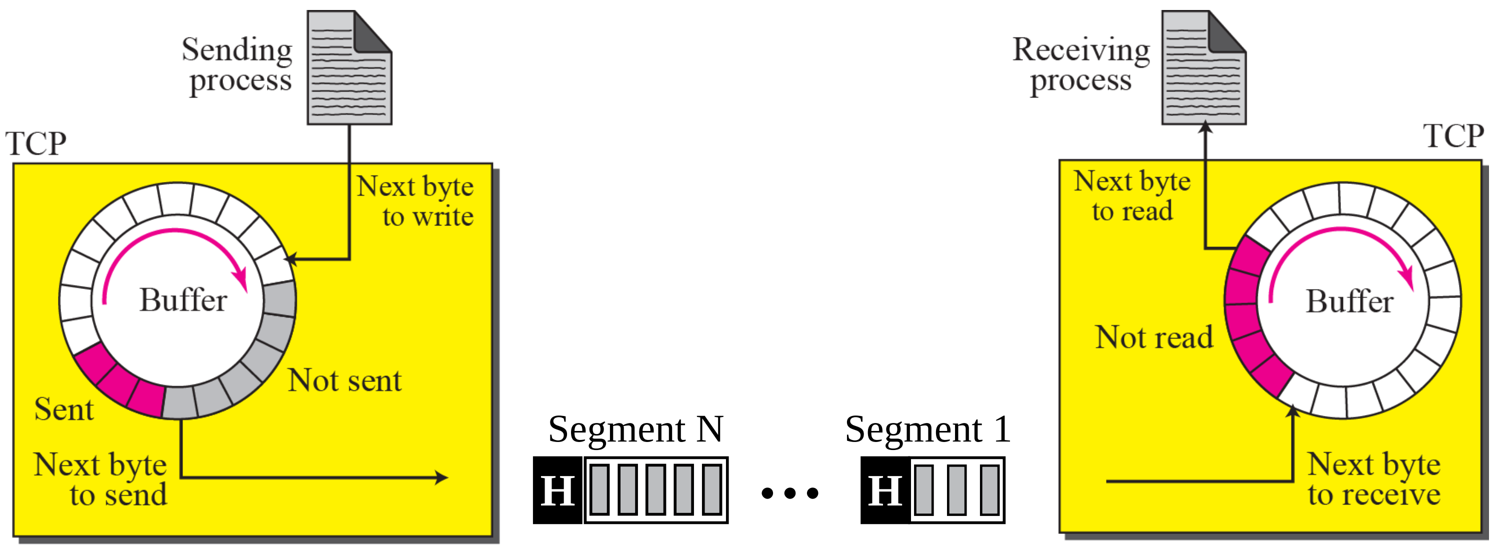
Connection-Oriented Transport: TCP (cont.)

Header Length – **4-bit field – represents the [# of 4-byte words] in the header**

- **header length 20 - 60 bytes ⇒ field value always 5 - 15**

Reserved – **6-bit field – reserved for future use**

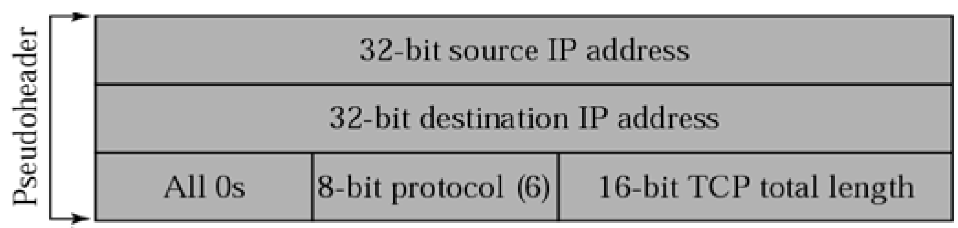
Window Size – **16-bit field – defines the # of bytes, beginning with sequence number indicated in the acknowledgment field that receiver is willing to accept**



Connection-Oriented Transport: TCP (cont.)

