CS348: Computer Networks

Transport Layer Introduction, UDP

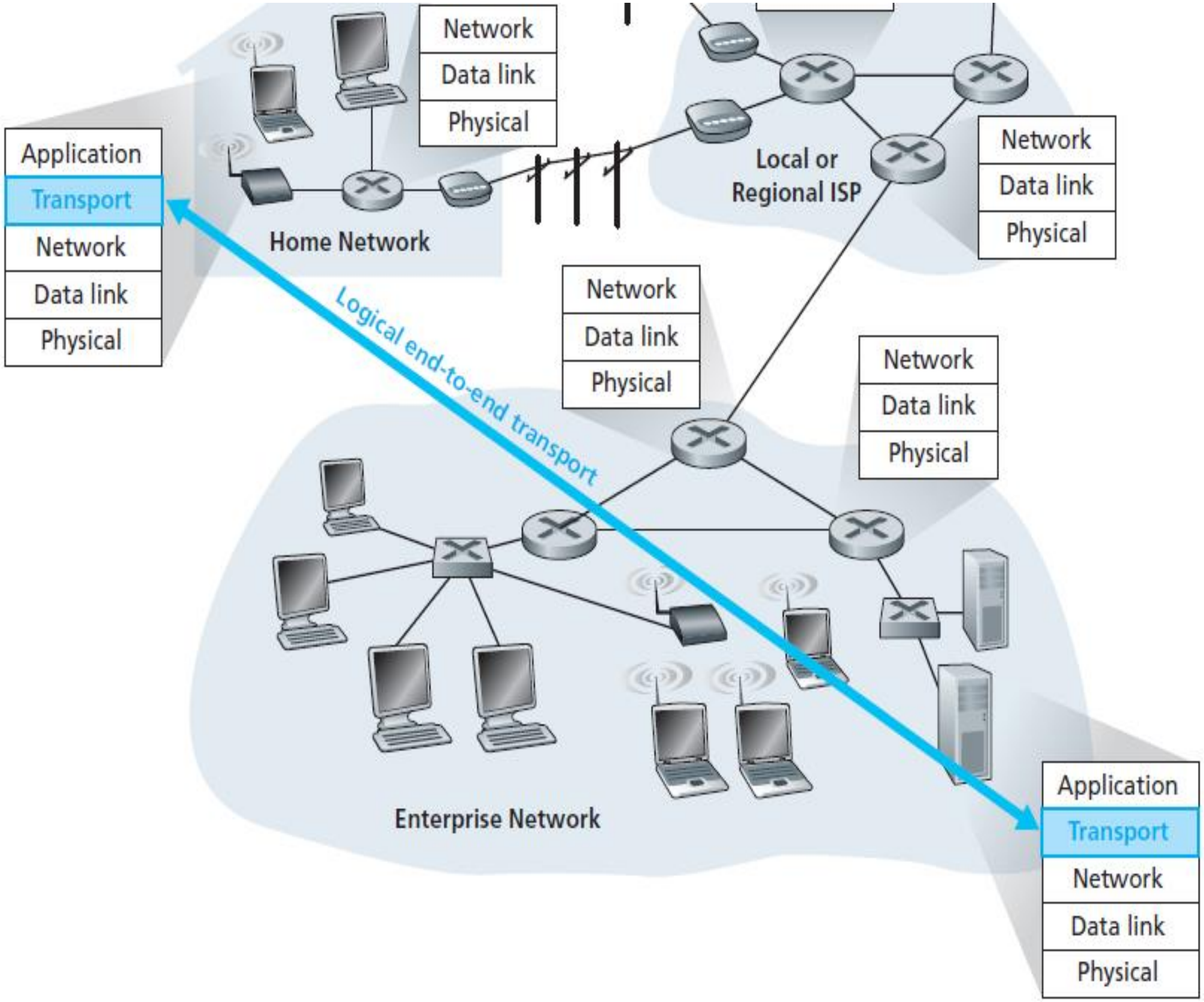
Dr. Manas Khatua
Asst. Professor
Dept. of CSE, IIT Guwahati
E-mail: manaskhatua@iitg.ac.in

Transport Layer

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 service



- provide *logical communication* between application 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

Cont…

- In TCP/IP suite, it provides services to the Application layer and receives services from the Network layer.

