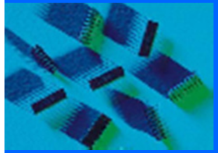


Lecture 11
Process-to-process Delivery UDP and TCP

Textbook: Ch. 23, 24

Main Topics

A. **Transport Layer**
- Transport-Layer Services
- Port Number and Socket Address

B. **Transport Layer Protocol**
- Transport Layer in TCP/IP (24.1)

C. **UDP (24.2)**
- User Datagram Format
- UDP Applications and Examples

D. **TCP (24.3)**
- TCP Connection Establishment
- TCP Data Transfer
- Sequence Number and Acknowledgement
- Retransmission and Timeout

A. Transport Layer

- The transport layer is located between the application layer and the network layer.
- It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host.
- Communication is provided using a logical connection.

Idea behind this logical connection.

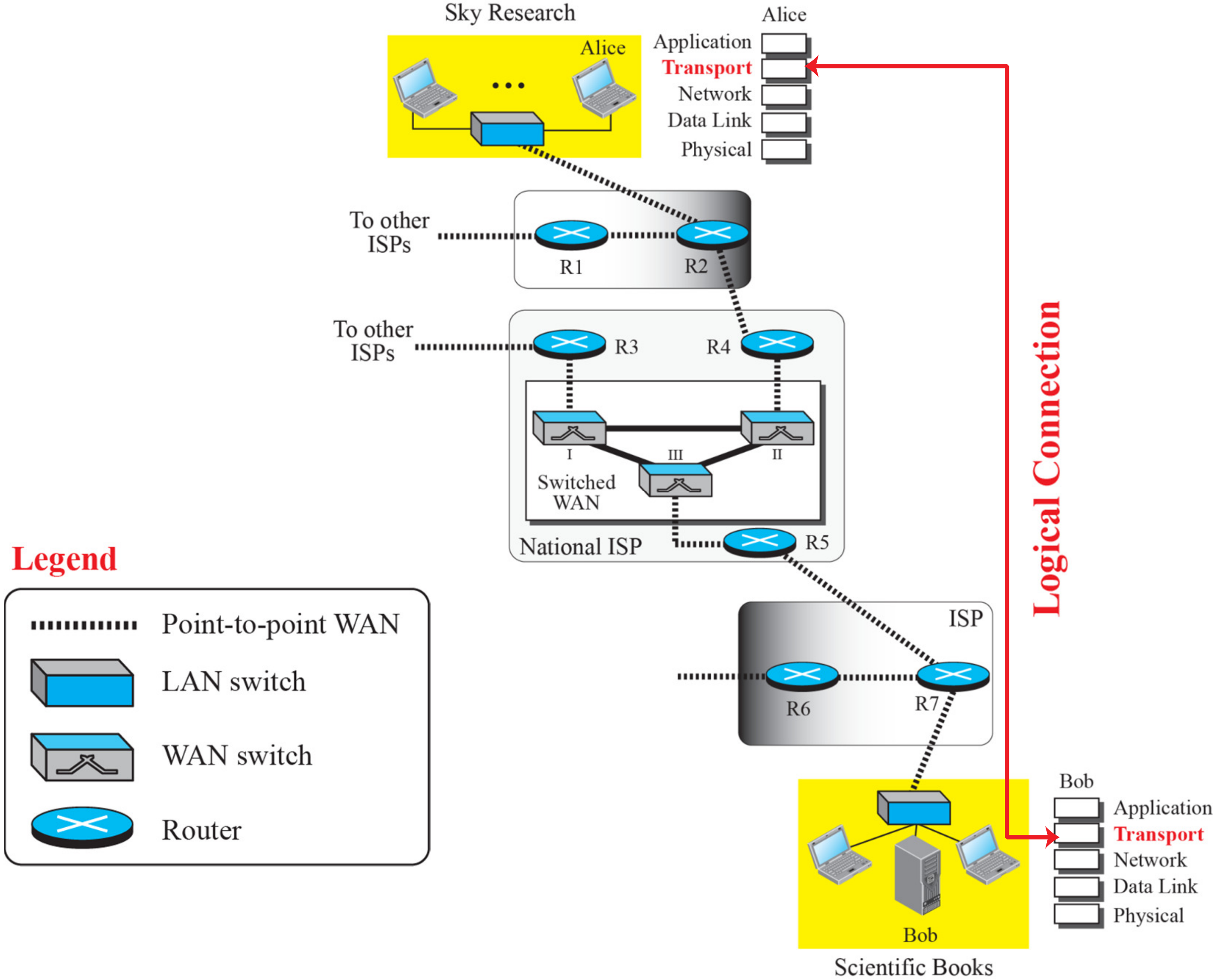
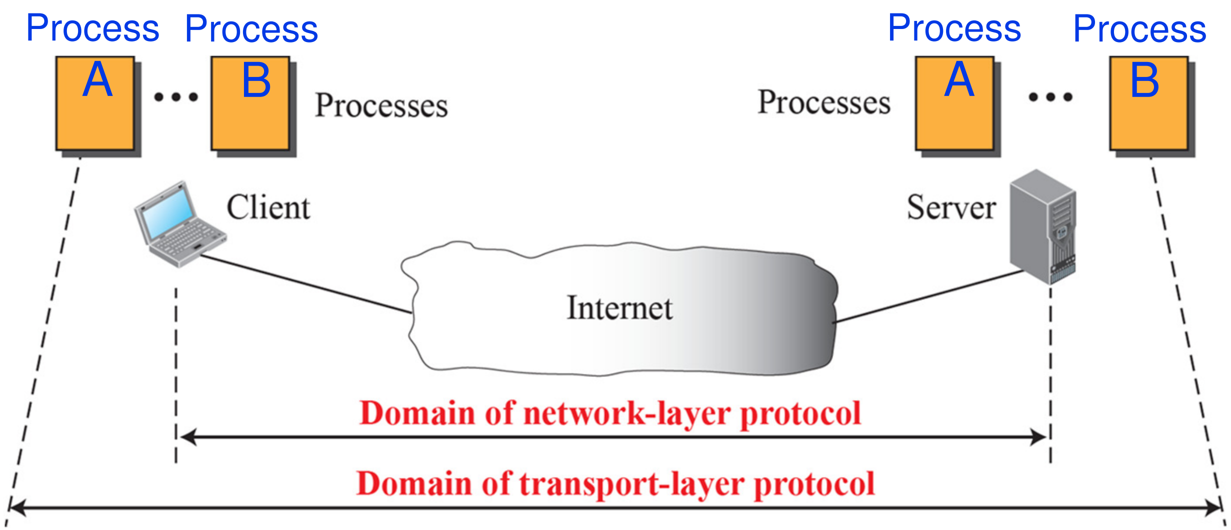


Figure 23.1: Logical connection at the transport layer

Transport-Layer Services

- The transport layer
  - is responsible for providing services to the *application layer*; and
  - receives services from the *network layer*.
- Process-to-process Communication



Addressing : Port Numbers

- A computer can run several server programs and/or several client programs at the same time.
- To identify a process, **a port number** is used.
  - E.g. A service called "Daytime" uses 13 as the port number.

