COMPUTER NETWORKS

Animesh Giri
Department of Computer Science & Engineering

COMPUTER NETWORKS

Transport Layer

Animesh Giri
Department of Computer Science & Engineering

COMPUTER NETWORKS

Multiplexing & Demultiplexing

Animesh Giri
Department of Computer Science & Engineering

In this segment

What is transport-layer multiplexing and demultiplexing

- How demultiplexing works
  - TCP / UDP segment format
- Connectionless demultiplexing - UDP
- Connectionless demux: Example
- Connection-oriented demux - TCP
- Connection-oriented demux: Example

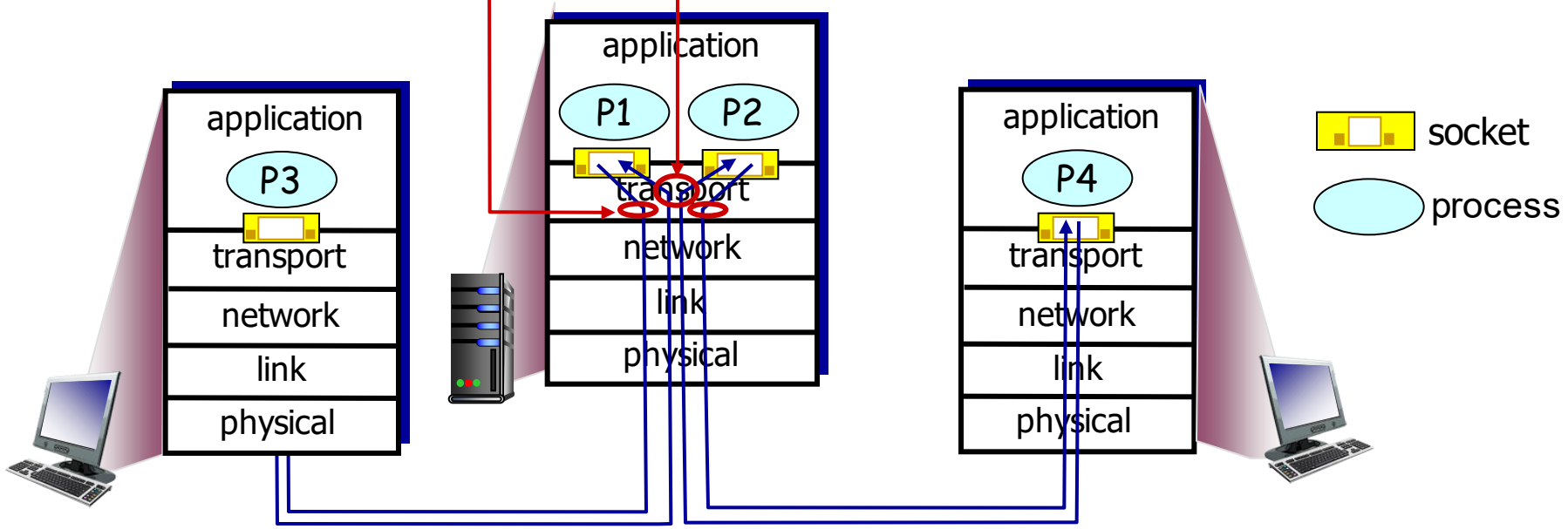
Multiplexing / demultiplexing

multiplexing at sender:

handle data from multiple sockets, add transport header (later used for demultiplexing)

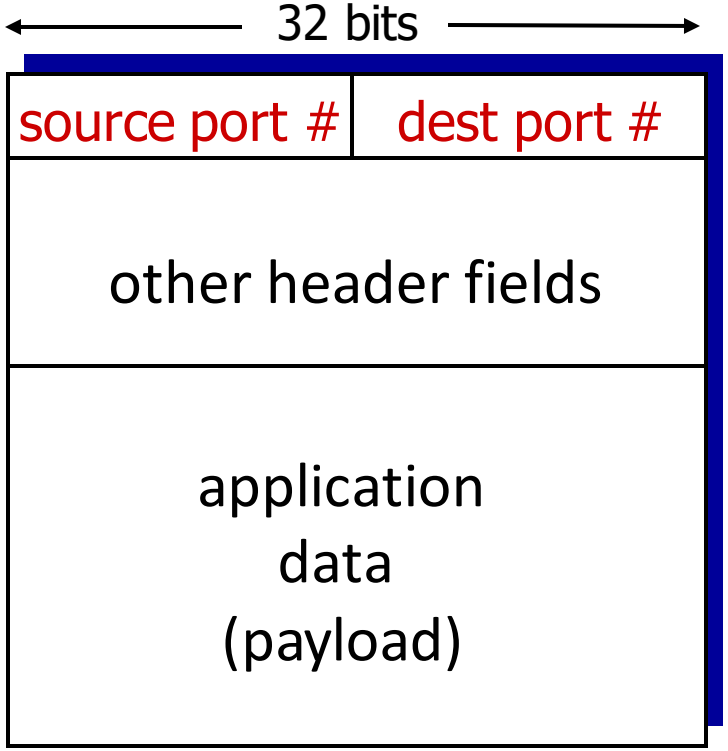
demultiplexing at receiver:

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 #

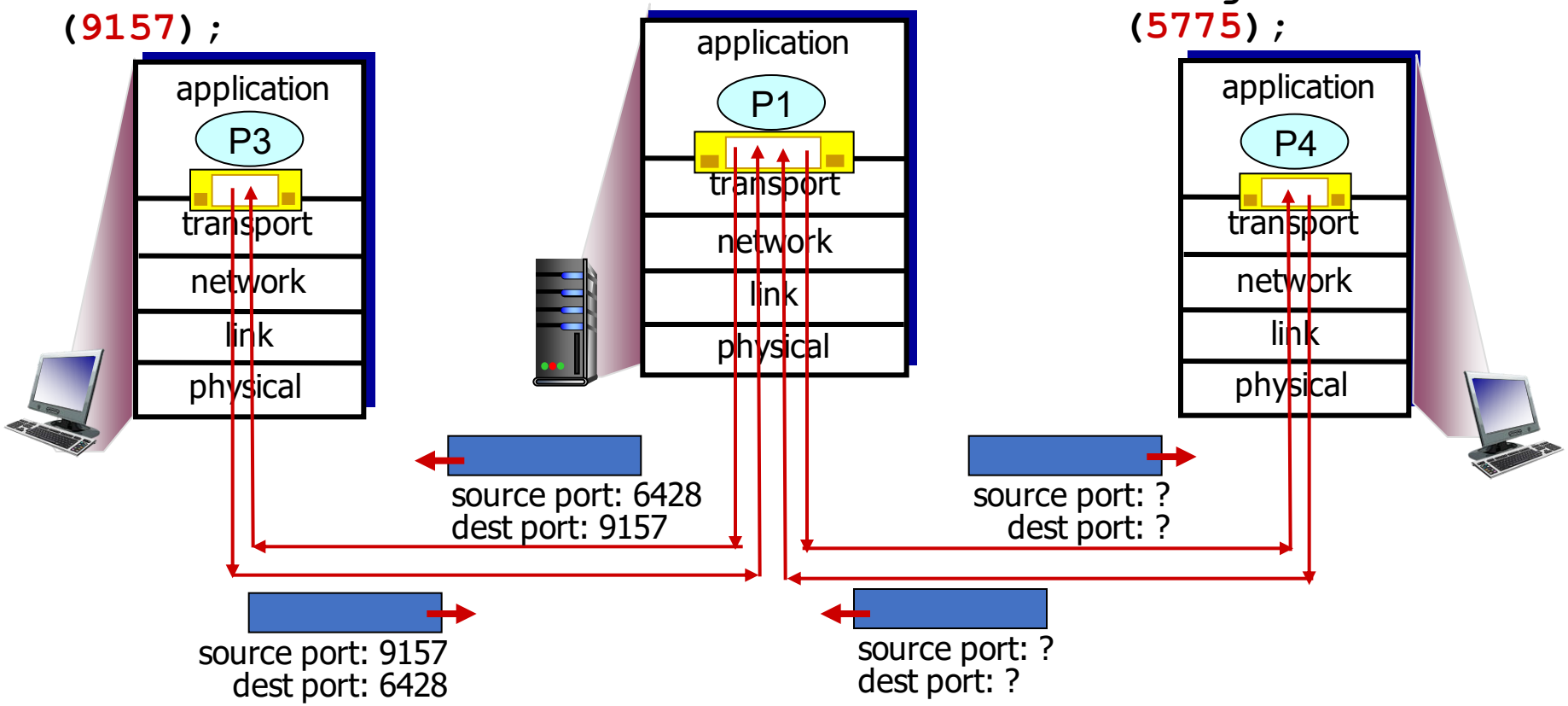
IP datagrams with *same dest. port #,* but different source IP addresses and/or source port numbers will be directed to *same socket* at dest

