

# CSE 3214: Computer Network Protocols and Applications –Transport Layer

Dr. Peter Lian, Professor  
Department of Computer Science and Engineering  
York University  
Email: [peterlian@cse.yorku.ca](mailto:peterlian@cse.yorku.ca)  
Office: 1012C Lassonde Building  
Course website: [http://wiki.cse.yorku.ca/  
course\\_archive/2012-13/W/3214](http://wiki.cse.yorku.ca/course_archive/2012-13/W/3214)

## Chapter 3: Transport Layer

### our goals:

- ❖ understand principles behind transport layer services:
  - multiplexing, demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- ❖ learn about Internet transport layer protocols:
  - UDP: connectionless transport
  - TCP: connection-oriented reliable transport
  - TCP congestion control

Transport Layer 3-2

## 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

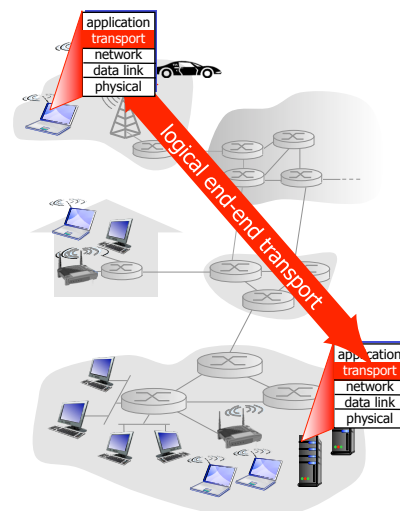
### 3.6 principles of congestion control

### 3.7 TCP congestion control

Transport Layer 3-3

## 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
  - receive side: reassembles segments into messages, passes to app layer
- ❖ more than one transport protocol available to apps
  - Internet: TCP and UDP



Transport Layer 3-4

## Transport vs. network layer

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

### *household analogy:*

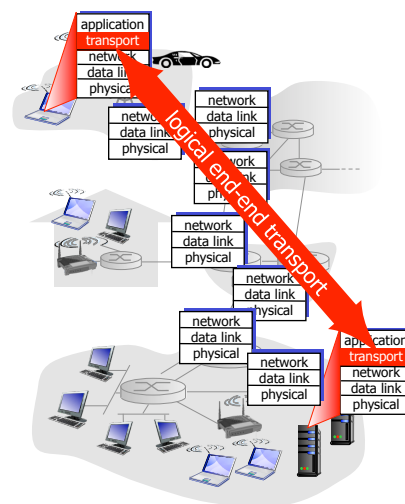
*12 kids in Ann's house sending letters to 12 kids in Bill's house:*

- ❖ 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

Transport Layer 3-5

## 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



Transport Layer 3-6

## 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

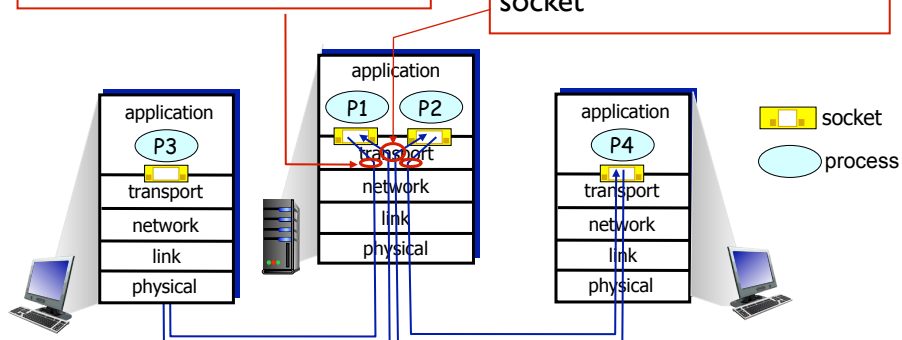
3.7 TCP congestion control

Transport Layer 3-7

## Multiplexing/demultiplexing

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

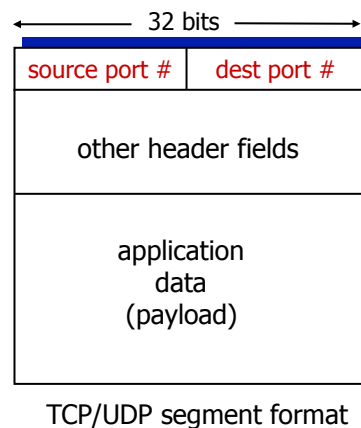
*demultiplexing at receiver:*  
use header info to deliver received segments to correct socket