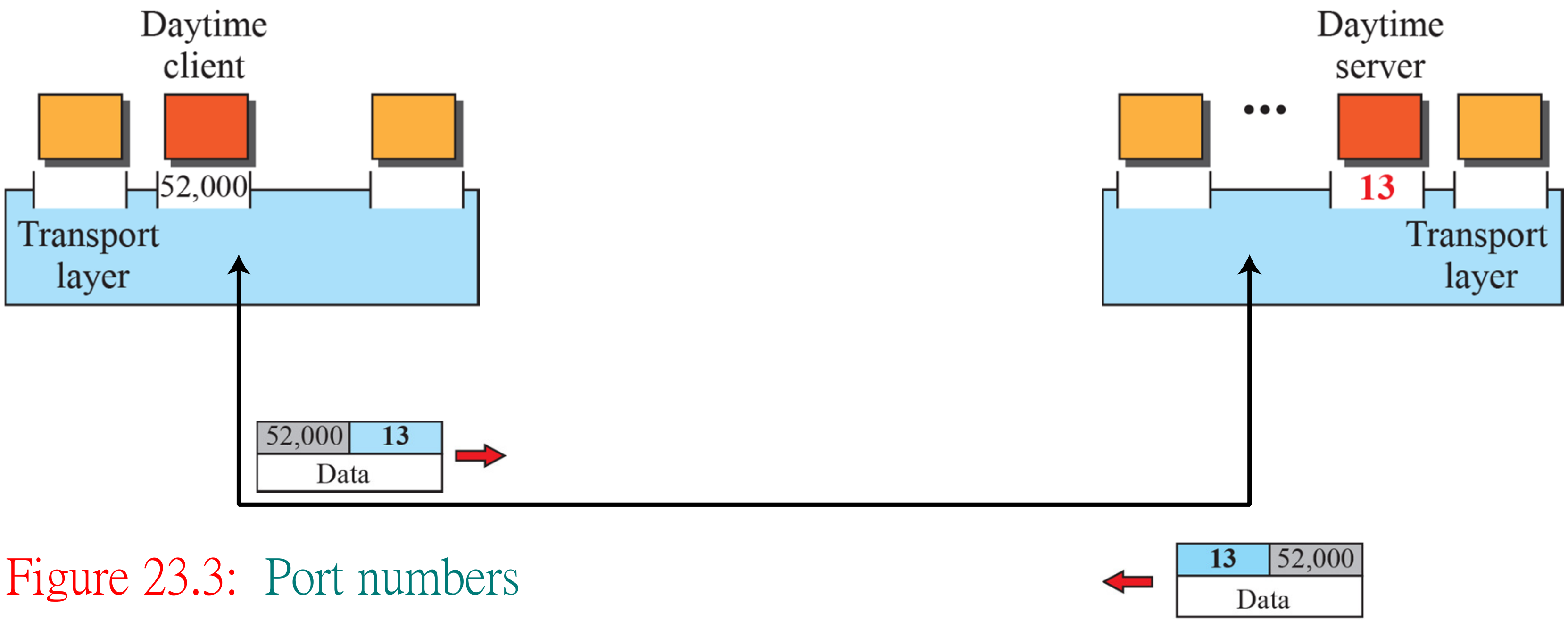
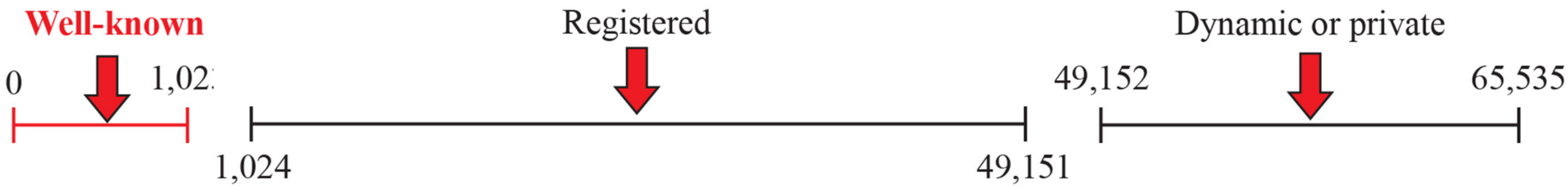


Figure 23.3: Port numbers

Port Number

- In TCP/IP protocol suite, the port numbers are integers between 0 and 65,535 (16 bits).
  - *Well-known ports:* 0 to 1023.
    - *E.g. 23: Telnet remote login, 80: HTTP*
  - *Registered ports:* 1024 to 49151.
    - *IANA maintains the official list.*
  - *Dynamic or private ports: 49152 to 65535.*
    - *One common use is for temporary ports.*



- The Full list can be found in Service Name and Transport Protocol Port Number Registry
  - http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Socket Address

- The combination of an IP address and a port number is called a socket address.

