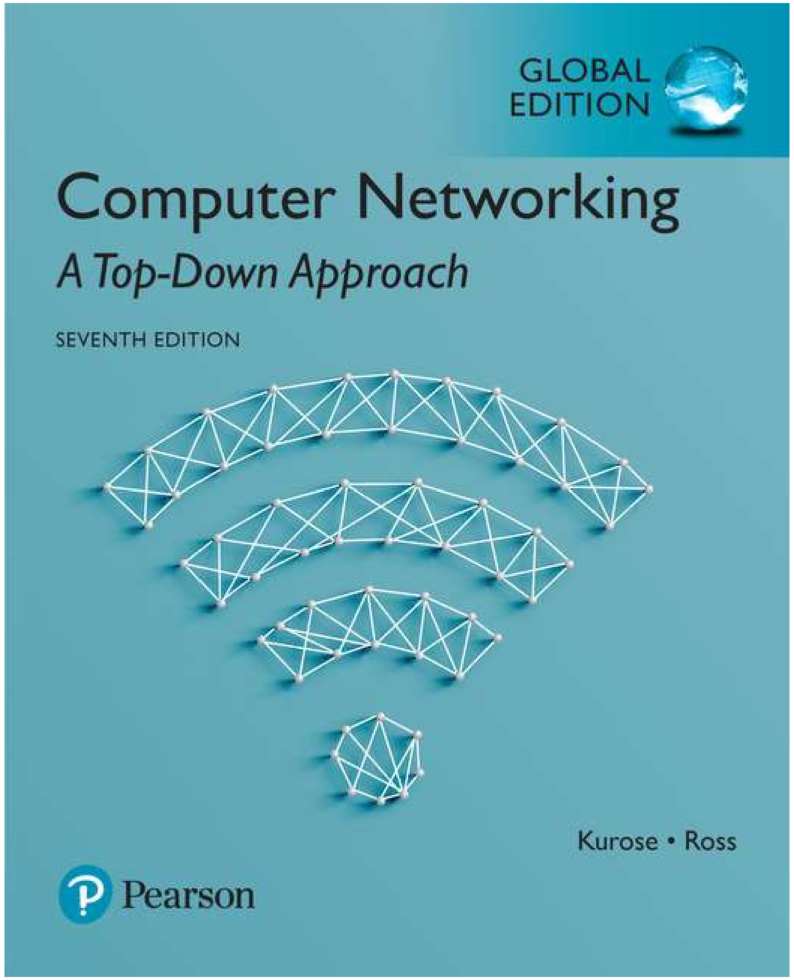
Chapter 3
Transport Layer

Computer Networking: A Top Down Approach

컴퓨터 네트워크
(2019년 1학기)

박승철교수

한국기술교육대학교
컴퓨터공학부



Pre-study Test :

1) 트랜스포트 계층 프로토콜의 최종 정보 전달 대상은 무엇인가?

① Host

② Program

③ Server

④ Process

2) 트랜스포트 계층 프로토콜이 전달하는 정보 단위의 이름은?

① 패킷(packet)

② 세그먼트(segment)

③ 프레임(frame)

④ 메시지(message)

3) 다음 트랜스포트 계층 프로토콜에서 제공하는 서비스가 아닌 것은?

① 오류 제어(error control)

② 다중화(multiplexing)

③ 지연시간 제어(delay control)

④ 혼잡 제어(congestion control)

4) 응용 프로세스가 트랜스포트 프로토콜 서비스를 사용하기 위해 생성하는 자료 구조는?

① Socket

② Port

③ Queue

④ Message box

5) 비연결형(connectionless) 서버 소켓의 식별자는?

① 서버 IP 주소

② 서버 Port 번호

③ 서버 IP 주소 + 서버 Port 번호

④ 클라이언트 IP 주소 + 클라이언트 Port 번호 + 서버 IP 주소 + 서버 Port 번호

6) 연결형(connection-oriented) 서버 소켓의 식별자는?

① 서버 IP 주소

② 서버 Port 번호

③ 서버 IP 주소 + 서버 Port 번호

④ 클라이언트 IP 주소 + 클라이언트 Port 번호 + 서버 IP 주소 + 서버 Port 번호

7) 다음 중 UDP에 대한 설명 중 틀린 것은?

① 비연결형 프로토콜이다.