Checksum – 16-bit field – used to detect errors over entire TCP datagram (header + data) + 96-bit pseudoheader conceptually prefixed to header at the time of calculation

- pseudoheader contains several fields from IP header: source and destination IP addresses, protocol and segment length field
- pseudoheader offers protection against misdelivery by IP assume multiple bit errors in IP header: in IP dest. address & other ⇒ IP checksum Ok ☹ ⇒ TCP entity will detect the delivery error

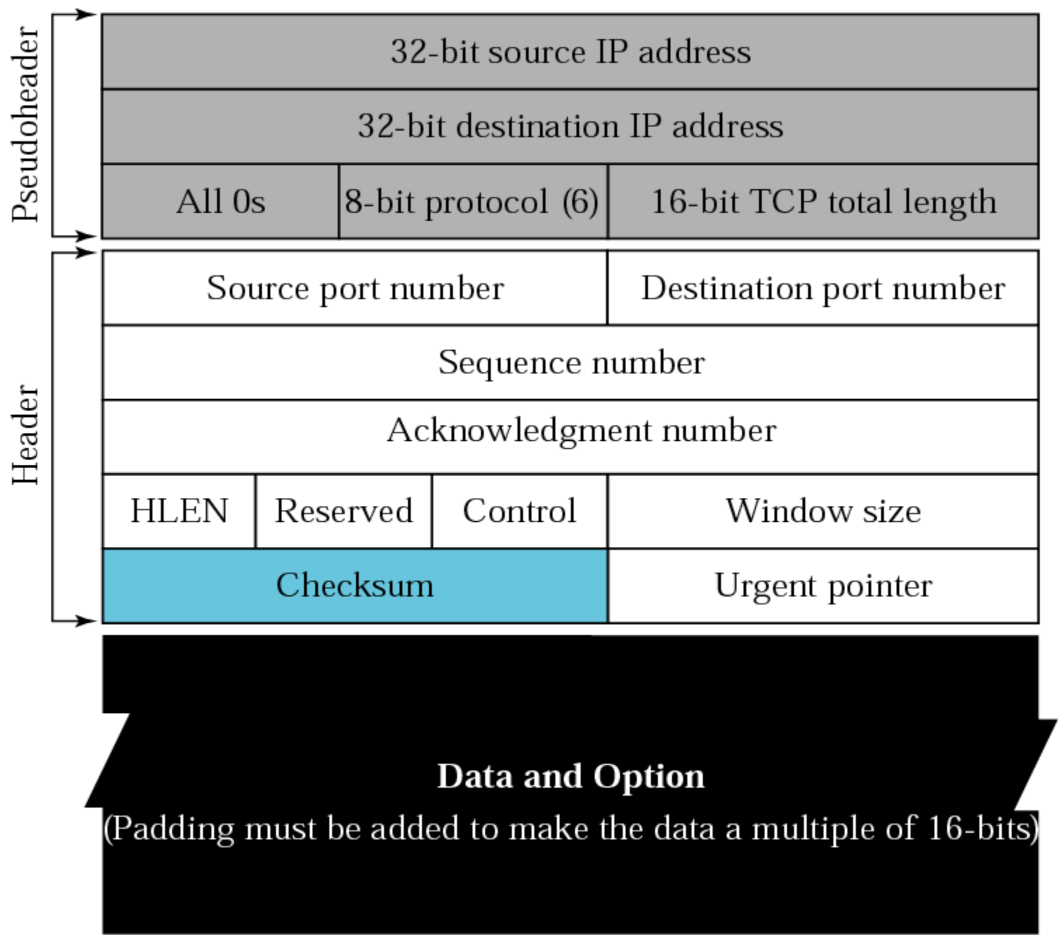


Urgent Pointer – 16-bit field – valid only if the urgent flag is set – contains the sequence # of the last byte in a sequence of urgent data

Options – there can be up to 40 bytes of optional information in the TCP header mostly related to flow/congestion control …

Padding – ensures that TCP header ends and data begins on 32-bit boundary – padding is composed of 0-s

Example **[Pseudoheader added to the TCP datagram]**



The inclusion of the checksum in TCP is mandatory!

Connection-Oriented Transport: TCP (cont.)

Control Flags

Flag	Description
URG	If this bit field is set, the receiving TCP should interpret the urgent pointer field. Used when a section of data should be read out by the receiving application quickly. The rest of the segment is processed normally.
ACK	If this bit field is set, the acknowledgement field is valid.
PSH	If this bit field is set, the sender/receiver should deliver this segment to the TCP module/receiving application as soon as possible, without waiting for *receive window* to get filled.
RST	If this bit is present, it signals the receiver that the sender is <u>aborting</u> the connection and all queued data and allocated buffers for the connection can be freely relinquished.
SYN	When present, this bit field signifies that sender is attempting to "synchronize" sequence numbers. This bit is used during the initial stages of connection establishment between a sender and receiver.
FIN	If set, this bit field tells the receiver that the sender has reached the end of its byte stream for the current TCP connection.