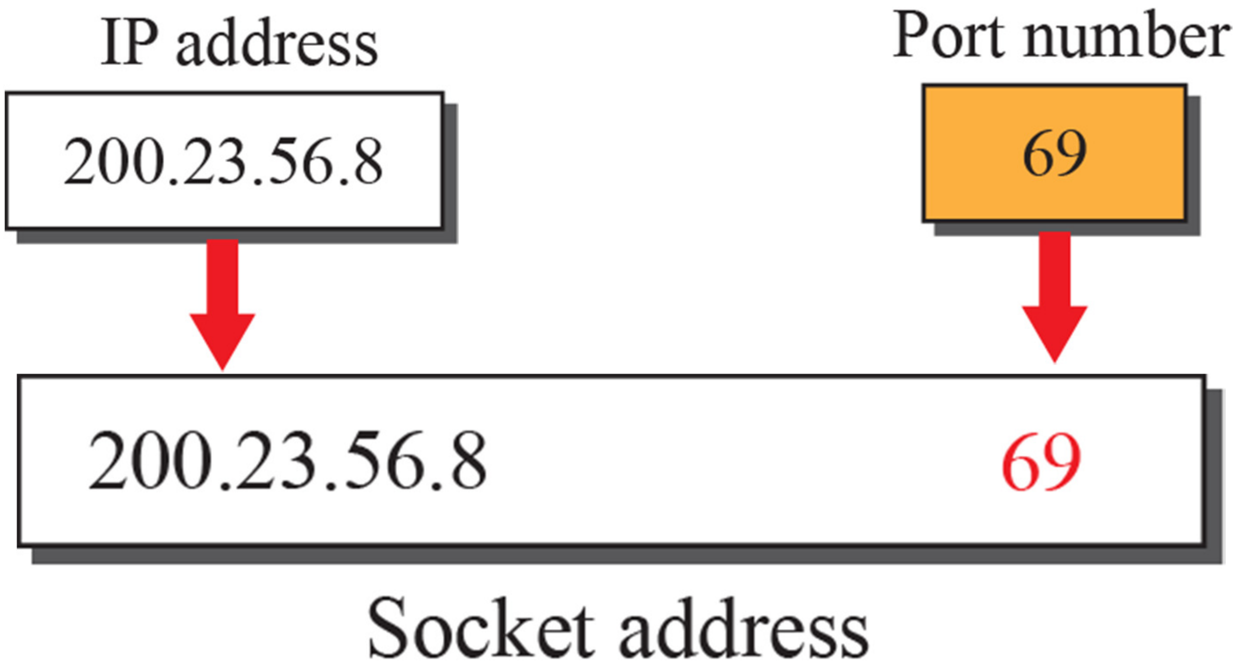


Figure 23.6: Socket address

B. Transport Layer in TCP/IP

- TCP/IP is a **set of protocols**, or protocol suite, that defines how all transmissions are exchanged across the Internet
- TCP/IP is a **five-layer** protocol: physical, data link, network, transport and application
- ***Transport layer***: (2 protocols/services)
  - Transmission Control Protocol (**TCP**) - data unit is called TCP segment
  - User Datagram Protocol (**UDP**)
  - ***Network layer***: Internet Protocol (**IP**)

Transport-layer protocols in the TCP/IP protocol suite

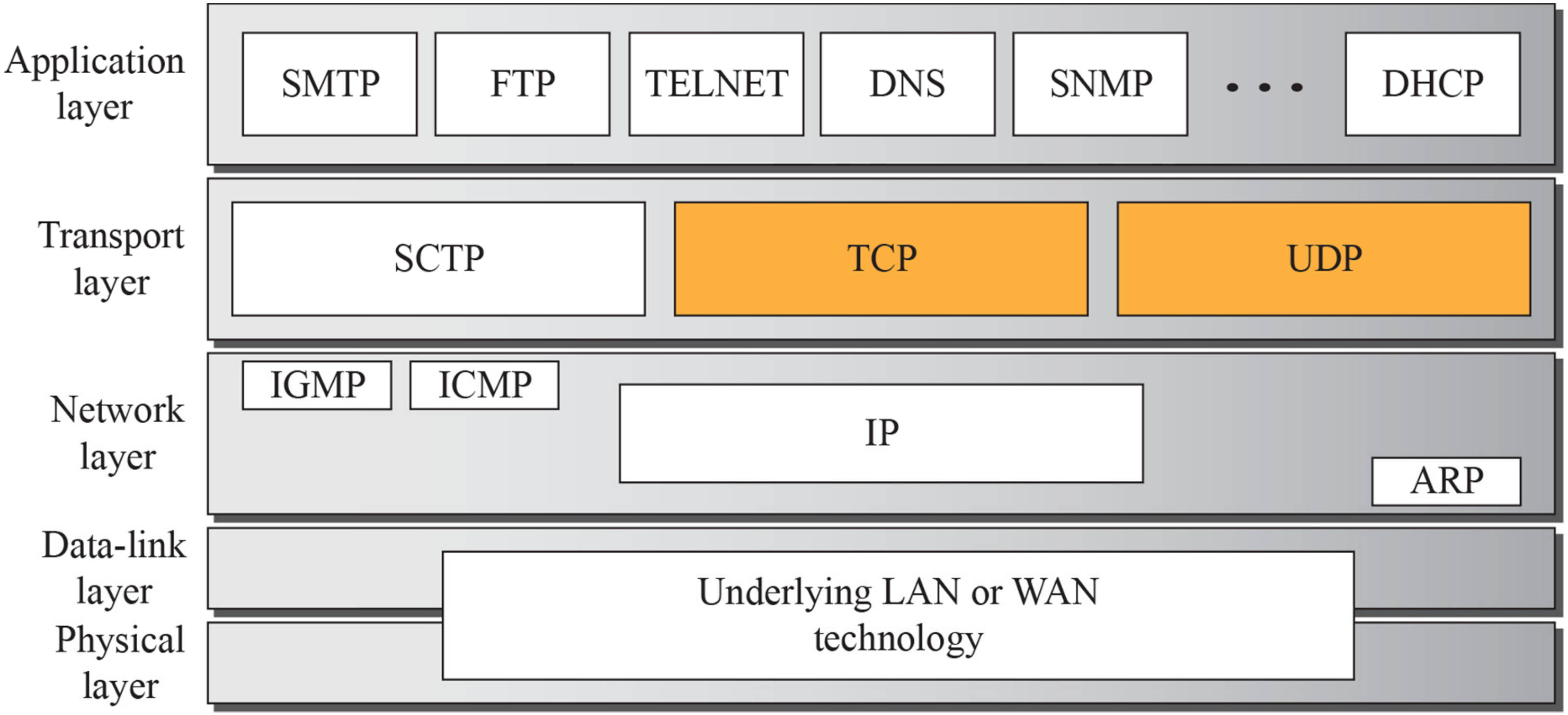


Figure 24.1: Position of transport-layer protocols in the TCP/IP protocol suite

Some well-known UDP and TCP Protocols

Port	*Protocol*	*UDP*	*TCP*	*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, 21	FTP		√	File Transfer Protocol
23	TELNET		√	Terminal Network
25	SMTP		√	Simple Mail Transfer Protocol
53	DNS	√	√	Domain Name Service
67	DHCP	√	√	Dynamic Host Configuration Protocol
69	TFTP	√		Trivial File Transfer Protocol
80	HTTP		√	Hypertext Transfer Protocol
111	RPC	√	√	Remote Procedure Call
123	NTP	√	√	Network Time Protocol
161, 162	SNMP		√	Simple Network Management Protocol

Table 24.1: Some well-known ports used with UDP and TCP

C. User Datagram Protocol (UDP)

- **The User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol.**
- UDP packets, called user datagrams, have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).
- **If UDP is so powerless, why would a process want to use it?**
  - **UDP is a very simple protocol using a minimum of overhead.**