- General transport services
  - process-to-process connection
  - logical addressing
  - multiplexing and de-multiplexing
  - reliable delivery
  - flow and congestion control

- Transport-Layer Protocol strategies
  - Stop-and-Wait
  - Go-back-N
  - Selective-Repeat

- Transport-Layer Protocols for the Internet
  - Connection less protocol: UDP
  - Connection oriented protocol : TCP

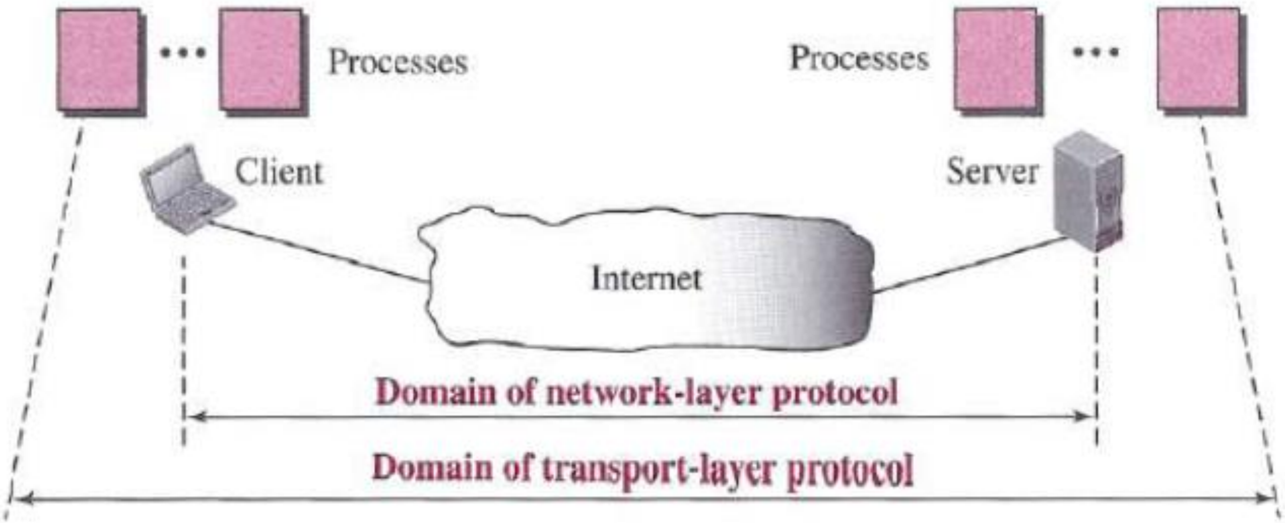
Network **v/s** Transport Layers

- Network layer provides *logical communication* between two hosts located in two different networks

- IP service model: best-effort delivery but unreliable service

- Network layer protocols are also implemented in network routers

Process-to-Process Communication



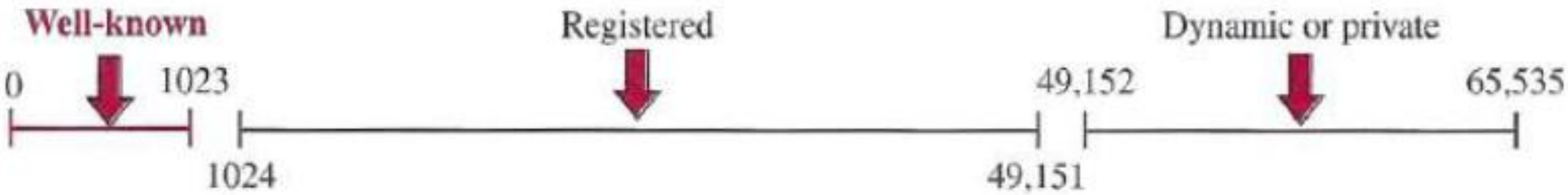
- One architecture for process-to-process communication: client-server approach

