Transport Layer, UDP, TCP

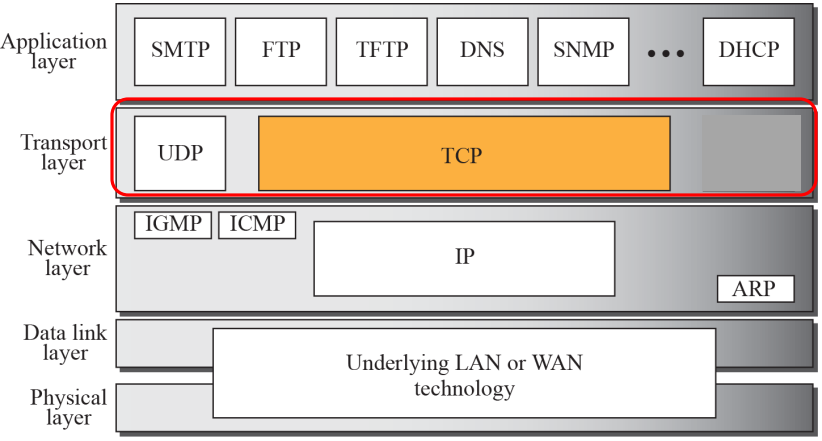
Required reading:
Kurose 3.2, 3.3.1, 3.3.2, 3.5.1, 3.5.2, 3.5.6

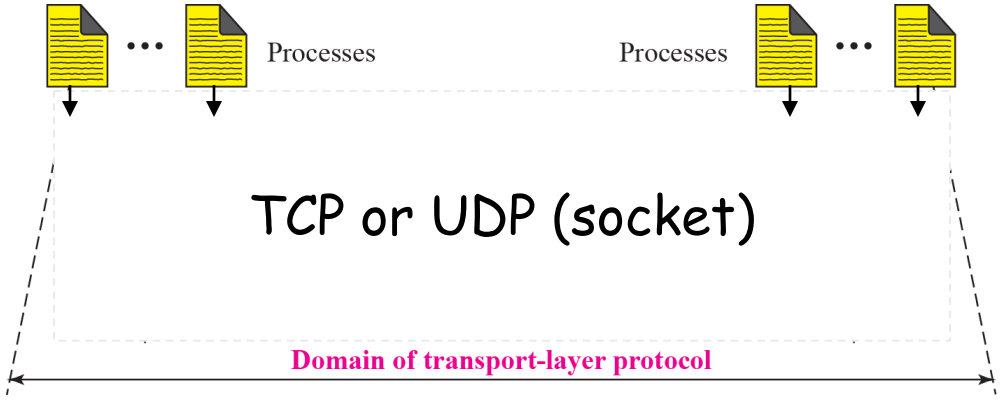
EECS 3214, Winter 2020
Instructor: N. Vlajic

Transport Layer

Transport Layer – layer that oversees delivery of data from a process, i.e. running application, on one computer to a process on another computer

- transport layer enhances network layer service and enables logical communication between processes (not hosts!)
- to application-layer programs physical networks are a homogeneous cloud that takes data and somehow delivers it to its destination





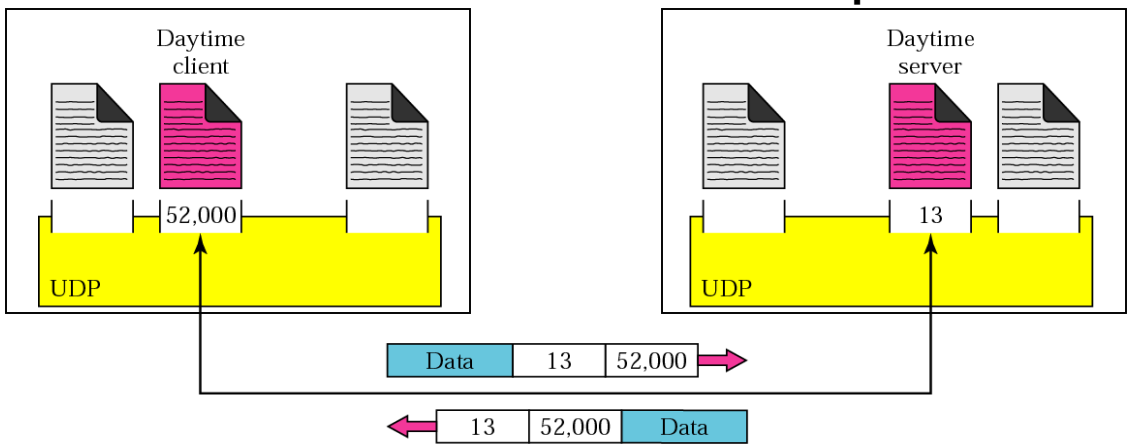
Transport Layer Responsibilities:

(1) Addressing – must feature !!!

transport layer address (port number) distinguishes among multiple processes running on the host; processes can be:

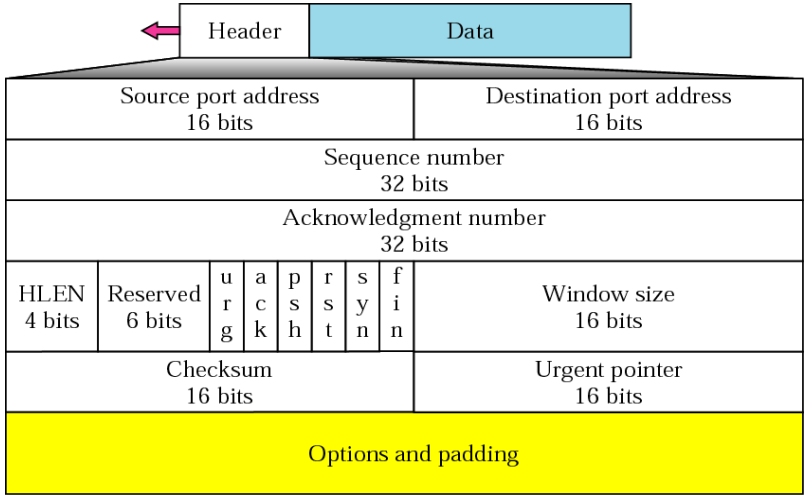
- **client program defines itself with a port number chosen randomly by the transport layer software running on the client host – these are called ephemeral port number**
- **server program uses well-known port numbers so that client program does not have to enquire about port number each time it requests service ⇒ less overhead**

client – demands service **server – provides service, always 'on'**



What is the largest port number?

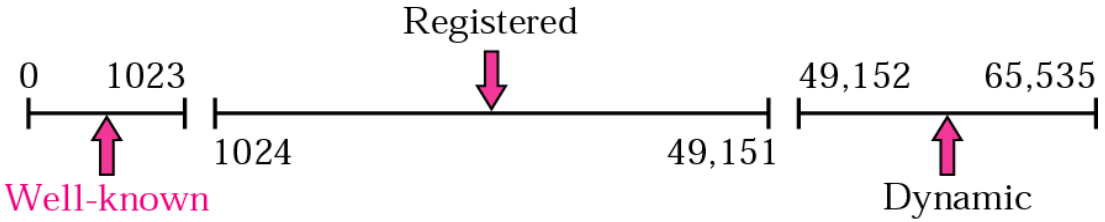
How, specifically, are port numbers distributed?