UDP Format

- **The first two fields define the source and destination port numbers.**
- **The third field defines the total length of the user datagram, header plus data.**
  - **The 16 bits can define a total length of 0 to 65,535 bytes.**
    - **However, the total length needs to be less because a UDP user datagram is stored in an IP datagram with the total length of 65,535 bytes.**

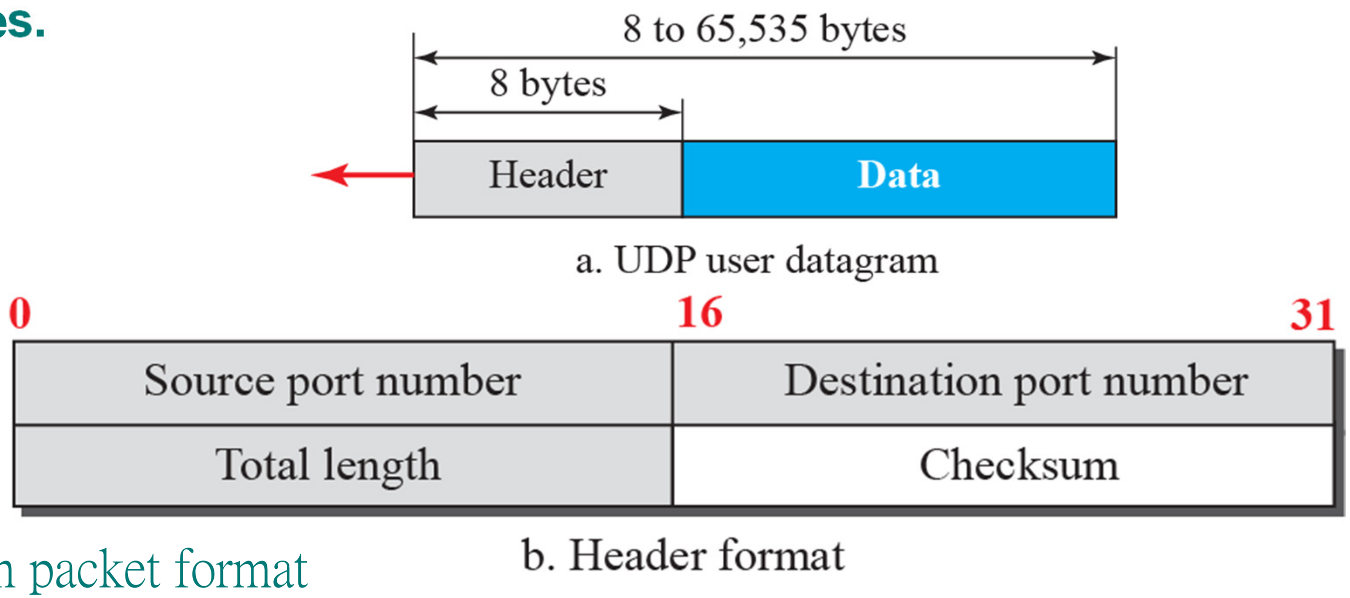


Figure 24.2: User datagram packet format

Example 24.1

The following is the contents of a UDP header in hexadecimal format.

CB84000D001C001C

a. What is the source port number?
b. What is the destination port number?
c. What is the total length of the user datagram?
d. What is the length of the data?
e. Is the packet directed from a client to a server or vice versa?
f. What is the client process?

Example 24.1

Solution

a. The source port number is the first four hexadecimal digits $(CB84)_{16}$ or 52100

b. The destination port number is the second four hexadecimal digits $(000D)_{16}$ or 13.

c. The third four hexadecimal digits $(001C)_{16}$ define the length of the whole UDP packet as 28 bytes.

d. The length of the data is the length of the whole packet minus the length of the header, or $28 - 8 = 20$ bytes.

e. Since the destination port number is 13 (well-known port), the packet is from the client to the server.

f. The client process is the Daytime (see Table 3.1).

Example 24. 3 - DNS

- **Domain Name Service (DNS)**
  - **A client-server application**
  - **uses the services of** UDP
    - Because a client needs to 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.

- Quick reference for DNS:
  - http://www.youtube.com/watch?v=ZBi8GCxk7NQ
  - http://www.youtube.com/watch?v=2ZUxoi7YNgs

Real-time interactive application

- Audio and video are divided into frames and sent one after another (Such as Skype).
- If the transport layer is supposed to resend a corrupted or lost frame,
  - The synchronizing of the whole transmission may be lost.
    - The viewer suddenly sees a blank screen and needs to wait until the second transmission arrives.
    - This is not tolerable.
  - Each small part of the screen is sent using one single user datagram
    - The receiving UDP can easily ignore the corrupted or lost packet and deliver the rest to the application program.
    - That part of the screen is blank for a very short period of time, which most viewers do not even notice.

Very large text file?

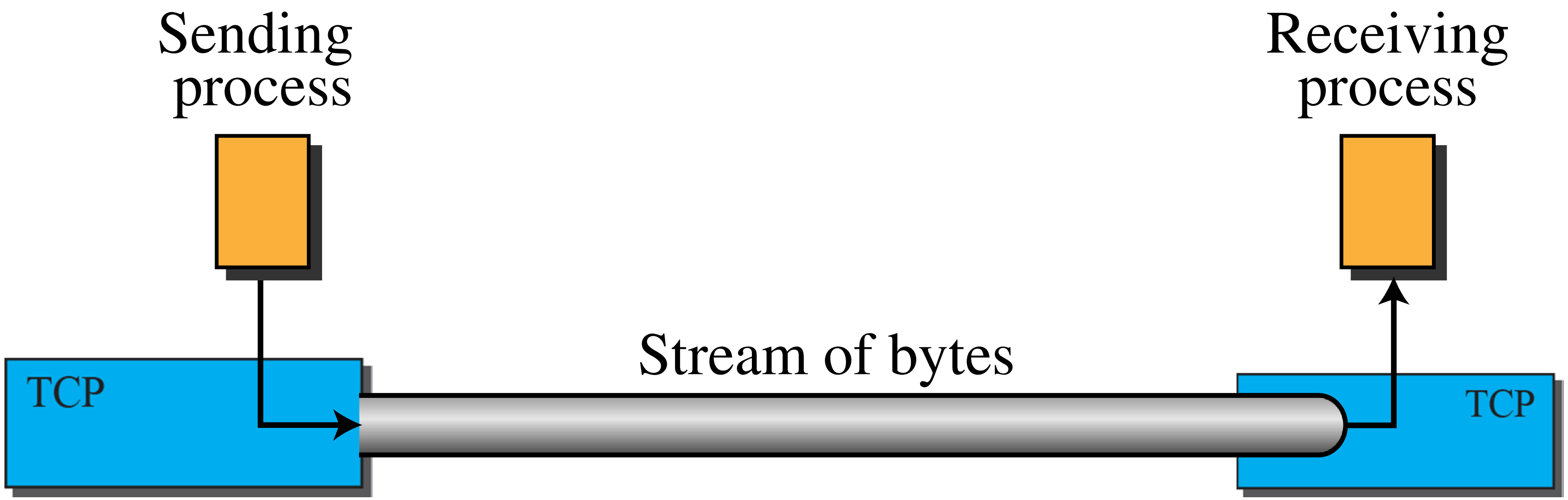
- How about downloading a very large text file from the Internet?
  - We definitely need to use a transport layer that provides reliable service.
  - We don’t want part of the file to be missing or corrupted.
  - The delay created between the deliveries of the parts is not an overriding concern for us; we wait until the whole file is composed before looking at it.
  - In this case, UDP is not a suitable transport layer.
- Then, use TCP service.