② Port 번호로 다중화 서비스를 제공한다.

③ 전달 순서를 보장한다.

④ 오류 검출 서비스를 제공한다.

8) 다음 중 UDP의 특징이 아닌 것은?

① 오류 처리 로직이 있지만 간단하다.

② 헤더가 간단하다.

③ 지연시간이 작다.

④ 혼잡제어 로직이 있지만 간단하다.

9) 이진수 0110의 1의 보수값(one's complement)은 얼마인가?

① 0110

② 1001

③ 0110

④ 1111

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

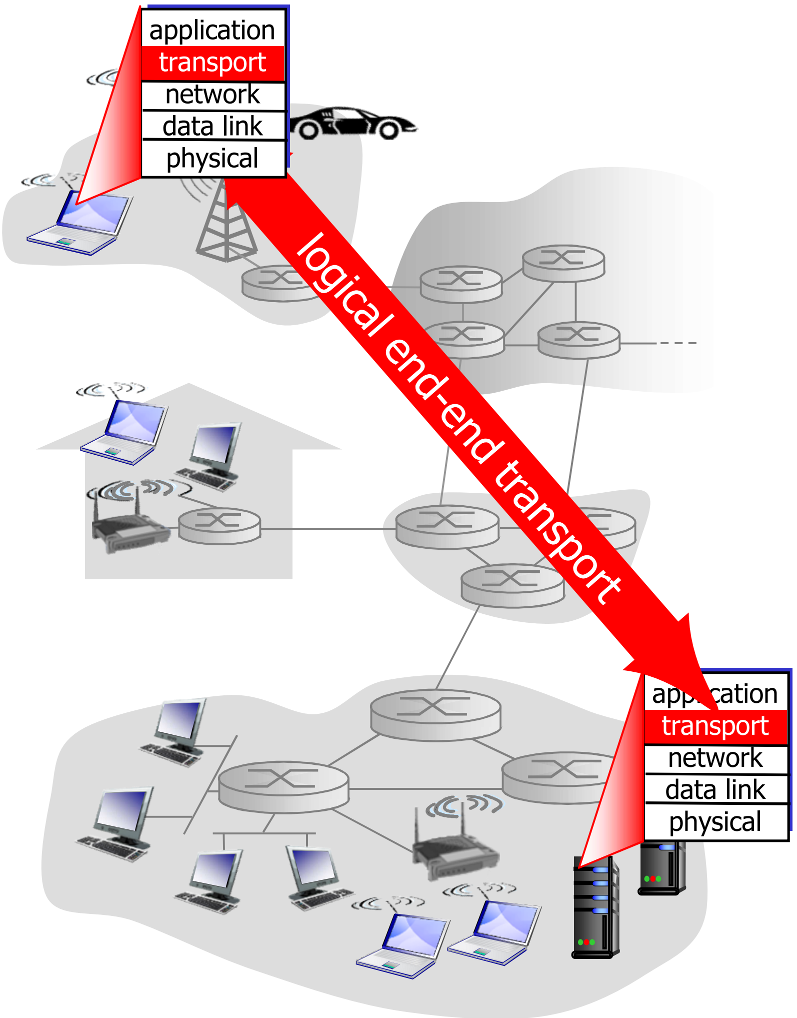
3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Transport services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into *segments*, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Transport vs. network layer

- *network layer:* logical communication between hosts
- *transport layer:* logical communication between processes
  - relies on, enhances, network layer services