Transport Layer: Addressing (cont.)

Port Number – **16-bit integers between 0 and 65,535 used in TCP or UDP communication to name the end of logical connection**

- **well-known port numbers (0-1023) – aka reserved / contact ports – assigned & controlled by ICANN** (Internet Corporation for Assigned Names & Numbers) **– can only be used by standardized applications running as servers and passively listening for connections (e.g. FTP = 21, TELNET = 23, HTTP = 80, etc.)**
- **registered port numbers (1024 – 49,151) – registered with ICANN, to prevent applications from conflicting with each other – they are typically used by 'ordinary applications' (may not be servers) and are accessible by any non-privileged user on an OS!**
- **dynamic port numbers (49,152 – 65,535) – neither reserved nor maintained by ICANN - they can be used for any purpose without registration**



Transport Layer: Addressing (cont.)

Port Range Name	Port Number Range	Description
Well-Known (Privileged) Port Numbers	0 to 1,023	These port numbers are managed by IANA and reserved for only the most universal TCP/IP applications. The IANA assigns these port numbers only to protocols that have been standardized using the TCP/IP RFC process, that are in the process of being standardized, or that are likely to be standardized in the future. On most computers, these port numbers are used only by server processes run by system administrators or privileged users. These generally correspond to processes that implement key IP applications, such as Web servers, FTP servers and the like. For this reason, these are sometimes called *system port numbers*.
Registered (User) Port Numbers	1,024 to 49,151	There are many applications that need to use TCP/IP but are not specified in RFCs, or are not so universally used that they warrant a worldwide well-known port number. To ensure that these various applications do not conflict with each other, IANA uses the bulk of the overall port number range for registered port numbers. Anyone who creates a viable TCP/IP server application can request to reserve one of these port numbers, and if approved, the IANA will register that port number and assign it to the application. These port numbers are generally accessible by any user on a system and are therefore sometimes called *user port numbers*.
Private/Dynamic Port Numbers	49,152 to 65,535	These ports are neither reserved nor maintained by IANA. They can be used for any purpose without registration, so they are appropriate for a private protocol used only by a particular organization

http://www.tcpipguide.com/free/t_TCPIPApplicationAssignmentsandServerPortNumberRang-2.htm

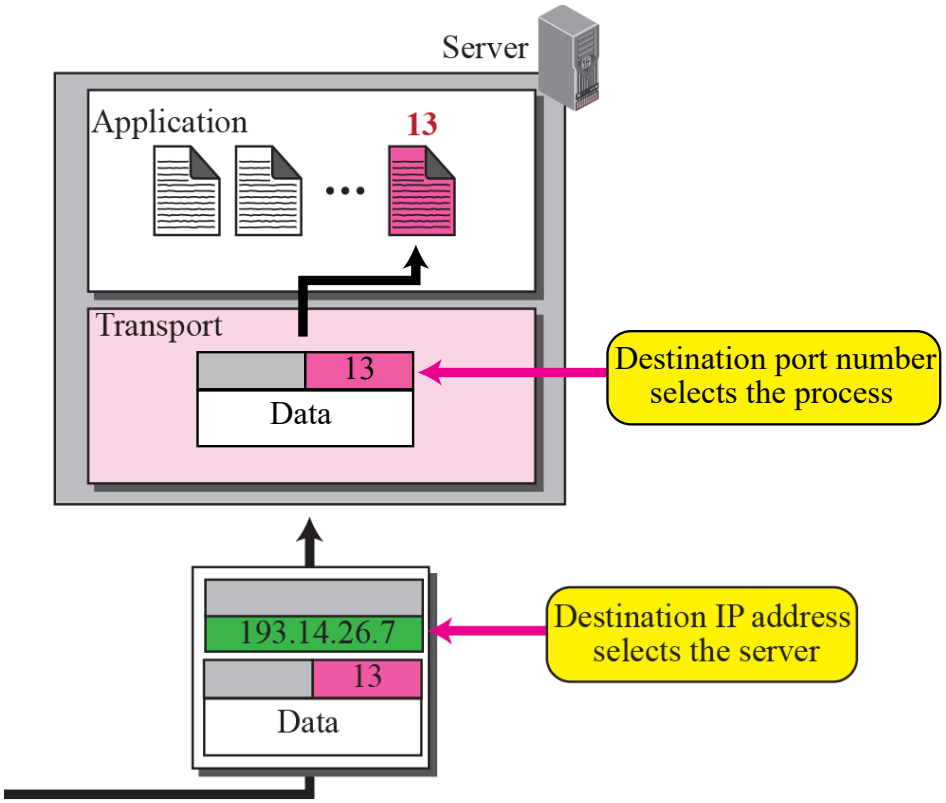
Transport Layer: Addressing (cont.)

6889–6890	Yes	Yes	BitTorrent part of full range of ports used most often
8530	?	?	Windows Server Update Services over HTTP[269][*further explanation neede*
8531	?	?	Windows Server Update Services over HTTPS[270][*further explanation nee*
8834	?	?	Nessus, a vulnerability scanner – remote XML-RPC web server[27
12043	Yes		*Second Life*, used for LSL HTTPS in-bound[308]
12046	Yes		*Second Life*, used for LSL HTTP in-bound[308]

https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Transport Layer: Addressing (cont.)