D. Transmission Control Protocol (TCP)

- Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol.
- TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.
- TCP uses a combination of Go-back N (GBN) and Selective Repeat (SR) protocols to provide reliability.

TCP Services

- TCP provides process-to-process communication using port numbers.
- It is a stream-oriented protocol.
- TCP creates an environment in which the two processes seem to be connected by an "imaginary tube" that carries their bytes across the network.



TCP Segment

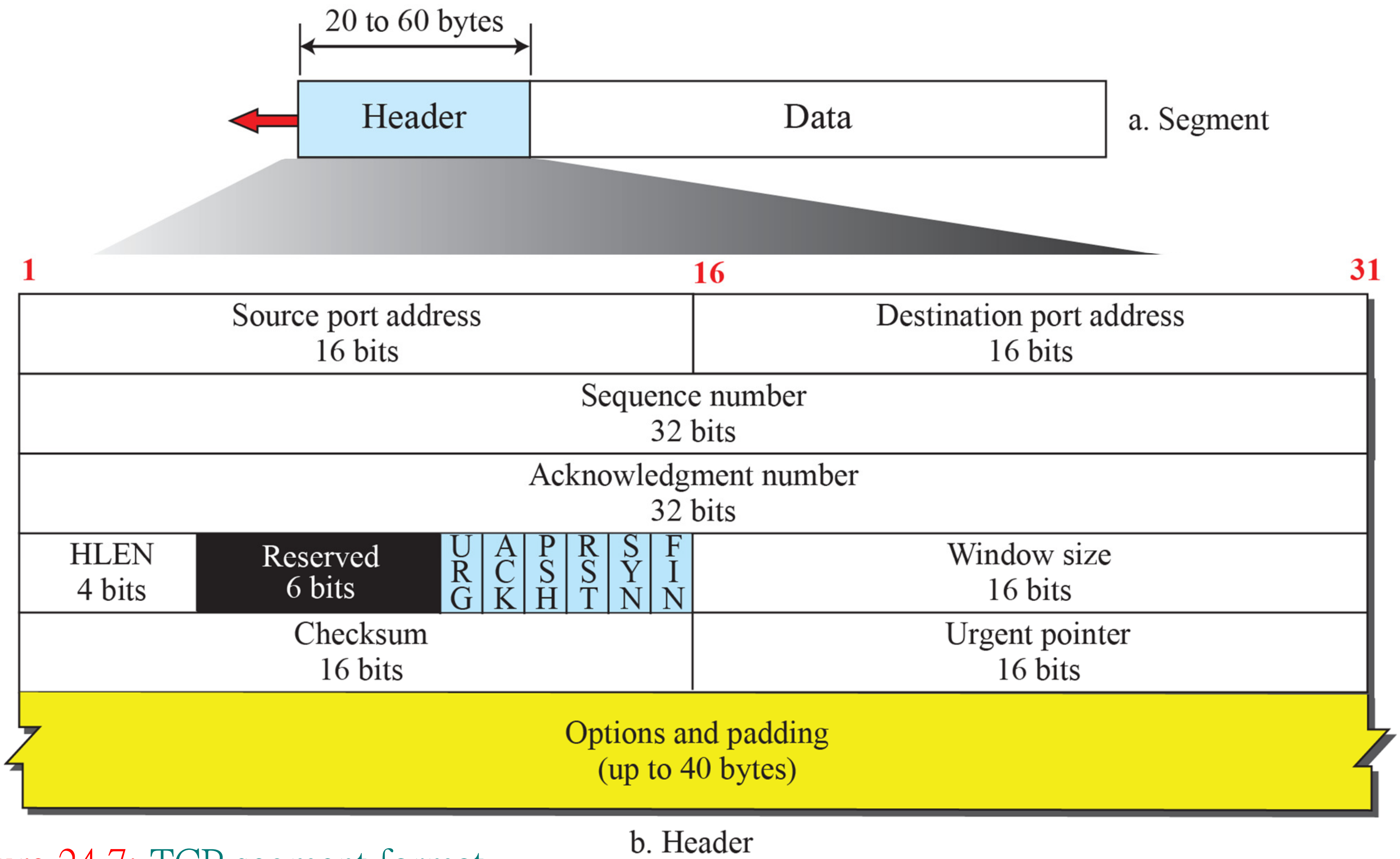


Figure 24.7: TCP segment format

Header Length and Control Field

- Header Length (HLEN)
  - Indicates the number of 4-byte words in the TCP header.
  - Header length is between 20 to 60 bytes.
  - So, the field is 5 (5x4=20) to 15.
- Control Field (6 bits)
  - ACK – indicate that the value carried in the acknowledgement field is valid
  - RST, SYN, FIN – for connection setup and teardown (skip details)
  - PSH – indicate that the receiver should pass the data to the upper layer immediately
  - URG – indicate that this segment contains urgent data; the location of the last byte is indicated by the *urgent data pointer field*

TCP Connection

- TCP is connection-oriented.
- All of the segments belonging to a message are then sent over this logical path.
  - Using a single logical pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.
- How TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented?
  - The point is that a TCP connection is logical, 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.