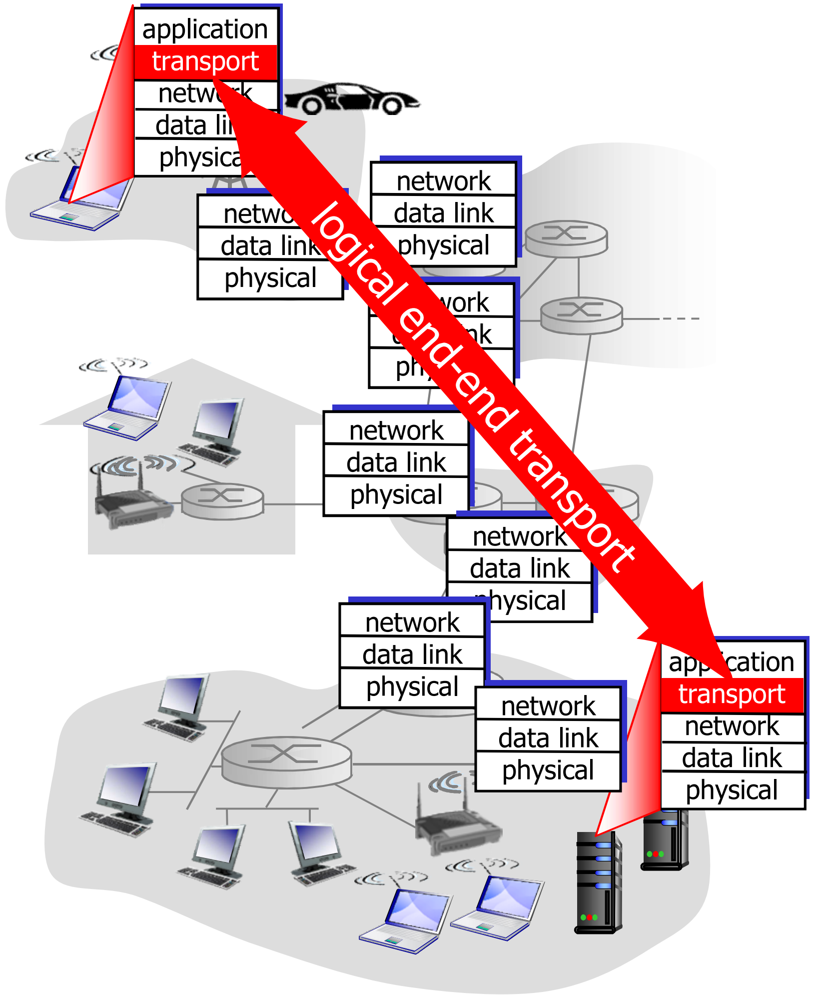
household analogy:

12 kids in Ann's house sending letters to 12 kids in Bill's house:

- hosts = houses
- processes = kids
- app messages = letters in envelopes
- transport protocol = Ann and Bill who demux to in-house siblings
- network-layer protocol = postal service

Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - no-frills extension of "best-effort" IP
- services not available:
  - delay guarantees
  - bandwidth guarantees