Example **[multiplexing and demultiplexing]**



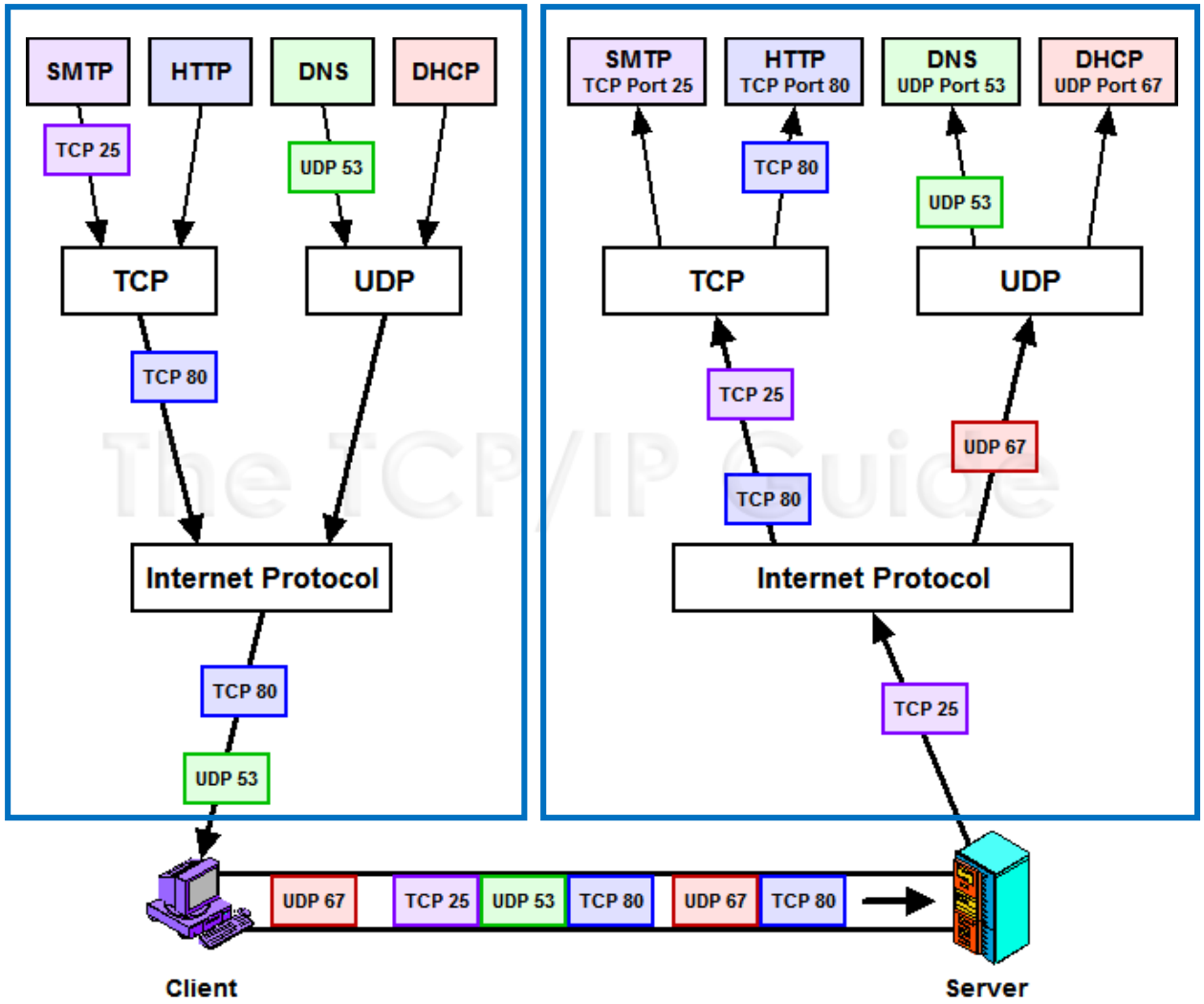
The destination IP address defines the host among the different hosts in the world.
The port number defines one of the processes on that particular host.

Port-addressing mechanism enables application multiplexing and demultiplexing.

Transport Layer: Addressing (cont.)

Multiplexing: required when an entity accepts items from more than one source.

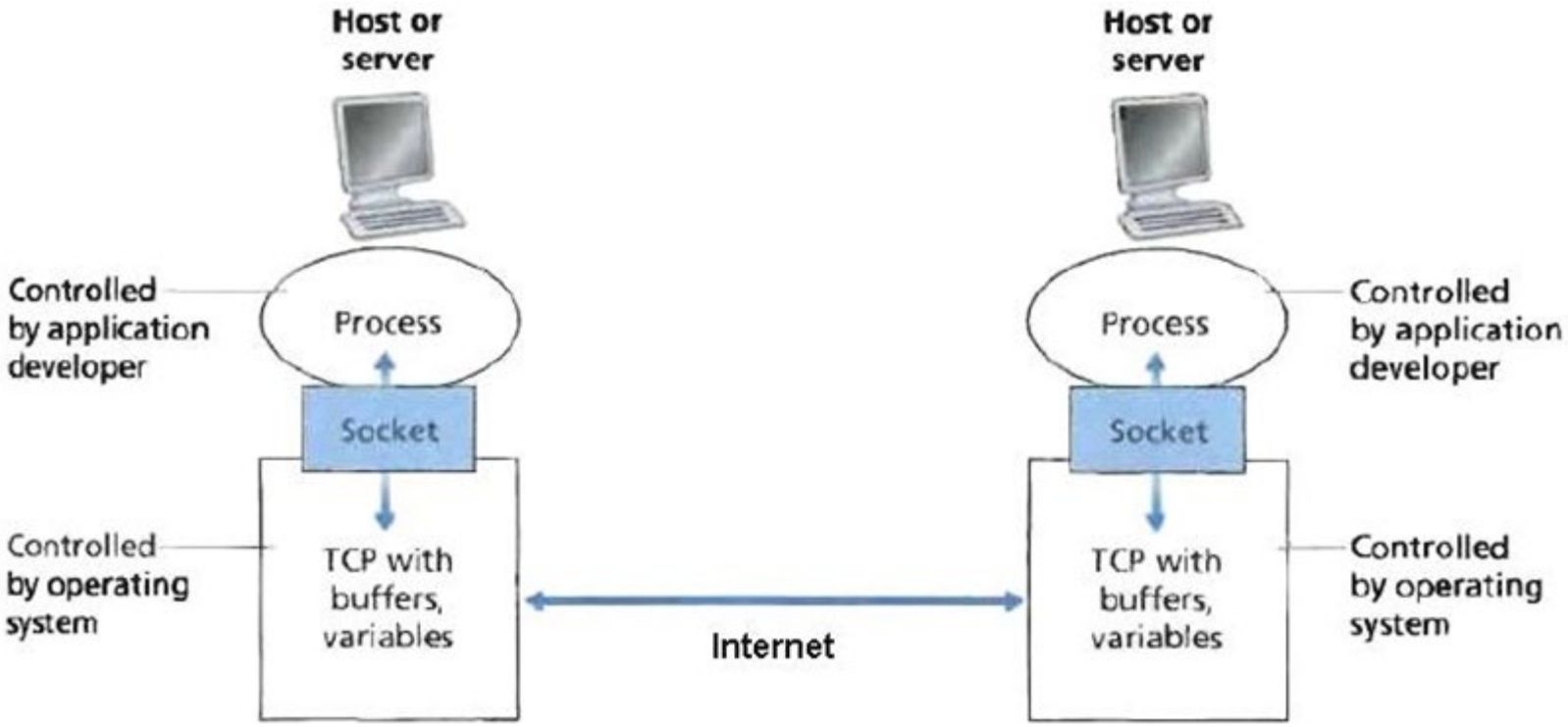
Demultiplexing: required when an entity delivers items to more than one destination.



Transport Layer: Addressing (cont.)