Connection establishment using three-way handshaking

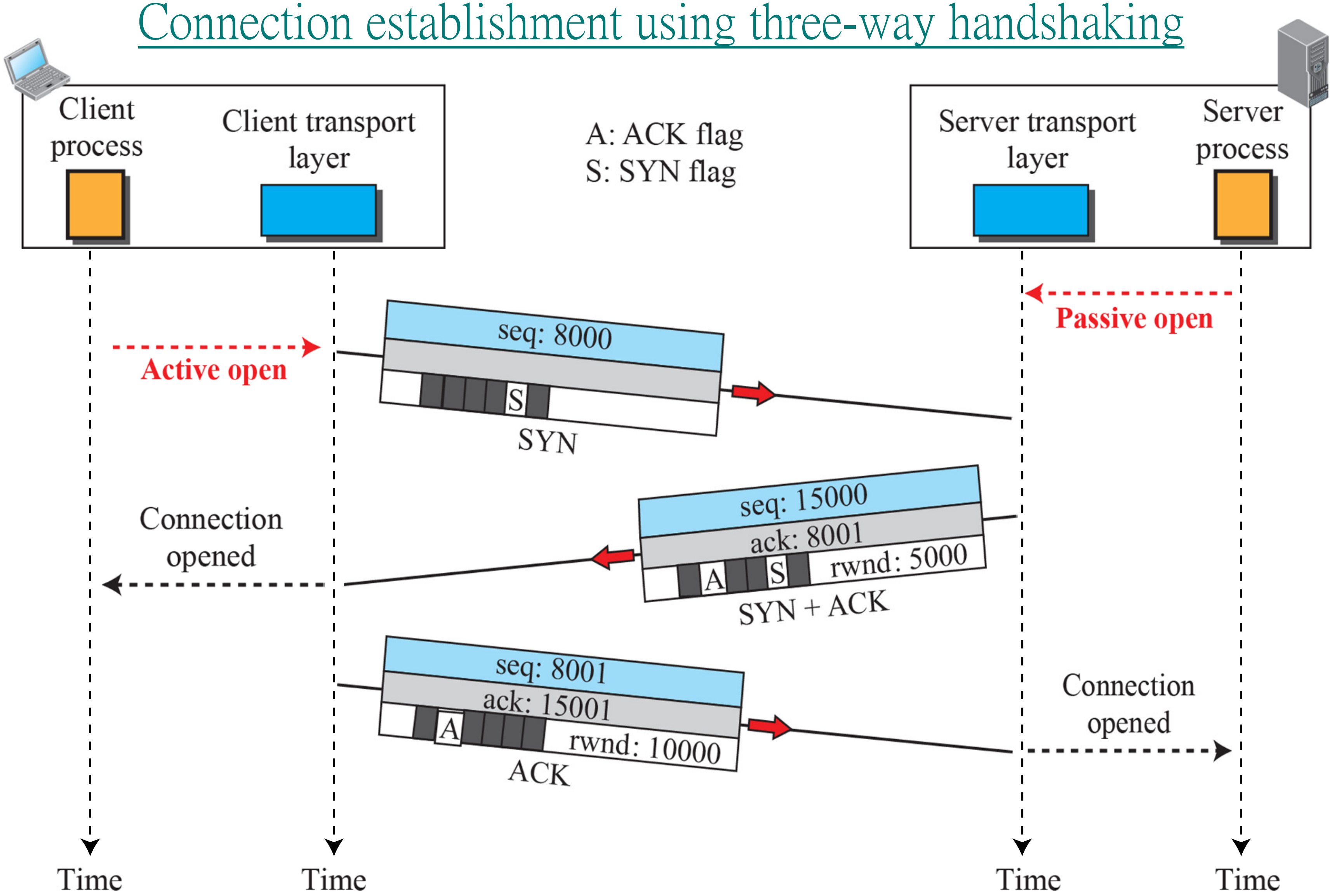


Figure 24.10: Connection establishment using three-way handshaking

Connection Establishment

- The server program tells its TCP that it is ready to accept a connection. The request is called a **passive open**.
- When a client wants to connect with the server, it issues a request for **active open**, and TCP will start a **Three- Way Handshaking**:
  - Step One: Client sends a SYN segment (with Clients' sequence number).
  - Step Two: Server sends a SYN + ACK segment (with servers' sequence number).
  - Step Three: Client sends an ACK segment.

Data Transfer

- In the example:
  - After a connection is established, the client sends 2,000 bytes of data in two segments.
  - The server then sends 2,000 bytes in one segment.
  - The client sends one more segment.
  - The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there is no more data to be sent.
    - PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.