- Host is identified by IP address
- Process is identified by port number

- TCP/IP supports port numbers 0-65535 (16 bits)
  - Client uses: ephemeral ports (short-lived ports, >1023)
  - Server Uses: well-known ports in general

Logical Addressing

- Internet Corporation for Assigned Names and Numbers (ICANN)



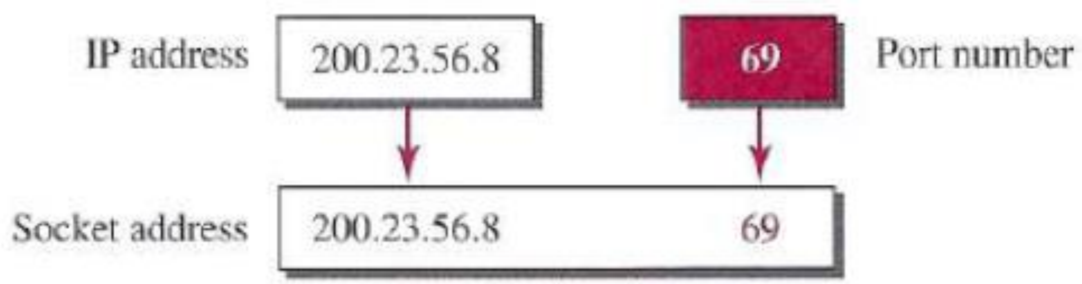
- Example:
  - ports are stored in /etc/services
  - FTP : 20, 21
  - SSH, SCP : 22
  - Echo : 7
  - DNS : 53
  - HTTP : 80
  - SNMP : 161,162
  - BGP : 179

Private port numbers are available for use by any application to use in communicating with any other application, using the Internet's TCP or UDP.

Companies and other users should register Registered port number with the ICANN for use by their applications.