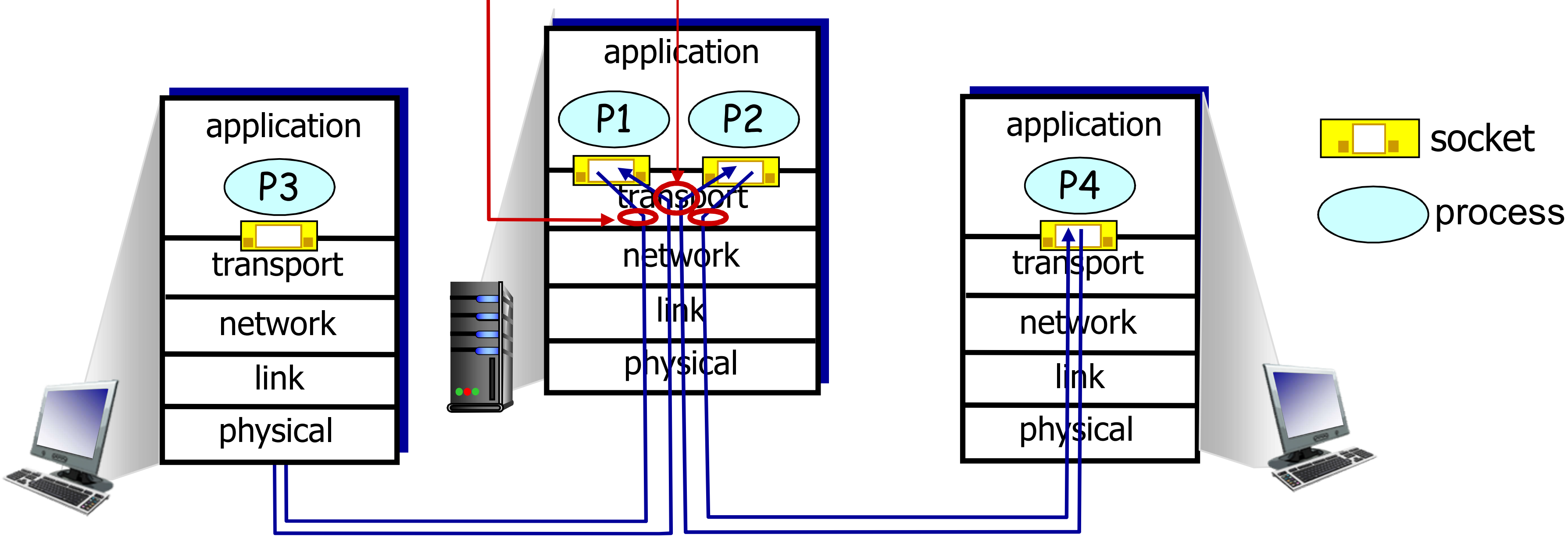


Chapter 3 outline

Multiplexing/demultiplexing

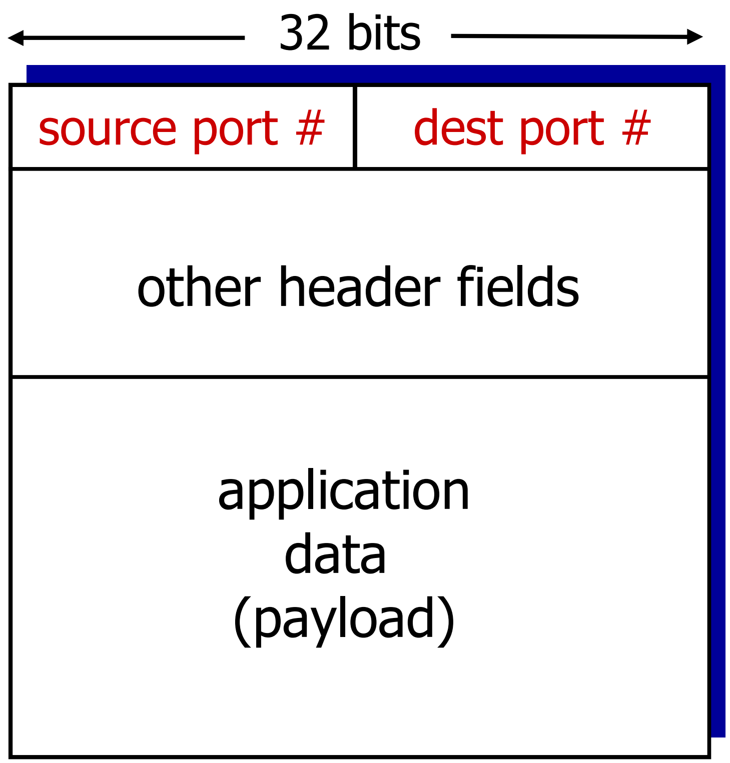
multiplexing at sender:
handle data from multiple sockets, add transport header (later used for demultiplexing)

demultiplexing at receiver:
use header info to deliver received segments to correct socket



How demultiplexing works

- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries one transport-layer segment
  - each segment has source, destination port number
- host uses *IP addresses & port numbers* to direct segment to appropriate socket



TCP/UDP segment format

Connectionless demultiplexing

- *recall:* created socket has host-local port #:

```
DatagramSocket mySocket1
= new DatagramSocket(12534);
```

- *recall:* when creating datagram to send into UDP socket, must specify
  - destination IP address
  - destination port #

- when host receives UDP segment:
  - checks destination port # in segment
  - directs UDP segment to socket with that port #