Socket – **logical end point (software object / interface) for receiving and sending data from/to an application / process**

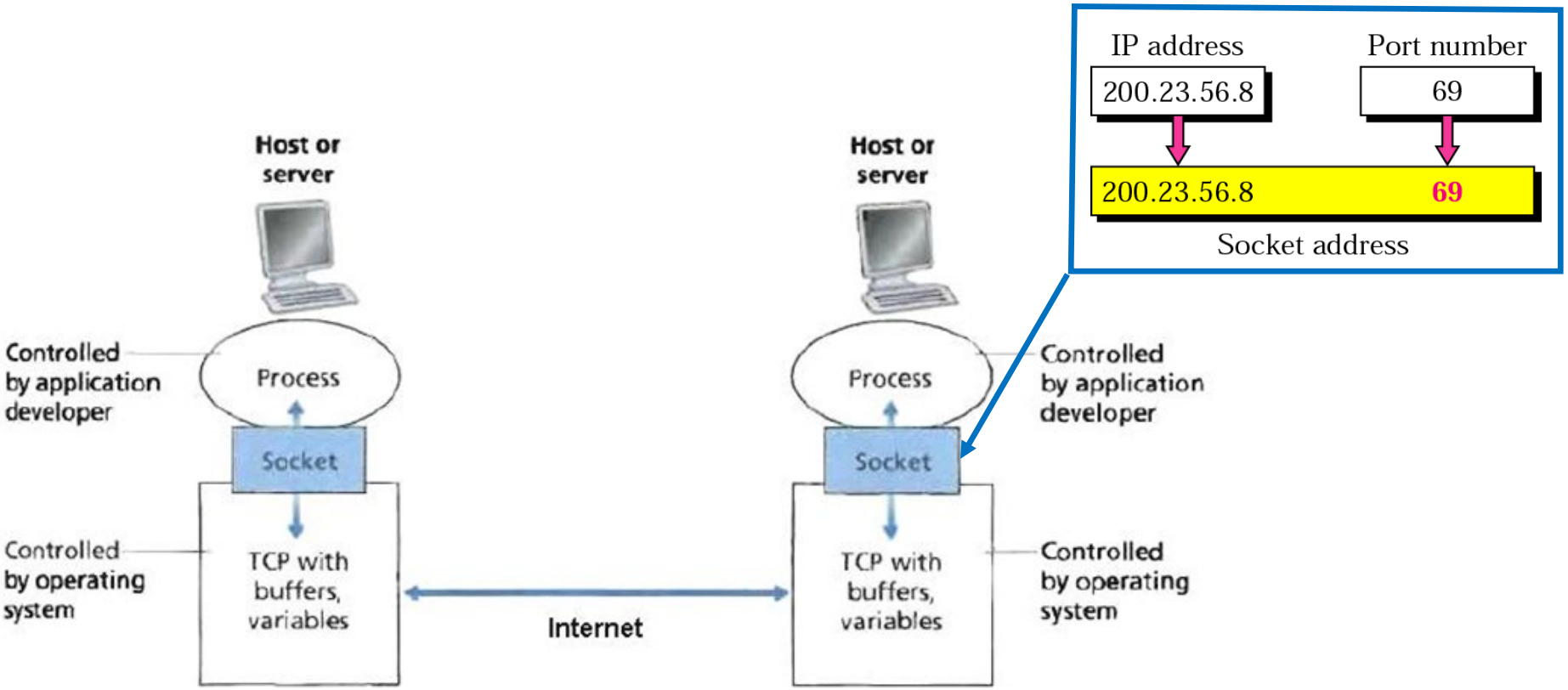
- **"door" between the application and transport layer (TCP or UDP) module**



Transport Layer: Addressing (cont.)

Socket Address – **combination of an IP address and a port number**

- **overall identifier of a TCP/IP application process on a device**
- **socket numbers are unique throughout the Internet**

