# 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
Office: 1012C Lassonde Building
Course website: http://wiki.cse.yorku.ca/course_archive/2012-13/W/3214

# Chapter 3: Transport Layer

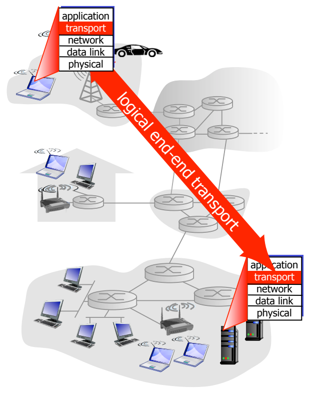
our goals:

- 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

## Chapter 3 outline

## 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 vs. network layer

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

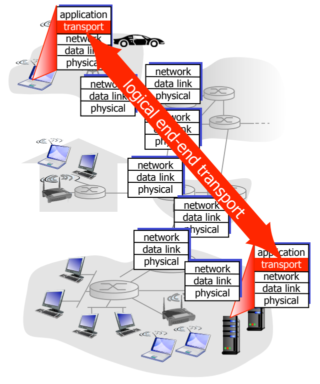
*household analogy:*

*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

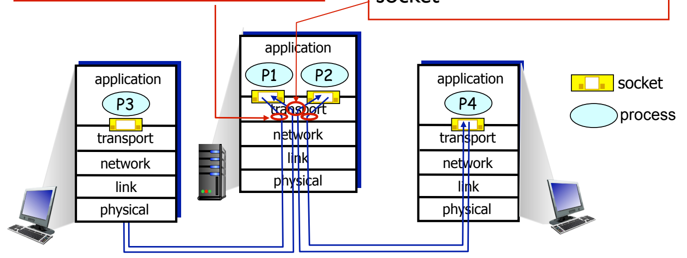


## Chapter 3 outline

## Multiplexing/demultiplexing

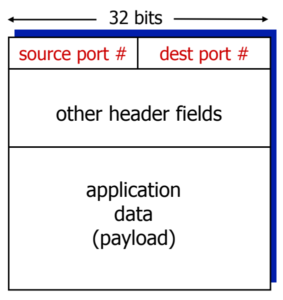
*multiplexing at sender:*
handle data from multiple sockets, add transport header (later used for demultiplexing)

*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 in Socket Programming:*

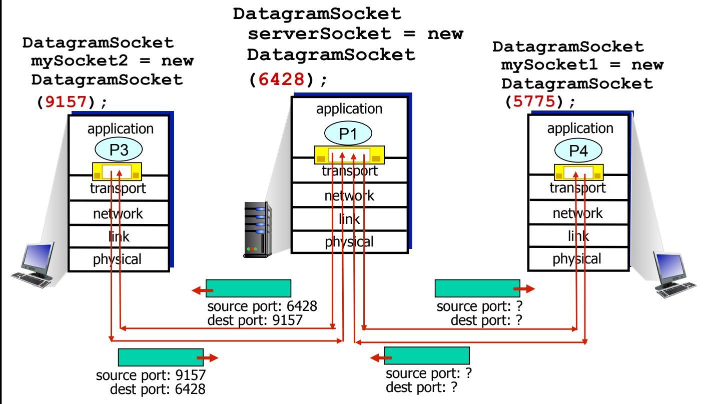
clientSocket=socket(AF_INET,
    SOCK_DGRAM)
clientSocket.sendto(message,(serverName,
    serverPort))

- 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

## Connectionless demux: example



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

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

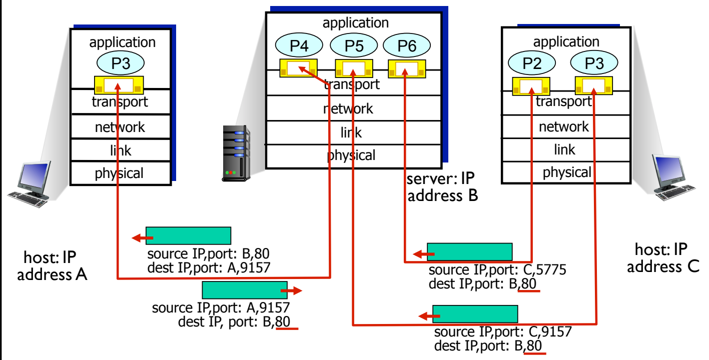
```
DatagramSocket
 mySocket1 = new
 DatagramSocket
 (5775);
```

## 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

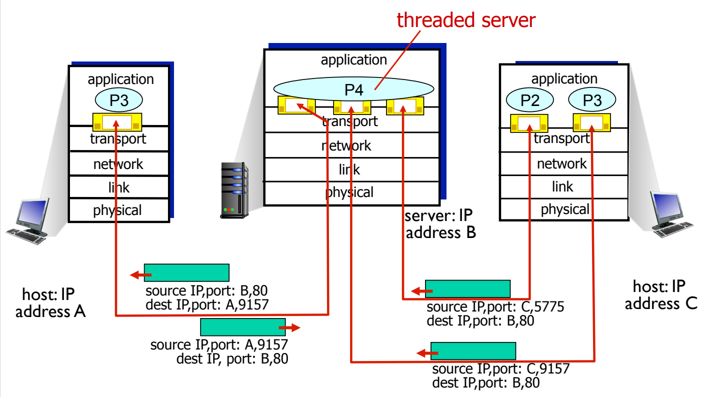
- server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple
- web servers have different sockets for each connecting client
  - non-persistent HTTP will have different socket for each request

## Connection-oriented demux: example



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

## Connection-oriented demux: example



threaded server
application
P3
transport
network
link
physical
host: IP address A
application
P4
transport
network
link
physical
server: IP address B
application
P2
P3
transport
network
link
physical
host: IP address C
source IP,port: B,80
dest IP,port: A,9157
source IP,port: A,9157
dest IP, port: B,80
source IP,port: C,5775
dest IP,port: B,80
source IP,port: C,9157
dest IP,port: B,80