Connectionless demux: example

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

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

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

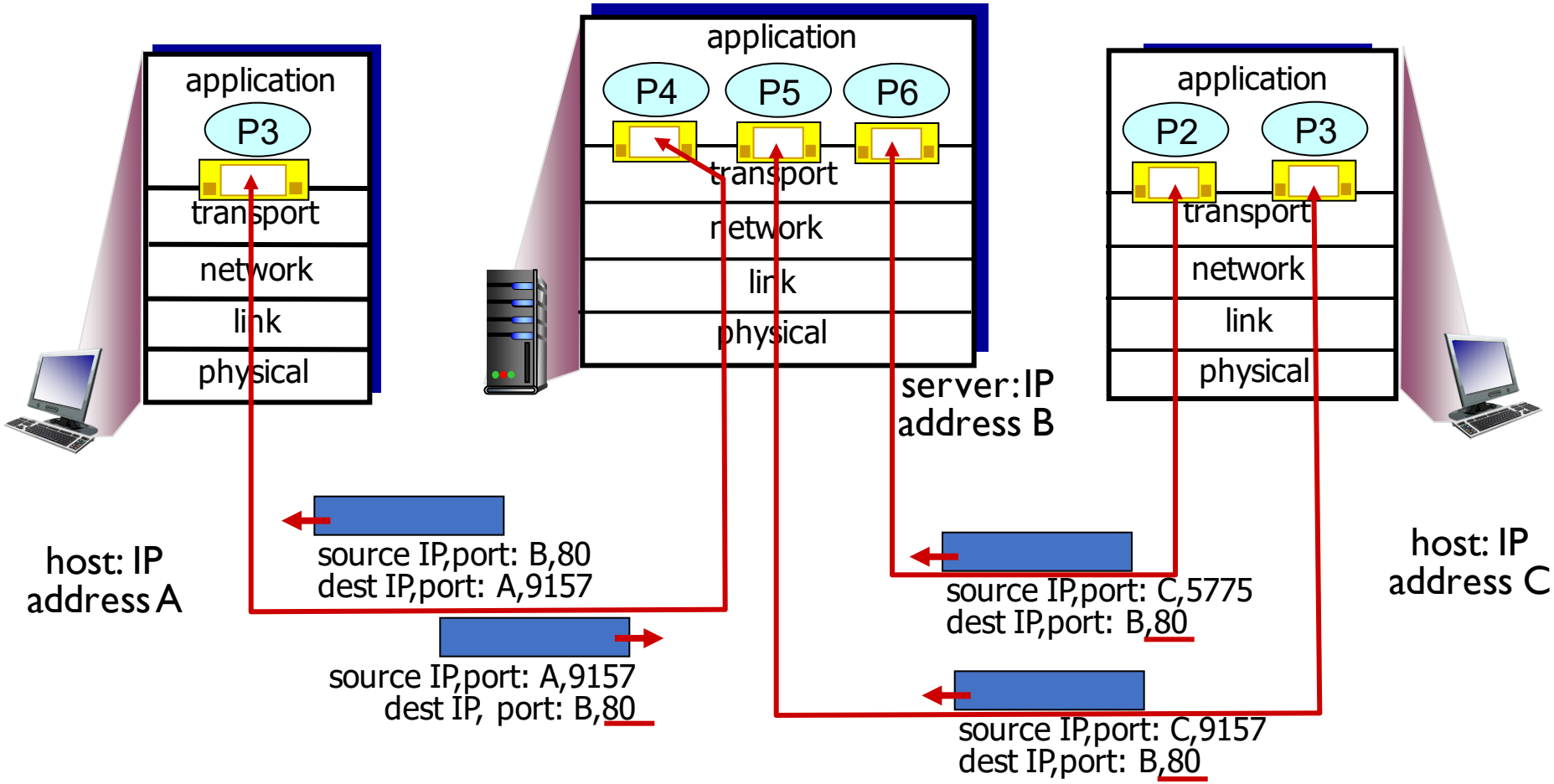


Connection-oriented demux

- TCP socket identified by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number
- demux: receiver uses all four values (4-tuple) 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

COMPUTER NETWORKS

Connection-oriented demux : example



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

THANK YOU

Animesh Giri
Department of Computer Science & Engineering

animeshgiri@pes.edu

+91 80 6618 6603