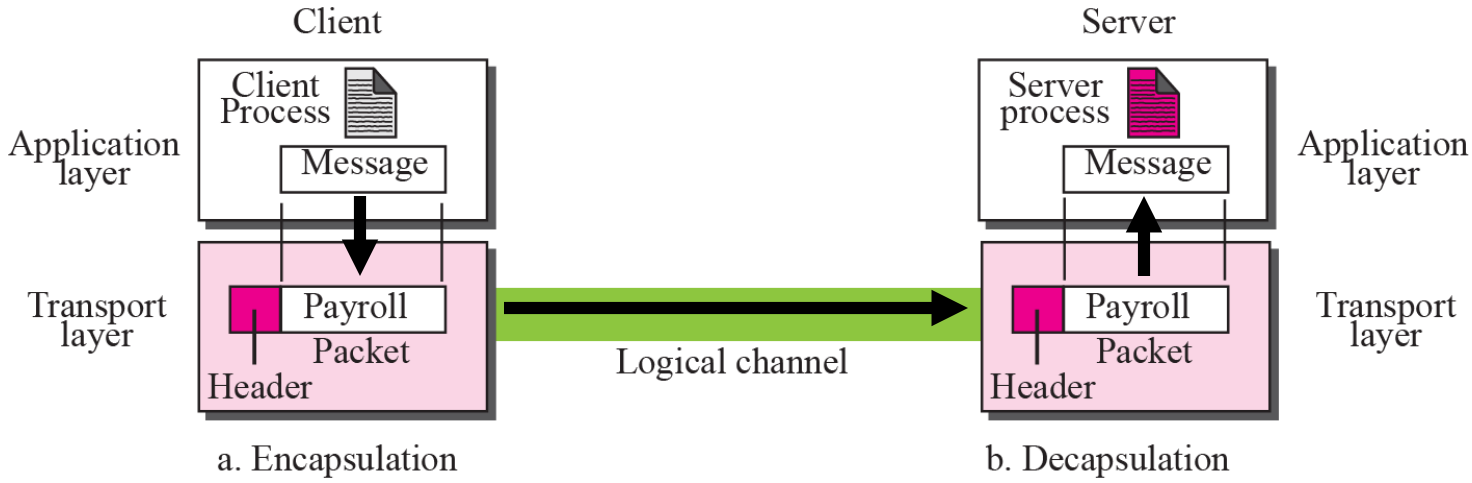


Transport Layer: Packetization

(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

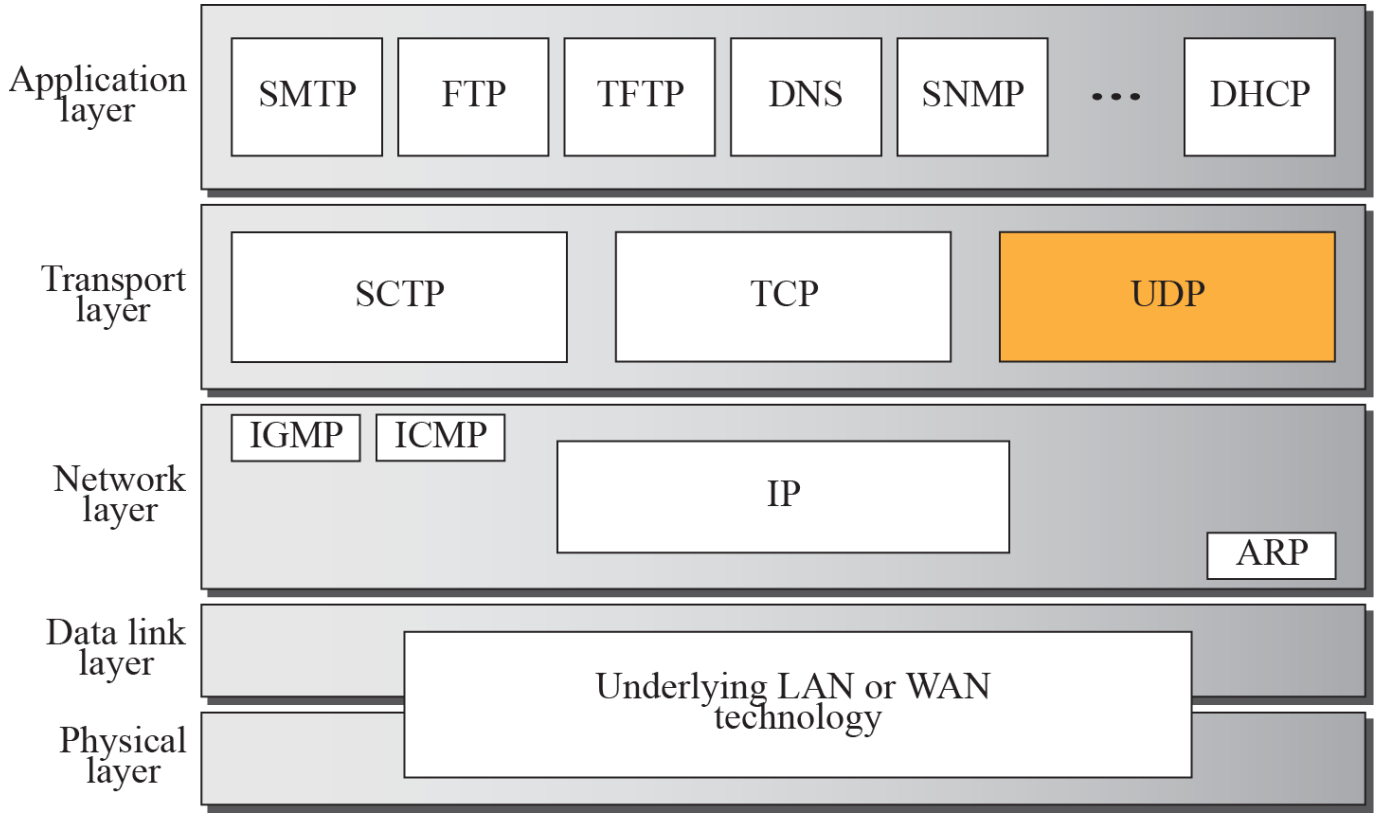
– **transport layer in TCP/IP networks provide two types of process-to-process packet delivery:**

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.

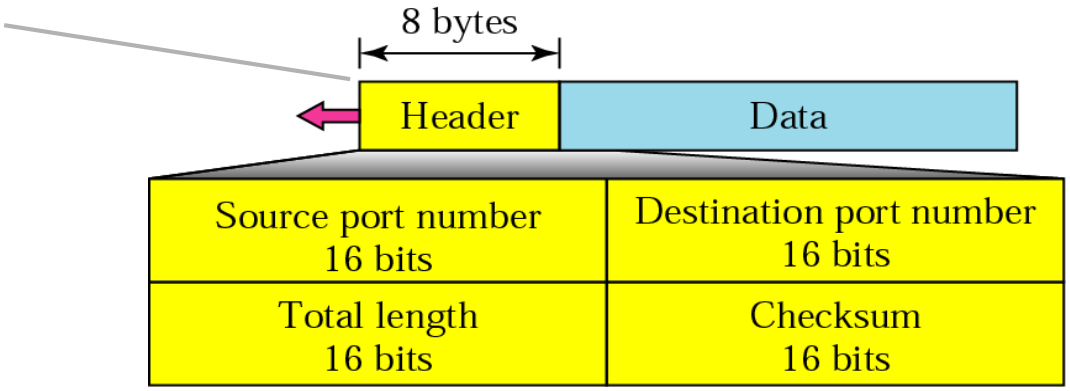
Connectionless Transport: UDP



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, header + data

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

Checksum – used to detect errors over entire UDP datagram (header + data)

- calculation of checksum is optional! - if it is not calculated, the field is filled with 0s
- 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 reliable data transfer**

2) **UDP is suitable for applications that require simple timely request-response communication with little concern for flow and error control**
   - **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 packet loss 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 ☹**

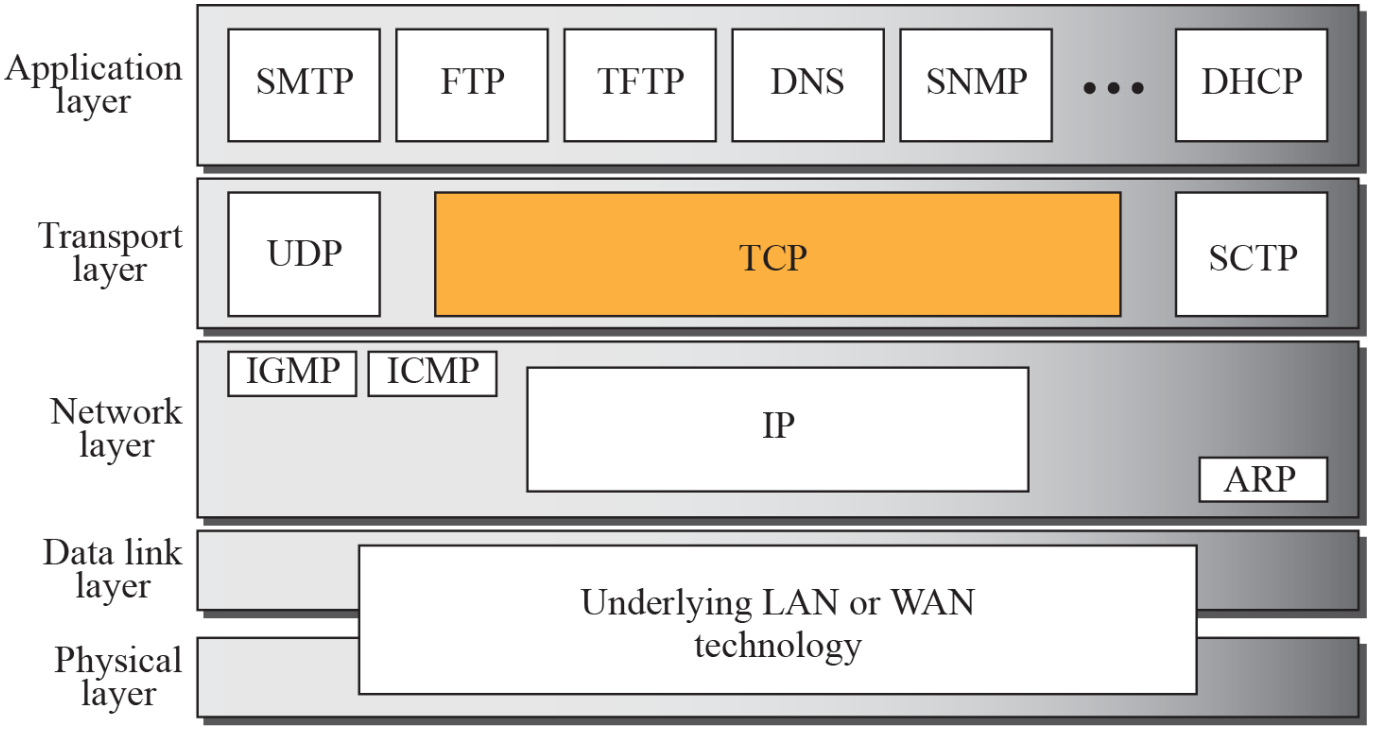
4) **UDP is suitable for multicast and broadcast applications**
   - **TCP supports only point-to-point, i.e. unicast, applications**