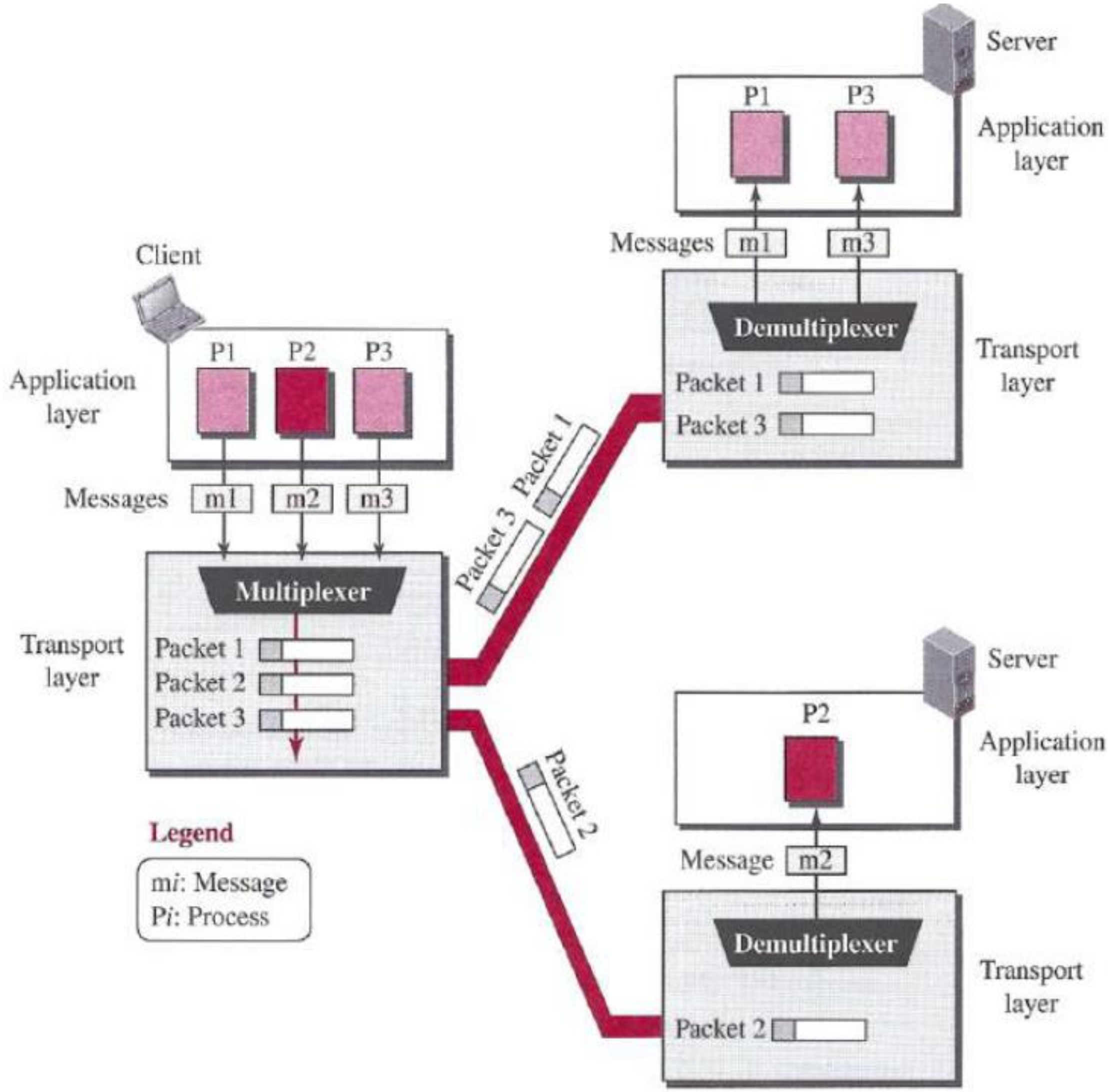
Socket Address

- To use the services of the transport layer in the Internet,
  - we need a pair of socket addresses:
    - the client socket address
    - the server socket address



Multiplexing / Demultiplexing

- Suppose you are sitting in front of a computer, and you are browsing Web pages while running one FTP session and Telnet sessions.
- So, 3 processes
  - HTTP
  - FTP
  - Telnet