TCP Data transfer

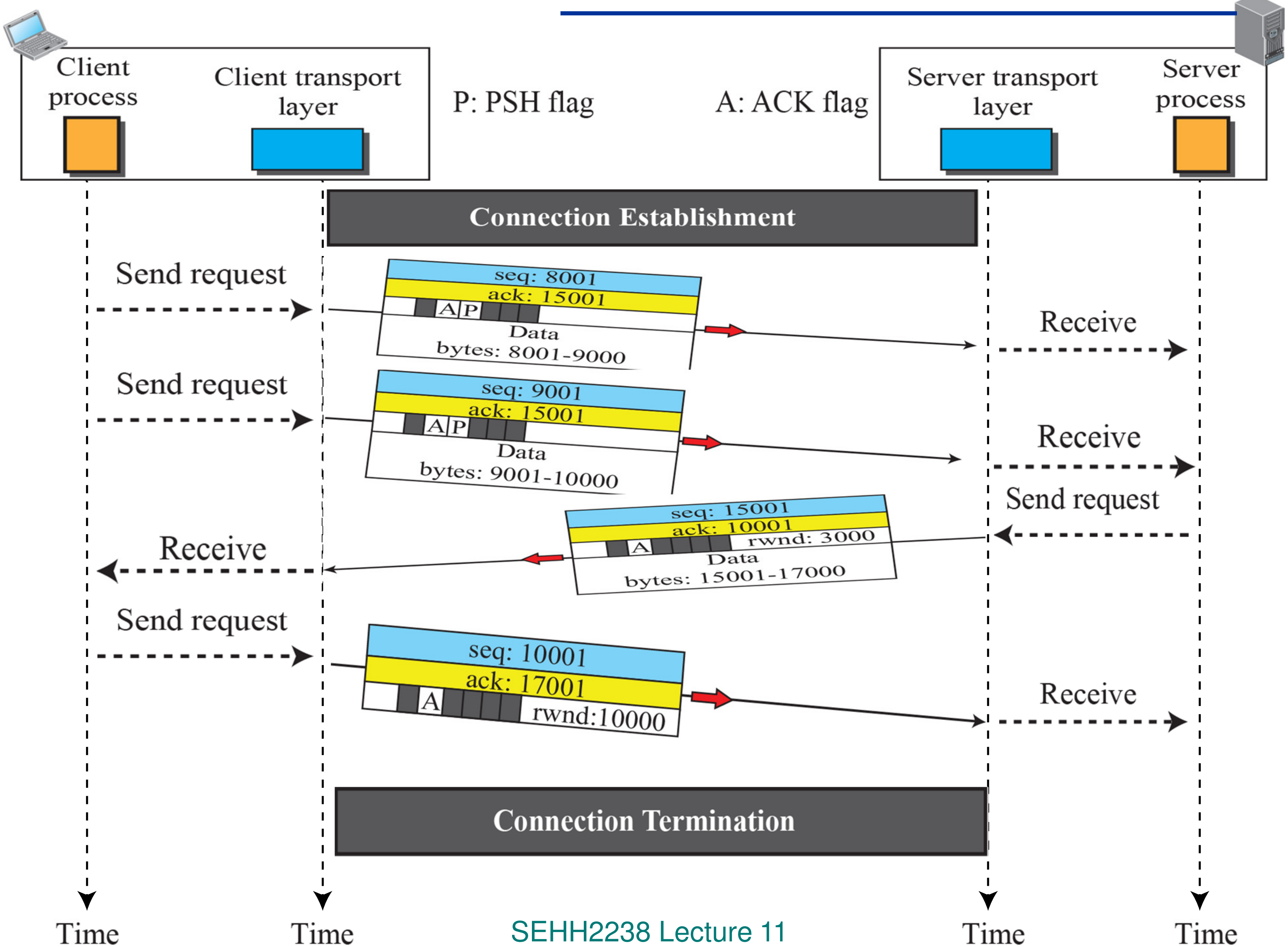


Figure 24.11: Data transfer

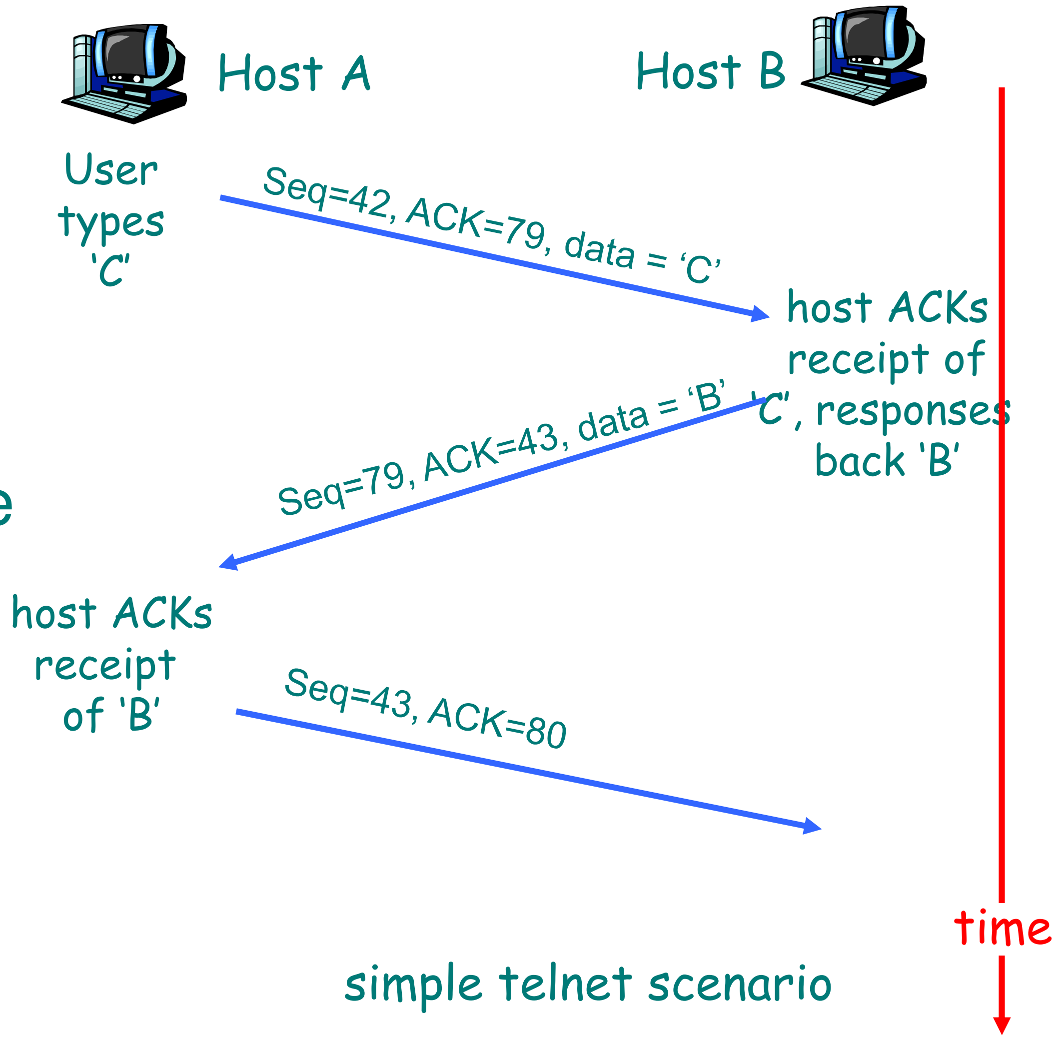
TCP seq. #'s and ACKs

Seq. #'s:

- byte stream "number" of first byte in segment's data

ACKs: (TCP is full-duplex)

- seq # of *next byte* expected from other side
- *cumulative* ACK



simple telnet scenario

TCP Round Trip Time and Timeout

Q: how to set TCP timeout value?

- *longer than RTT*
  - but RTT varies
- too short: premature timeout
  - unnecessary retransmissions
- too long: slow reaction to segment loss

Q: how to estimate RTT?

- **`SampleRTT`**: measured time from segment transmission until ACK receipt
  - ignore retransmissions
- **`SampleRTT`** will vary, want estimated RTT "smoother"
  - average several recent measurements, not just current **`SampleRTT`**