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]**

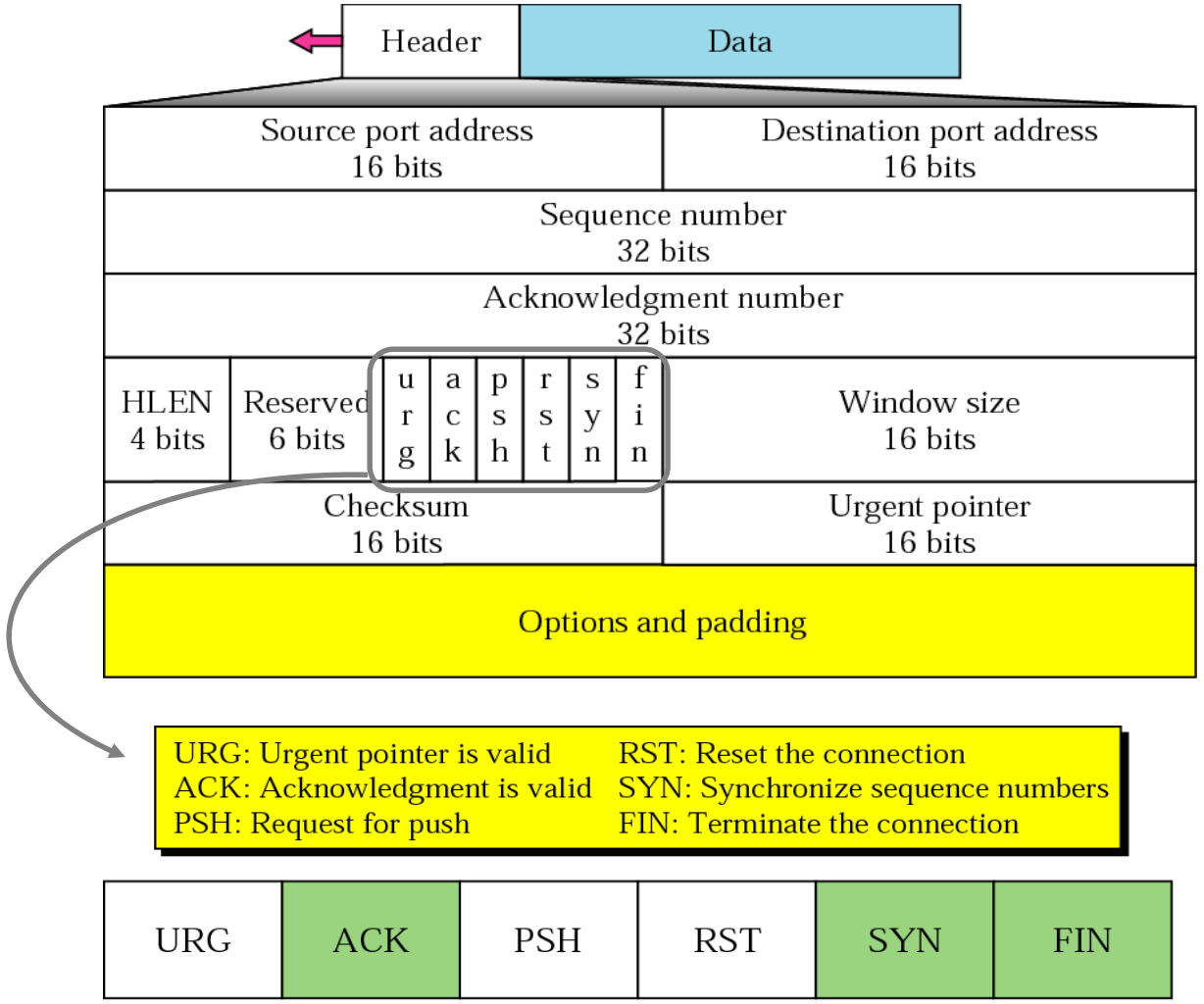
Port	Application Layer Protocol	Description
7	Echo	Echoes a received datagram back to the sender
11	Users	Active users
13	Daytime	Returns the date and the time
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
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"**



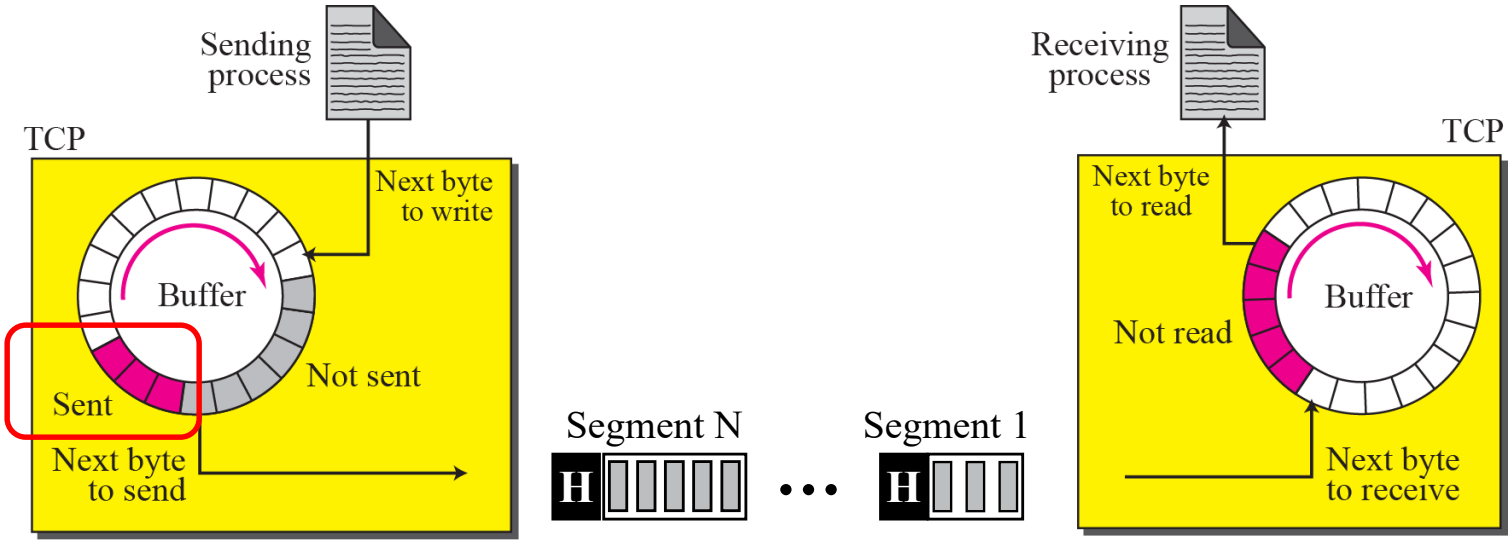
Connection-Oriented Transport: TCP (cont.)