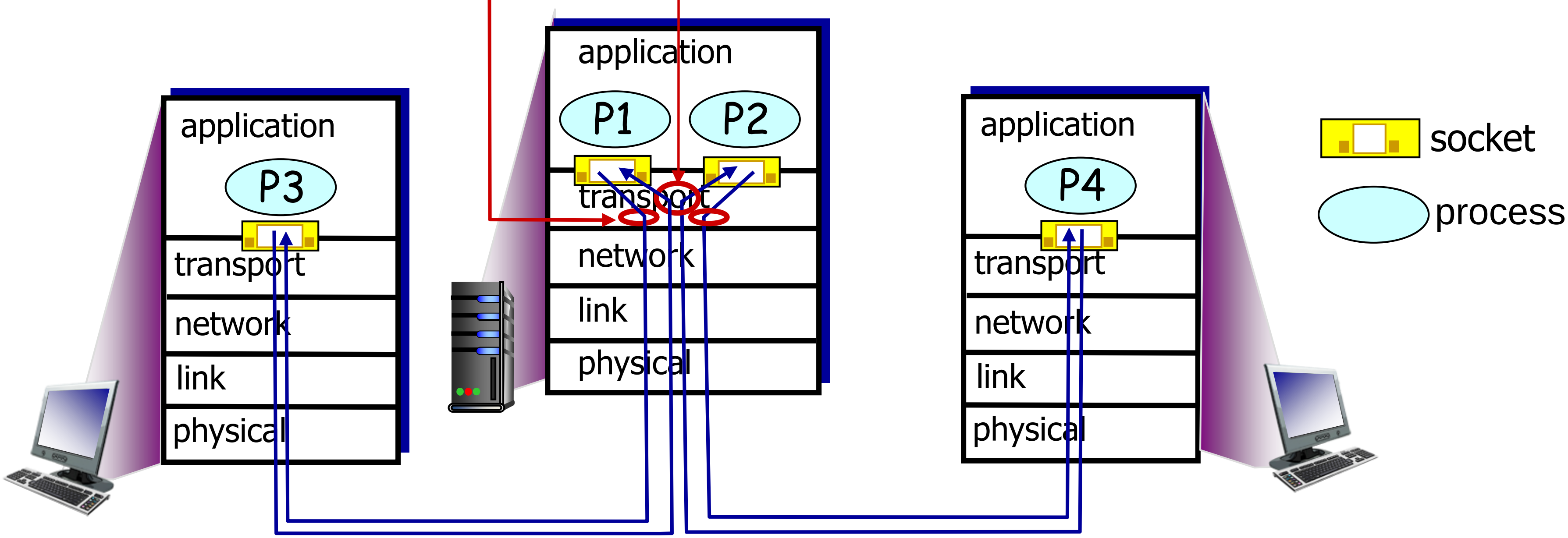


- How does a segment forwarded to intended process?
  - using Socket Address
  - Method: multiplexing / demultiplexing

Cont…

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



Connection-less De-multiplexing

- created socket has host-local port #:

```
clientSocket.bind('', 19157)
```

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

```
clientSocket.sendto(message,
   (serverName, serverPort))
```

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

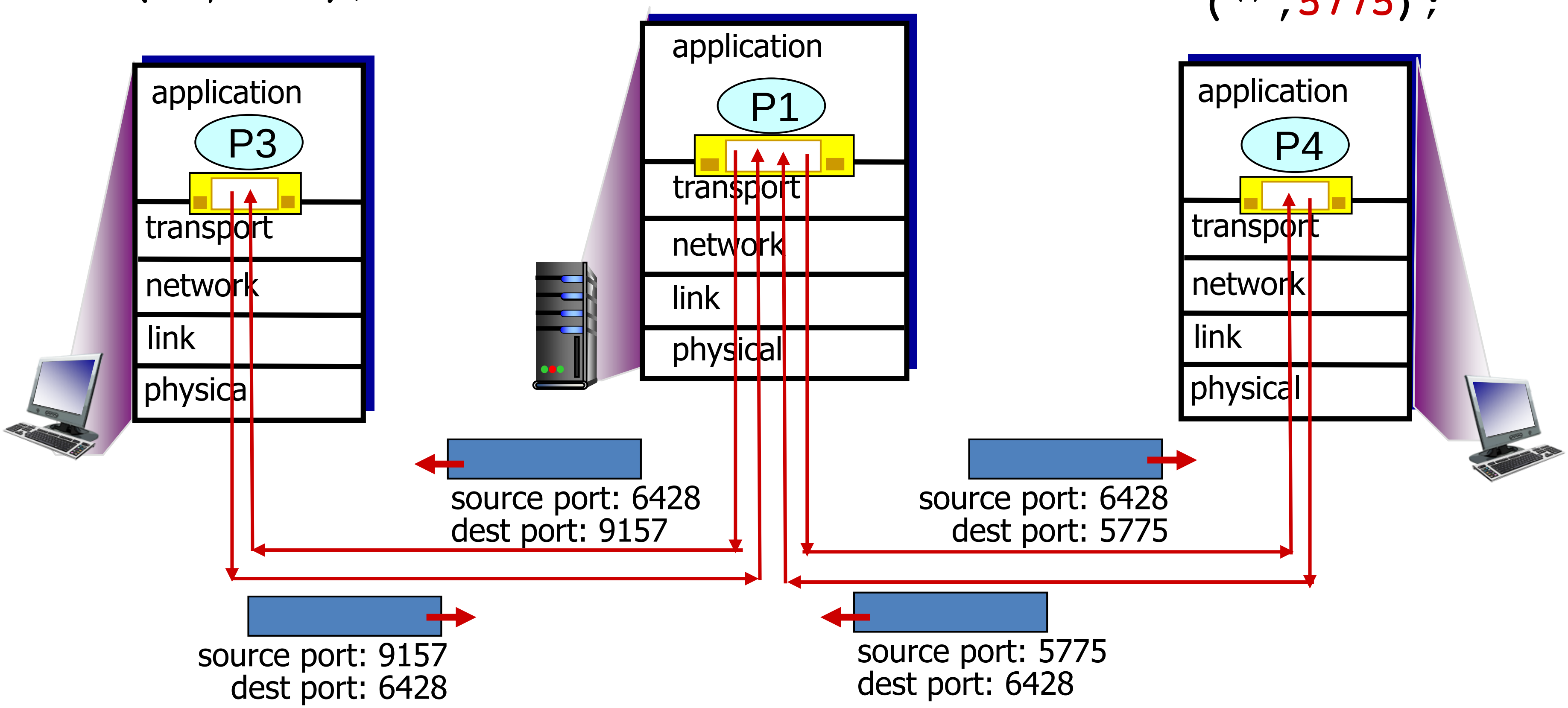
IP datagrams with *same dest. port #,* but different source IP addresses (i.e. host) and/or source port numbers (i.e. process) will be directed to *same socket* at destination.