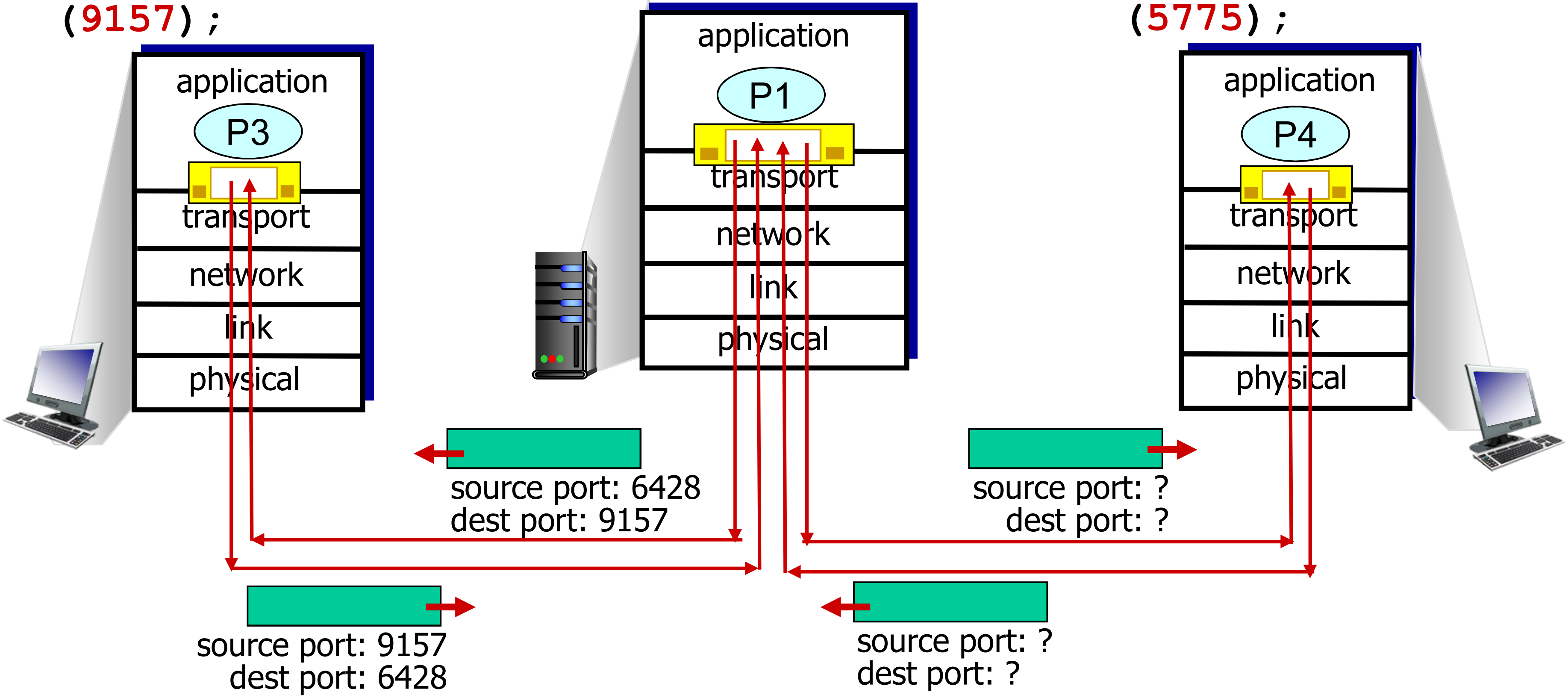
IP datagrams with *same dest. port #,* but different source IP addresses and/or source port numbers will be directed to *same socket* at dest

Connectionless demux: example

```
DatagramSocket
 mySocket2 = new
 DatagramSocket
 (9157);
```

```
DatagramSocket
 serverSocket = new
 DatagramSocket
 (6428);
```