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



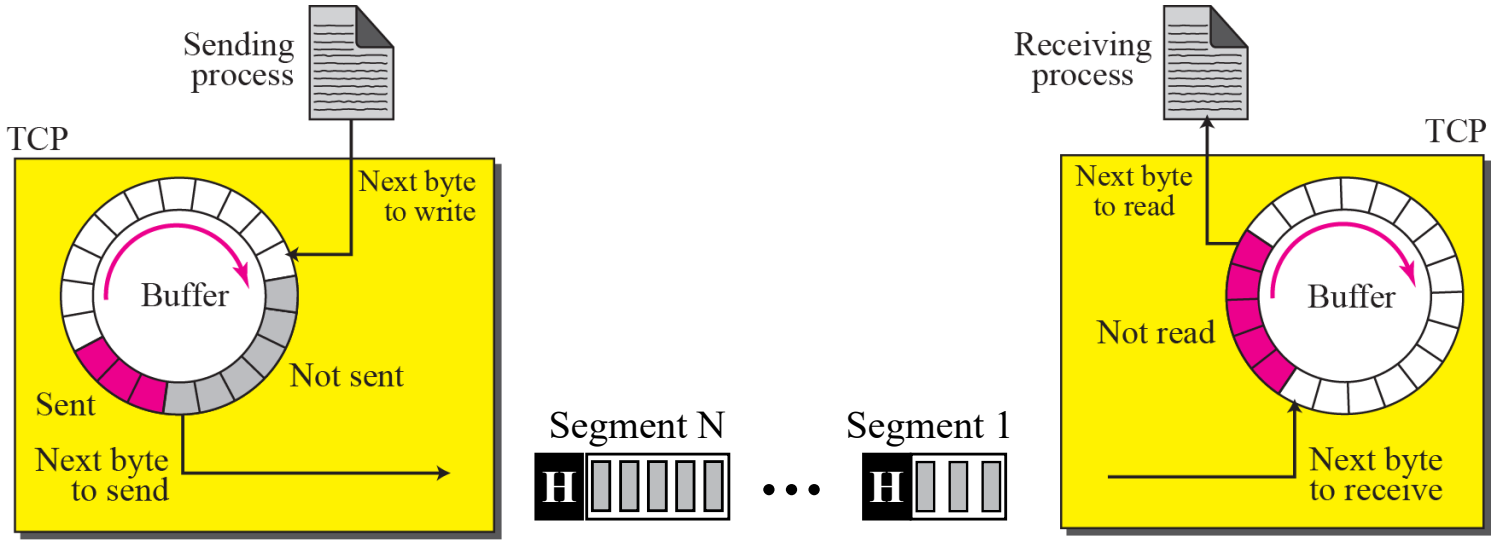
Connection-Oriented Transport: TCP (cont.)

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

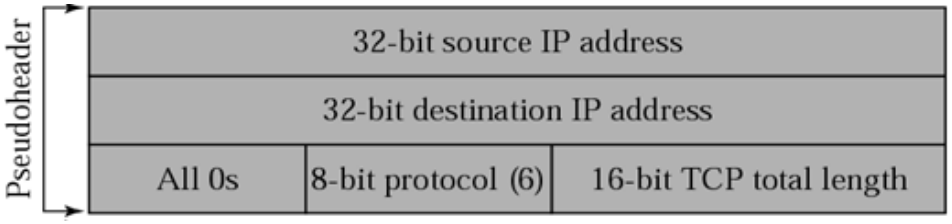
Window Size – 16-bit field – defines the # of bytes, beginning with sequence number indicated in the acknowledgment field that receiver is willing to accept