Connection-less demux example

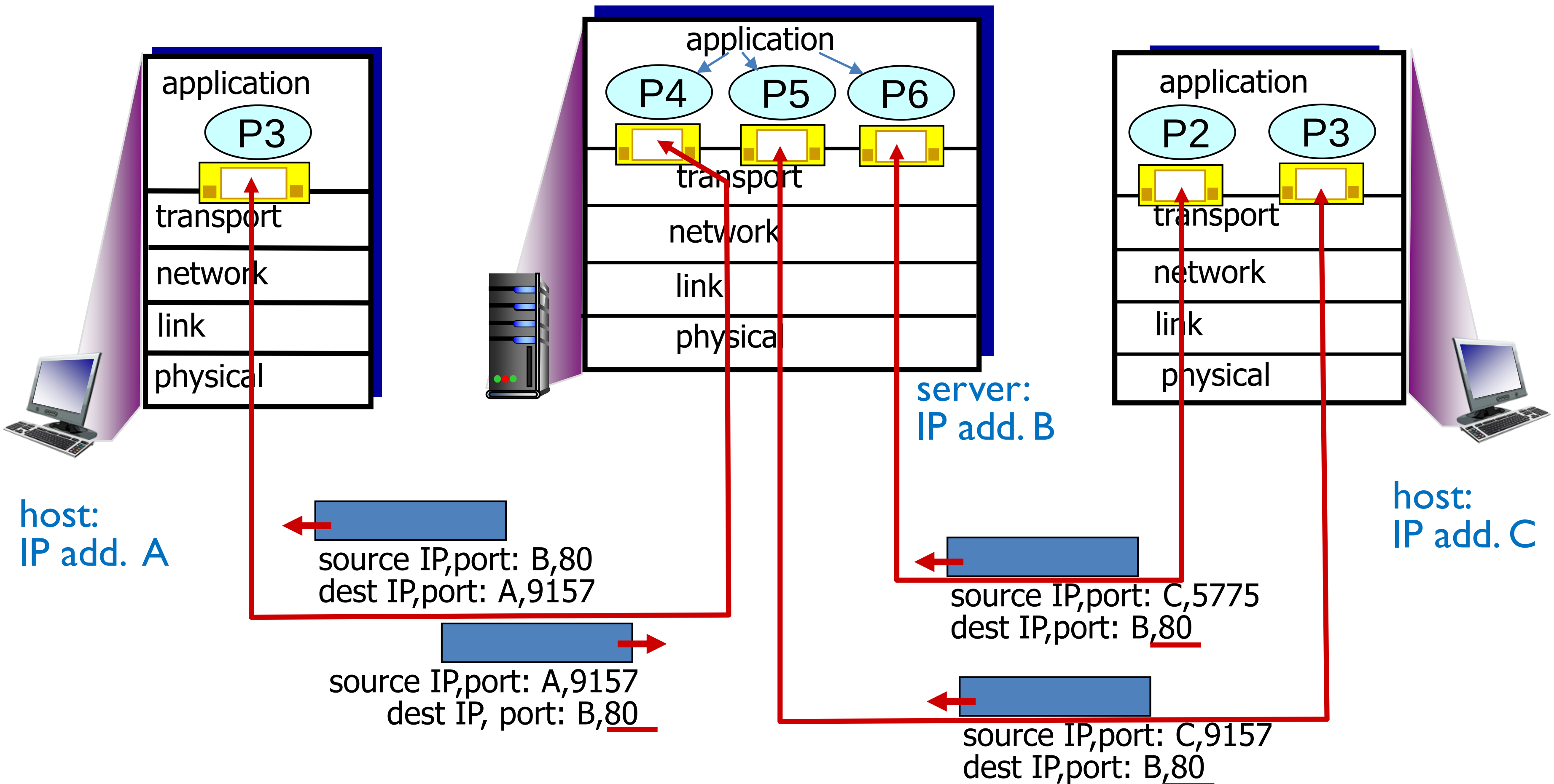
```
clientSocket =
socket(AF_INET,
SOCK_DGRAM);
clientSocket.bind
(('',9157);
```

```
serverSocket =
socket(AF_INET,
SOCK_DGRAM);
serverSocket.bind
(('',6428);
```

```
clientSocket =
socket(AF_INET,
SOCK_DGRAM);
clientSocket.bind
(('',5775);
```