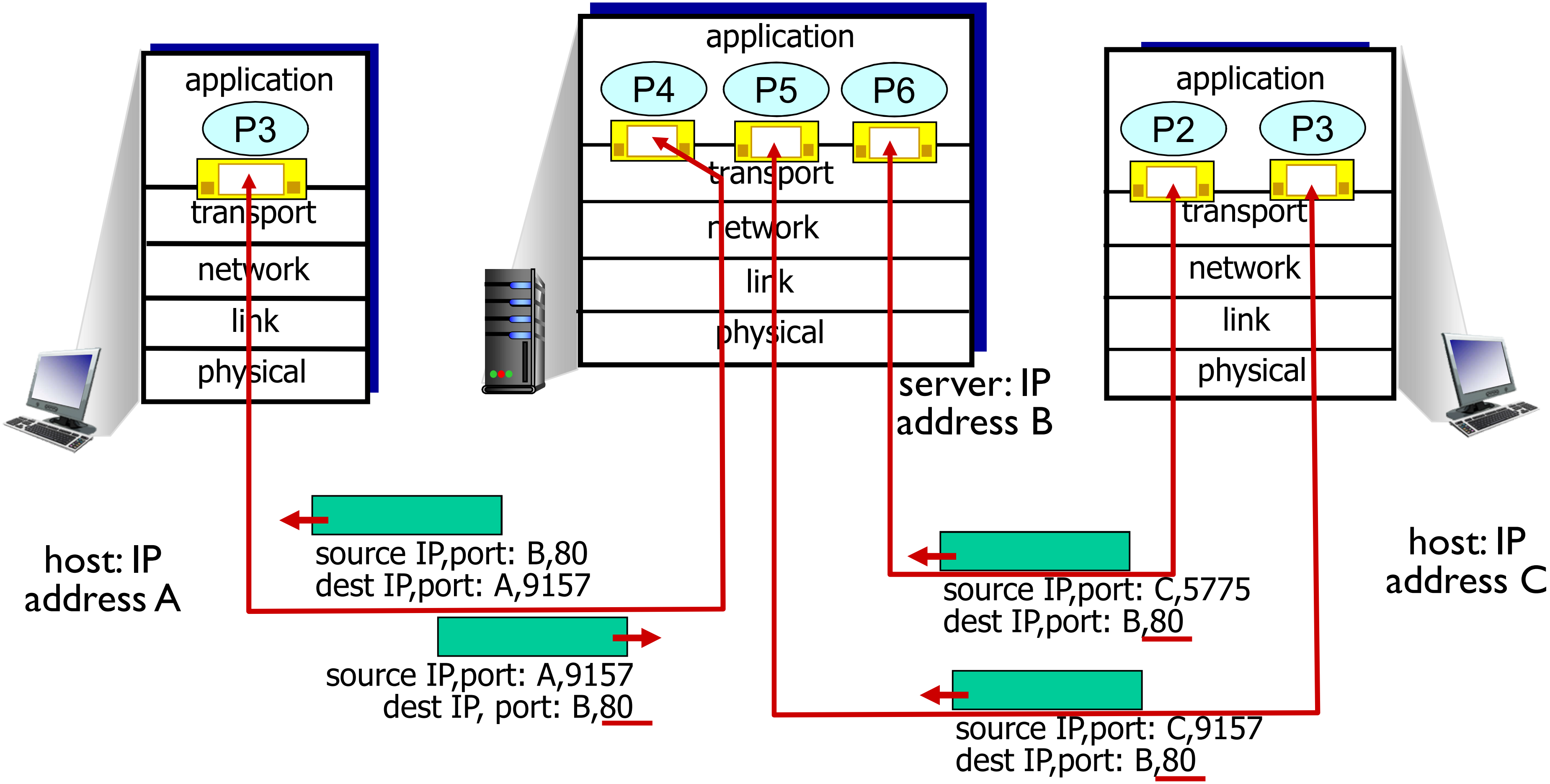
```
DatagramSocket
 mySocket1 = new
 DatagramSocket
 (5775);
```