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

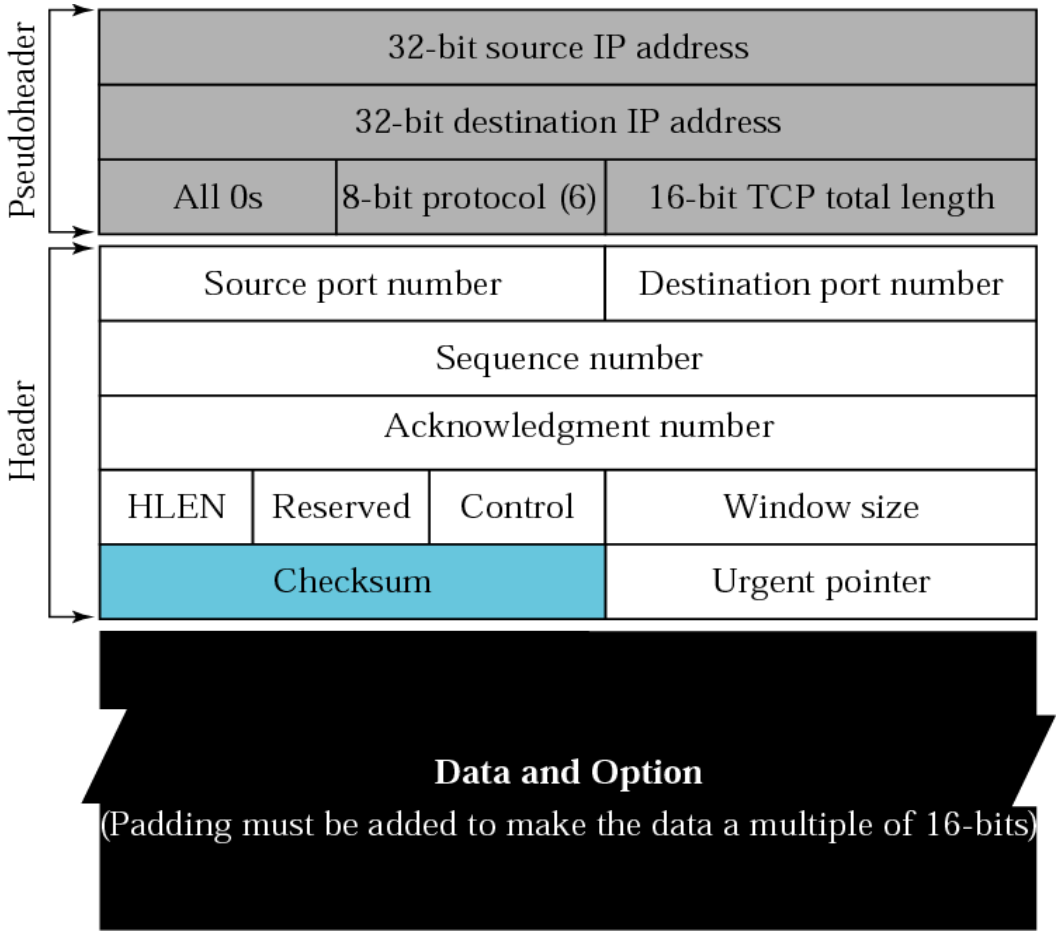


Urgent Pointer – 16-bit field – valid only if the urgent flag is set – contains the sequence # of the last byte in a sequence of urgent data

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]**

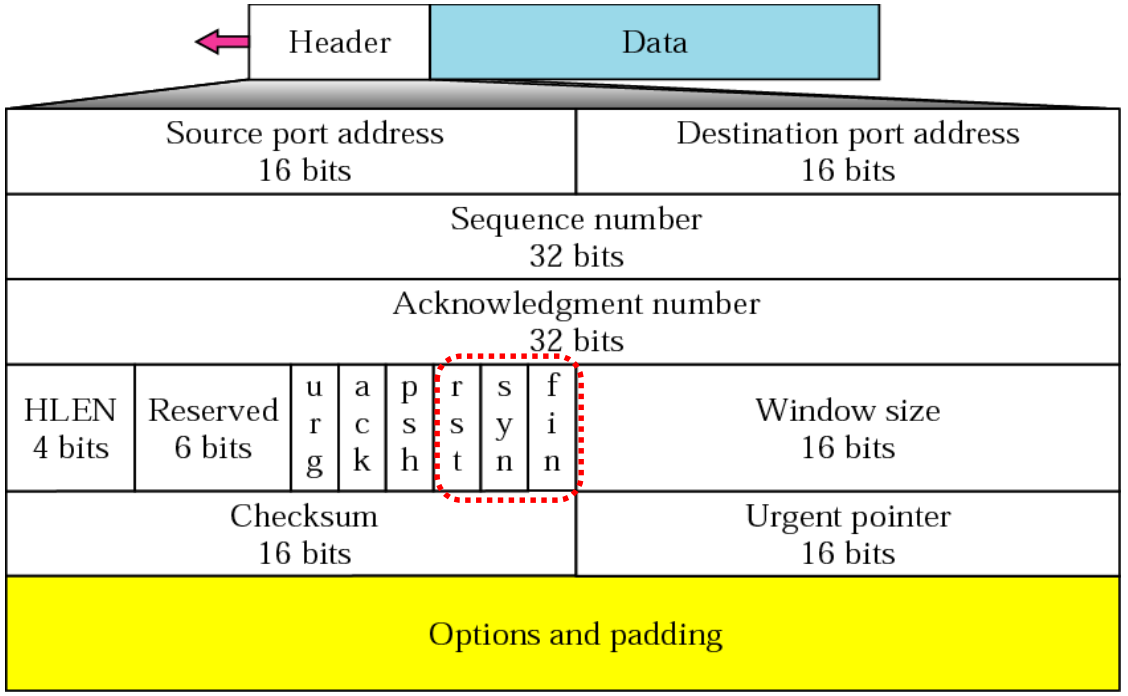


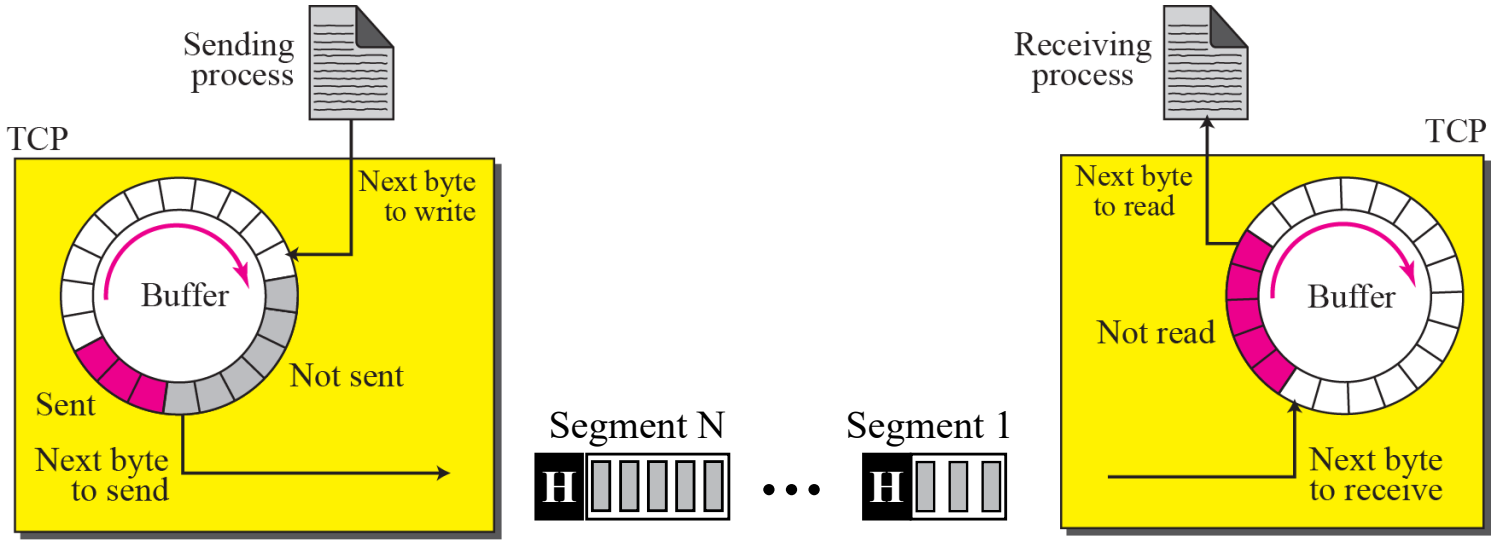
The inclusion of the checksum in TCP is mandatory!
(unlike in case of UDP)

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 aborting 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.


Header
Data
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


Sending process
TCP
Next byte to write
Buffer
Not sent
Sent
Next byte to send
Segment N
Segment 1
H
H
Receiving process
TCP
Next byte to read
Buffer
Not read
Next byte to receive