Connection-oriented demux example



three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets (created for data transfer)

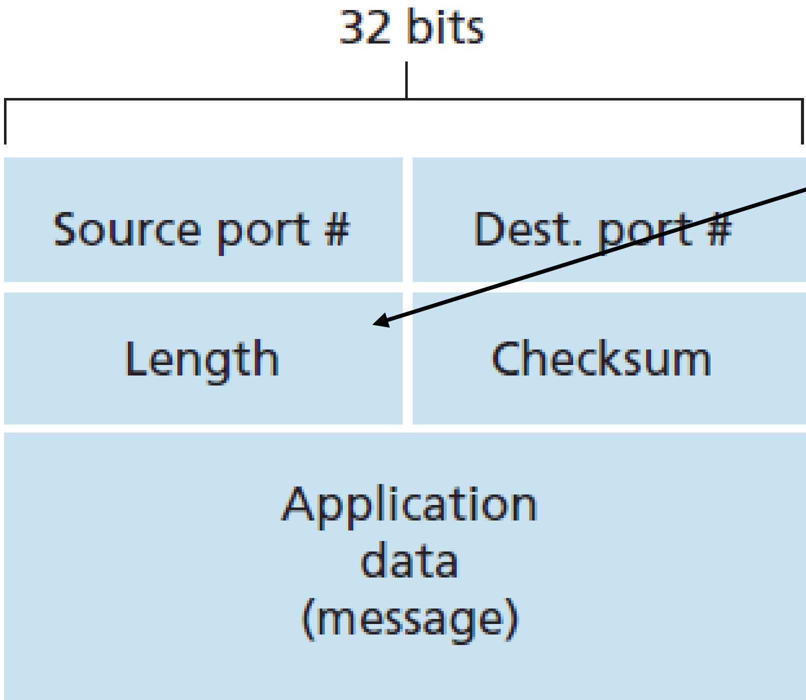
Note: TCP socket identified by 4-tuple: source IP & port#, dest. IP & port#

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 the destination
- *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
  - DHCP
- Reliable transfer over UDP
  - add reliability at application layer
  - application-specific error recovery!