TCP Round Trip Time and Timeout

EstimatedRTT =
(1- α)*EstimatedRTT + α*SampleRTT

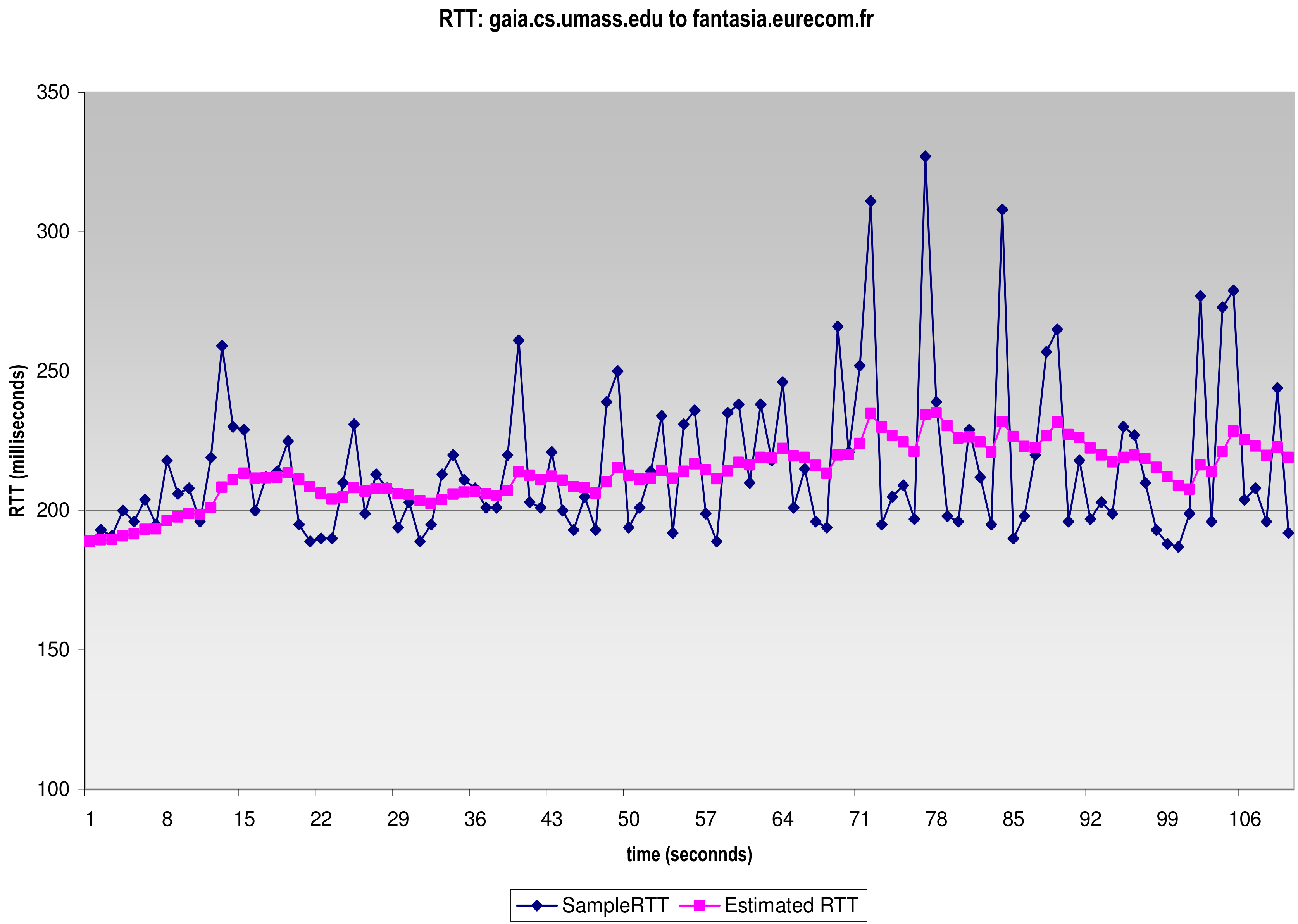
- Exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: α = 0.125

Example of Estimating RTT(E_RTT)

- T0 2:00 send data
  - 2:15 ACK back
  - SampleRTT=15
- T1 2:15 send data
  - 2:28 ACK back
  - SampleRTT=13
- T2 2:30 send data
  - 2:46 ACK back
  - SampleRTT=16
- T3 2:50 send data
  - 3:00 ACK back
  - SampleRTT=10
- T4 3:00 send data

- Assume α = 0.1 and
- initial E_RTT0 = 10 minutes
- **EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT**
- E_RTT1= 0.9x10 + 0.1 x 15 = 10.5
- E_RTT2= 0.9x10.5 + 0.1 x 13 = 10.75
- E_RTT3= 0.9x10.75 + 0.1 x 16 = 11.275
- E_RTT4= 0.9x11.275 + 0.1 x10= 11.1475

Example RTT estimation:



TCP Round Trip Time and Timeout

Setting the timeout

- **EstimtedRTT** plus "safety margin"
  - large variation in **EstimatedRTT** -> larger safety margin
- first estimate of how much SampleRTT deviates from EstimatedRTT:

```
DevRTT = (1-β)*DevRTT +
               β*|SampleRTT-EstimatedRTT|
(typically, β = 0.25)
```

Then set timeout interval:

```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

Example of Setting Timeout(TO)

- T0 2:00 send data
  - 2:15 ACK back
  - SampleRTT=15
- T1 2:15 send data
  - 2:28 ACK back
  - SampleRTT=13
- T2 2:30 send data
  - 2:46 ACK back
  - SampleRTT=16
- T3 2:50 send data
  - 3:00 ACK back
  - SampleRTT=10
- T4 3:00 send data

- Assume β = 0.2 and
- initial D_RTT0 = 1 minutes
- `DevRTT = (1-β)*DevRTT + β*|SampleRTT-EstimatedRTT|`
- `TimeoutInterval = EstimatedRTT+4*DevRTT`
- D_RTT1= 0.8x1 + 0.2 x |15-10| = 1.8

  TO1= 10.5 + 4x1.8 = 17.7
- D_RTT2= 0.8x1.8 +0.2 x|13-10.5|= 1.94

  TO2= 10.75 + 4x1.94 = 18.51
- D_RTT3=0.8x1.94 +0.2x|16-10.75|=2.60

  TO3= 11.275 + 4x2.6 = 21.675
- D_RTT4= 0.8x2.60+0.2 x|10-11.275|= 2.335

  TO4= 11.1475 + 4x2.335 = 20.4875

TCP reliable data transfer

- TCP creates reliable data transfer service on top of IP's unreliable service
- Pipelined segments
- Cumulative acks and Selective Repeat.
- TCP uses single retransmission timer (to reduce overhead)
- Retransmissions are triggered by:
  - Timeout events
  - Duplicate acks
- Initially consider simplified TCP sender:
  - ignore duplicate acks
  - ignore flow control, congestion control

Summary

- Transport Layer provides process-to-process logical connection
- Socket address = IP address + port number
- The User Datagram Protocol (UDP) is a connectionless, unreliable but simple.
- TCP creates reliable data transfer service
  - TCP Round Trip Time and Timeout

References

- Video on Comparing UDP and TCP
  - http://www.youtube.com/watch?v=Vdc8TCESIg8
- Revision Quiz
  - http://highered.mheducation.com/sites/0073376221/student_view0/chapter23/quizzes.html
  - http://highered.mheducation.com/sites/0073376221/student_view0/chapter24/quizzes.html