Connection-oriented demux

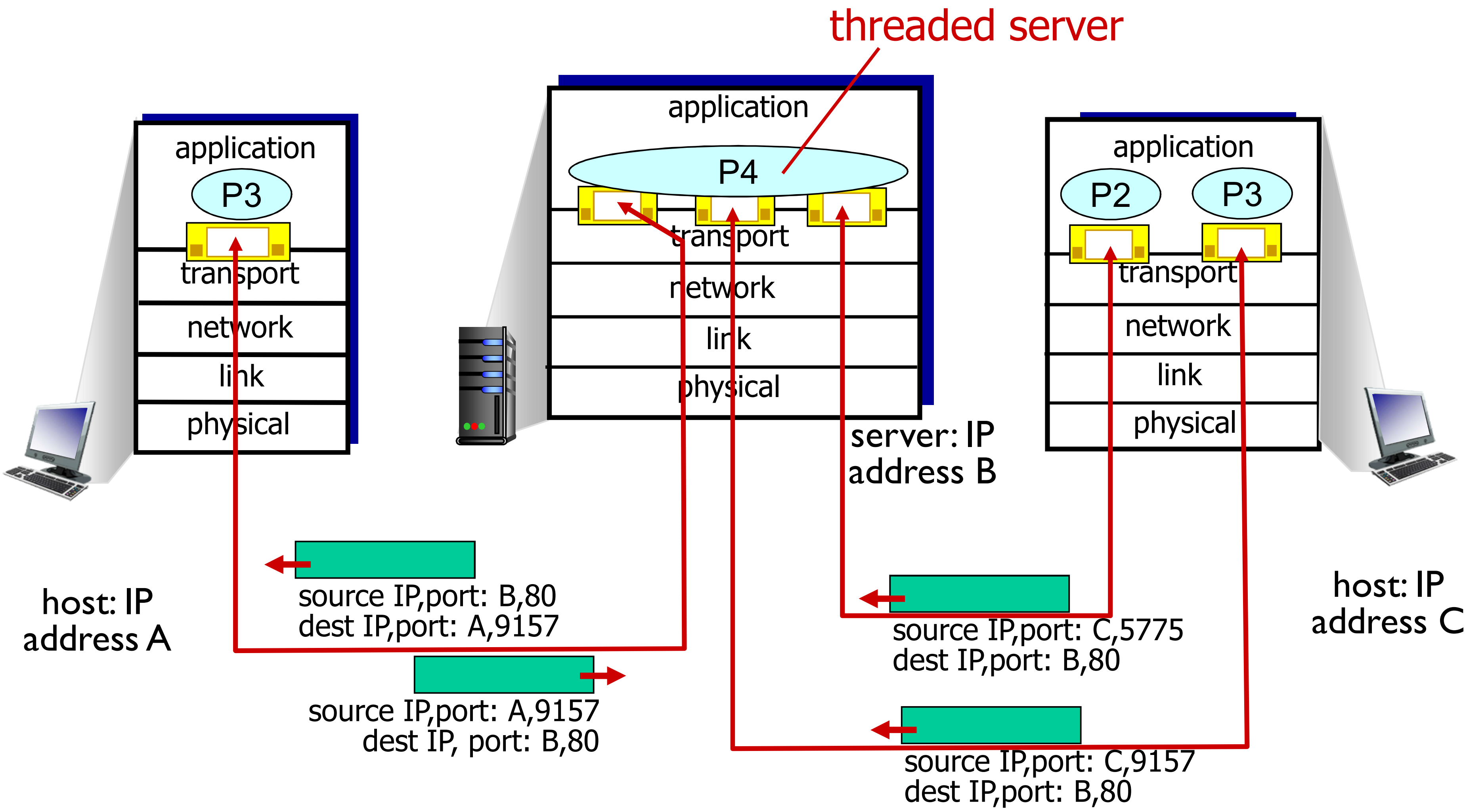
- TCP socket identified by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number
- demux: receiver uses all four values to direct segment to appropriate socket
- server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple
- web servers have different sockets for each connecting client
  - non-persistent HTTP will have different socket for each request