Transport Layer 3-8

## 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



Transport Layer 3-9

## Connectionless demultiplexing

- ❖ *recall in Socket Programming:*

```
clientSocket=socket(AF_INET,
                    SOCK_DGRAM)
clientSocket.sendto(message,(serverName,
                             serverPort))
```

- ❖ When creating datagram to send into UDP socket, must specify

- destination IP address
- destination port #

- ❖ when host receives UDP segment:

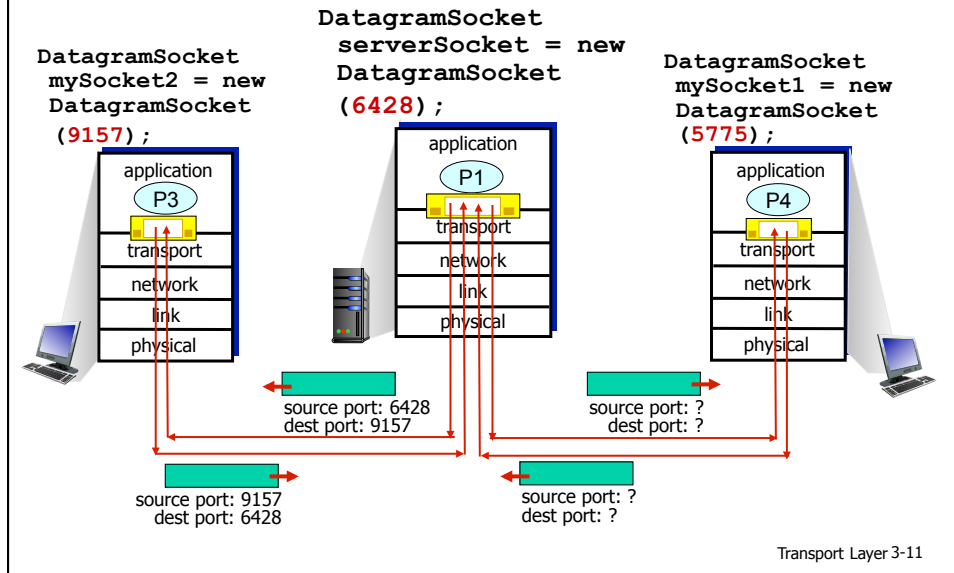
- checks destination port # in segment
- directs UDP segment to socket with that port #



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

Transport Layer 3-10

## Connectionless demux: example

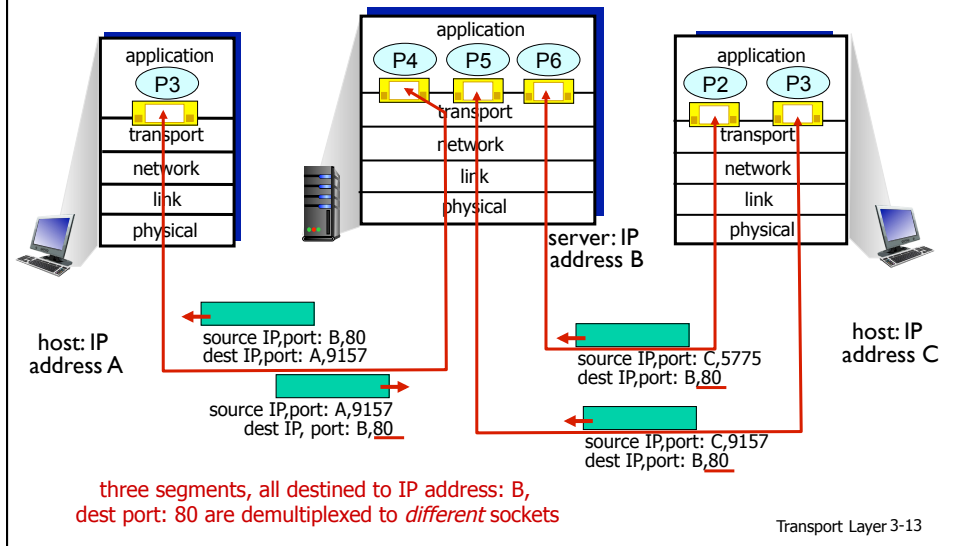


## Connection-oriented demux

- ❖ 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

Transport Layer 3-12

## Connection-oriented demux: example



## Connection-oriented demux: example