UDP segment structure



Length (in bytes) of UDP segment, including header

why is there a UDP?

- no connection establishment (which can add delay)
- no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired
- Finer application-level control over what data is sent, and when

UDP Services

- Process-to-process communication
  - Need socket address (IP + Port)
- Connectionless service
  - No sequence number
  - So, no relation between UDP datagrams
  - No connection establishment
  - So, datagram can travel through different path
  - No segmentation (message size < (65535 - 8))
- No flow control
  - No window mechanism
- No error control
  - Error detection through checksum, but no control
- No congestion control
  - Assumption is that congestion will not occur as UDP datagrams are small in size
- Queuing
  - Queues are associated with port
- Multiplexing / Demultiplexing
  - One UDP, but several process in application layer wants to use its services

Applications using UDP

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Figure 3.6 ♦ Popular Internet applications and their underlying transport protocols

UDP Checksum

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

- It is optional
- It considers three parts:
  - Pseudo header
    - fields from IP header: source IP, dest. IP, upper layer protocol, datagram length
  - UDP header
  - data (from application layer)

- Datalink layer has error detection mechanism.
- Why do we need checksum in transport layer?
  - using Link layer error detection (i.e. by CRC) neither link-by-link reliability nor in-memory (in intermediate node/device) error detection is guaranteed w.r.t. end-to-end service

- What is pseudoheader?
  - contains some information from the real IPv4 header.
  - it is not the real IPv4 header used to send an IP packet

- Why do we need pseudoheader?
  - Socket-address needs to be uncorrupted.
    - a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host!
  - to ensure that the packet belongs to UDP, and not to TCP

Cont…

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.

Cont…

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	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

Note: UDP at the sender side performs the 1's complement of the sum of all the16-bit words in the segment, and the result is put in the checksum field

Cont…

- What value is sent for the checksum in each one of the following cases?
  - Case1: The sender decides not to include the checksum.
  - Case2: the value of the sum is all 1s.
  - Case3: the value of the sum is all 0s.

- **Solutions**:
- **Case1**:
  - All 0s
- **Case2**:
  - When the sender complements the sum, the result is all 0s; the sender complements the result again before sending. The value sent for the checksum is all 1s.
  - The second complement operation is needed to avoid confusion with the previous case.
  - Note that this does not create confusion because the value of the checksum is never all 1s in a normal situation
- **Case3**:
  - This situation never happens because it implies that the value of every term included in the calculation of the sum is all 0s, which is impossible.

Application Types uses UDP

- If the request and response can each fit in a single user datagram, a connectionless service may be preferable.
  - e.g. DNS request and response;
  - but, not suitable in SMTP as e-mail size could be large
- Lack of error control is advantageous sometimes
  - e.g. real-time communication through Skype, Voice over IP, online games, live streaming
  - but, not suitable for file download, Video-On-demand
- Lack of congestion control
  - Advantageous in error-prone network
- It is simple
  - suitable for bootstrapping or other purposes without a full protocol stack,
  - e.g., the DHCP
- It is stateless
  - suitable for very large numbers of clients, such as in streaming media applications such as IPTv.
- It works well in unidirectional communication
  - suitable for broadcast information
  - e.g, many kinds of service discovery

Thanks!

Content of this PPT are taken from:

1) **Computer Networks: A Top Down Approach**, by J.F. Kuros and K.W. Ross, 6th Eds, 2013, Pearson Education.

2) **Data Communications and Networking**, by B. A. Forouzan , 5th Eds, 2012, McGraw-Hill.

3) **Chapter 3 : Transport Layer,** PowerPoint slides of "Computer Networking: A Top Down Approach", 6th Eds, J.F. Kurose, K.W. Ross