Connection-oriented demux: example



three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets

Connection-oriented demux: example



Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

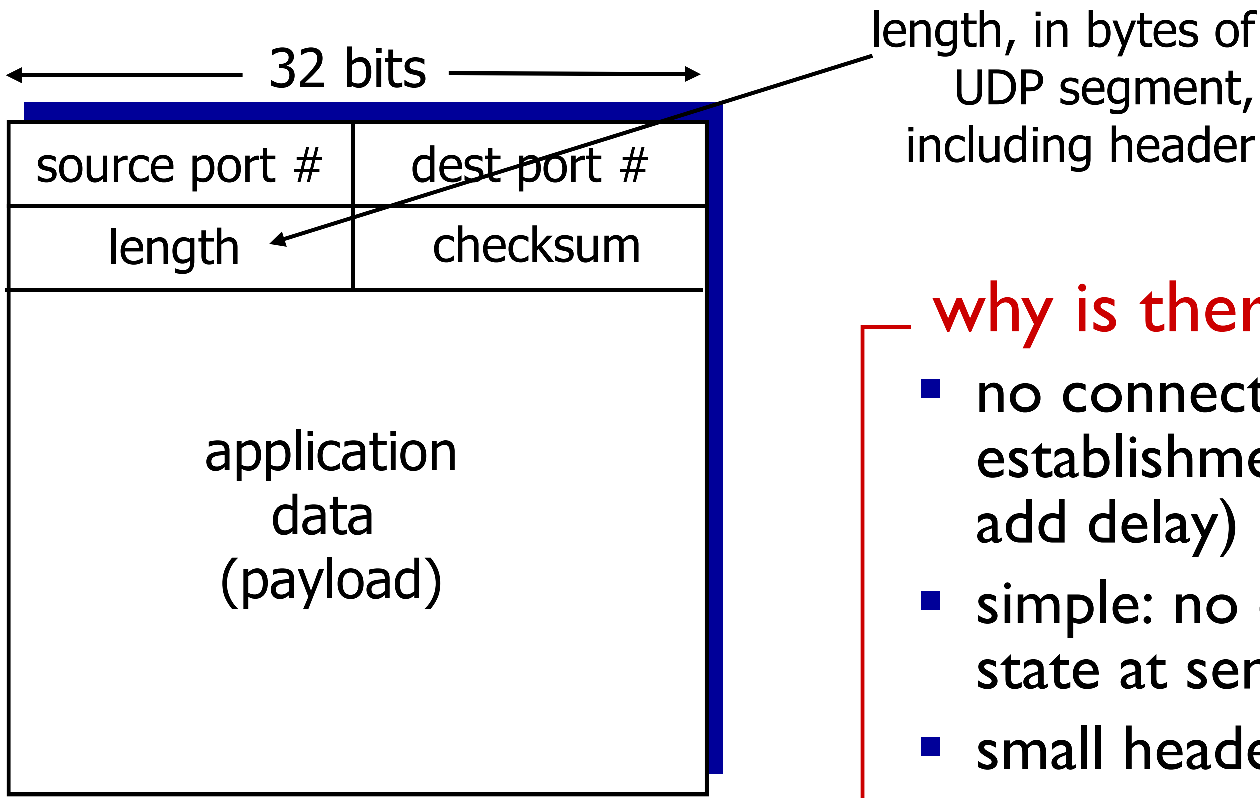
3.7 TCP congestion control

UDP: User Datagram Protocol [RFC 768]

- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - lost
  - delivered out-of-order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

- UDP use:
  - streaming multimedia apps (loss tolerant, rate sensitive)
  - DNS
  - SNMP
- reliable transfer over UDP:
  - add reliability at application layer
  - application-specific error recovery!

UDP: segment header



UDP segment format

why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired

UDP checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted segment

sender:

- treat segment contents, including header fields, as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?* More later ….

Internet checksum: example

example: add two 16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound (순환자리올림)	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

After-study Test :

1) 트랜스포트 계층 프로토콜의 최종 정보 전달 대상은 무엇인가?

① Host
② Program
③ Server
④ Process

2) 트랜스포트 계층 프로토콜이 전달하는 정보 단위의 이름은?

① 패킷(packet)
② 세그먼트(segment)
③ 프레임(frame)
④ 메시지(message)

3) 다음 트랜스포트 계층 프로토콜에서 제공하는 서비스가 아닌 것은?

① 오류 제어(error control)
② 다중화(multiplexing)
③ 지연시간 제어(delay control)
④ 혼잡 제어(congestion control)

4) 응용 프로세스가 트랜스포트 프로토콜 서비스를 사용하기 위해 생성하는 자료 구조는?

① Socket
② Port
③ Queue
④ Message box

5) 비연결형(connectionless) 서버 소켓의 식별자는?

① 서버 IP 주소
② 서버 Port 번호
③ 서버 IP 주소 + 서버 Port 번호
④ 클라이언트 IP 주소 + 클라이언트 Port 번호 + 서버 IP 주소 + 서버 Port 번호

6) 연결형(connection-oriented) 서버 소켓의 식별자는?

① 서버 IP 주소
② 서버 Port 번호
③ 서버 IP 주소 + 서버 Port 번호
④ 클라이언트 IP 주소 + 클라이언트 Port 번호 + 서버 IP 주소 + 서버 Port 번호

7) 다음 중 UDP에 대한 설명 중 틀린 것은?

① 비연결형 프로토콜이다.

② Port 번호로 다중화 서비스를 제공한다.

③ 전달 순서를 보장한다.

④ 오류 검출 서비스를 제공한다.

8) 다음 중 UDP의 특징이 아닌 것은?

① 오류 처리 로직이 있지만 간단하다.

② 헤더가 간단하다.

③ 지연시간이 작다.

④ 혼잡제어 로직이 있지만 간단하다.

9) 이진수 0110의 1의 보수값(one's complement)은 얼마인가?

① 0110

② 1001

③ 0110

④ 1111