BRAC UNIVERSITY

Inspiring Excellence

Introduction to Transport Layer

Lecture 4 | CSE421 – Computer Networks

Department of Computer Science and Engineering
School of Data & Science

Our goals: Objectives

- understand principles behind transport layer services
- learn about two transport layer protocols:
  - UDP: User Datagram Protocol
  - TCP: Transmission Control Protocol

Transport vs. Network layer

- ***transport layer:*** logical communication between processes
- ***network layer:*** logical communication between hosts

- Transport Layer PDU is called **Segments**

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

Functions of the Transport Layer

- Primary responsibilities:

1. Segmenting the data and managing each piece.
2. Reassembling the segments into streams of application data.
3. Identifying the different applications.
4. Multiplexing
5. Initiati[Reliability]ssion.
6. Performing flow control between end users.
7. Enabling error recovery.

Different Applications

Different Requirements

Web Applications

Emails

Online Games

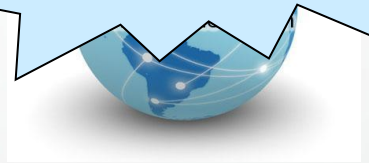
- Some applications need their data to be complete with no errors or gaps and they can accept a slight delay to ensure this.
- Some applications can accept occasional errors or gaps in the data but they cannot accept any delay.

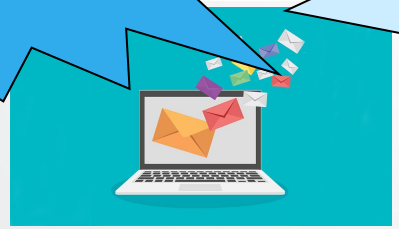
Solution : Two transport protocols?

Web Applications

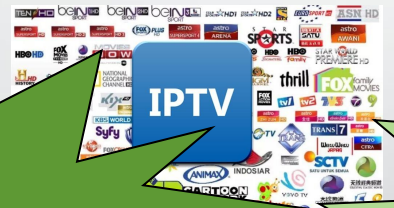
Reliable

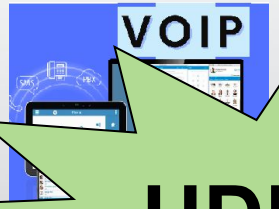
TCP





Emails





Fast

UDP

Online Games

UDP: User Datagram Protocol [RFC 768]

BRAC UNIVERSITY
Inspiring Excellence

UDP Protocol

- "no frills," "bare bones" transport layer protocol
- "best effort" service

why is there a UDP?

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

User Datagram Protocol (UDP)

- UDP is used by:
  - streaming multimedia apps (loss tolerant, rate sensitive)
  - SNMP

  But sometimes ……
  - DNS
  - HTTP/3

- reliable transfer over UDP:
  - add reliability at application layer
  - application-specific error recovery!

Functions of the Transport Layer

- Primary responsibilities:

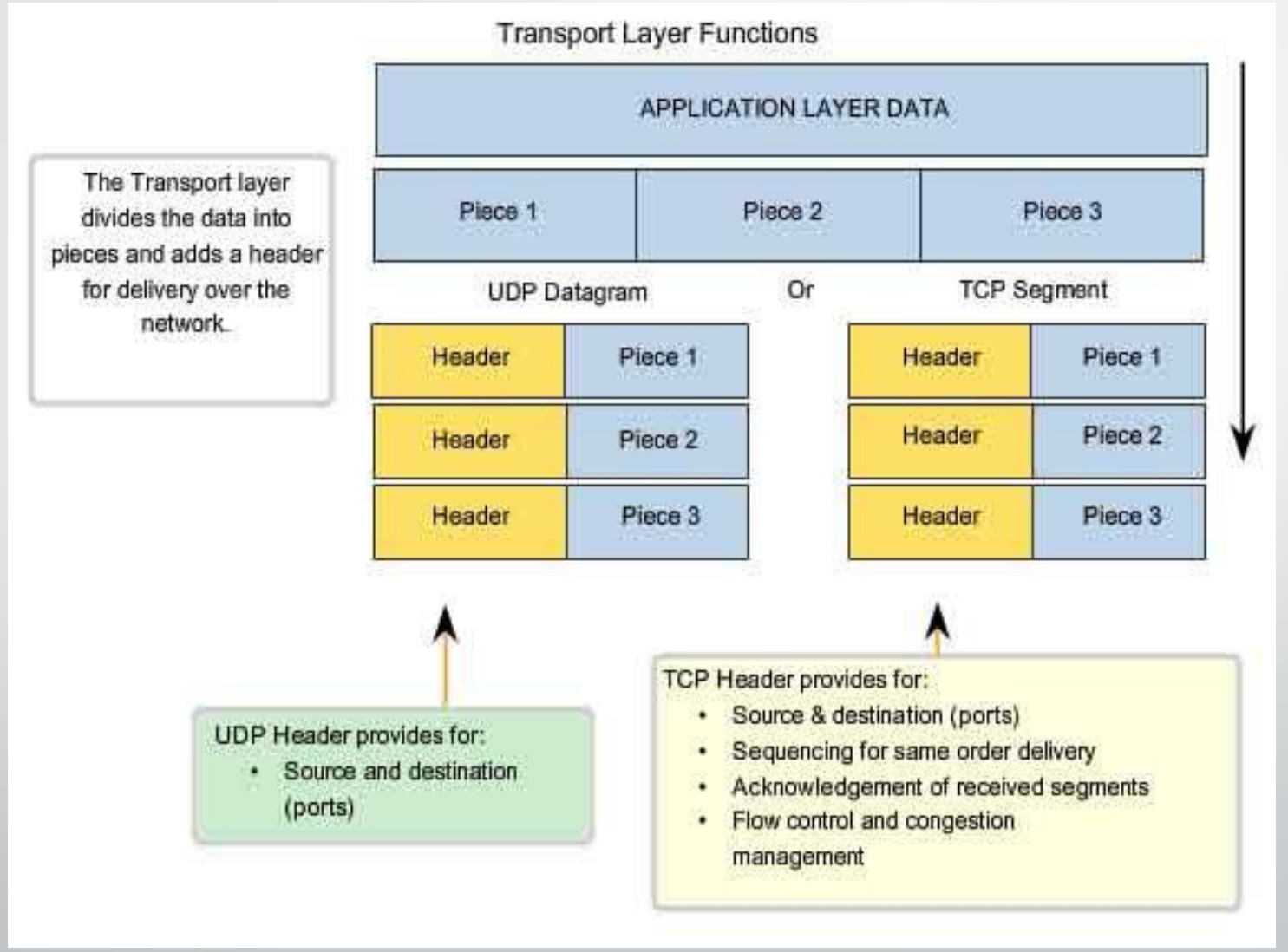
UDP &TCP

1. Segmenting the data and managing each piece.
2. Reassembling the segments into streams of application data.
3. Identifying the different applications.
4. Multiplexing

Only TCP

5. Establishing and terminating a connection
6. Enabling error recovery.
7. Performing flow control between end users.

Function 1&2 – Segmentation and Reassembly



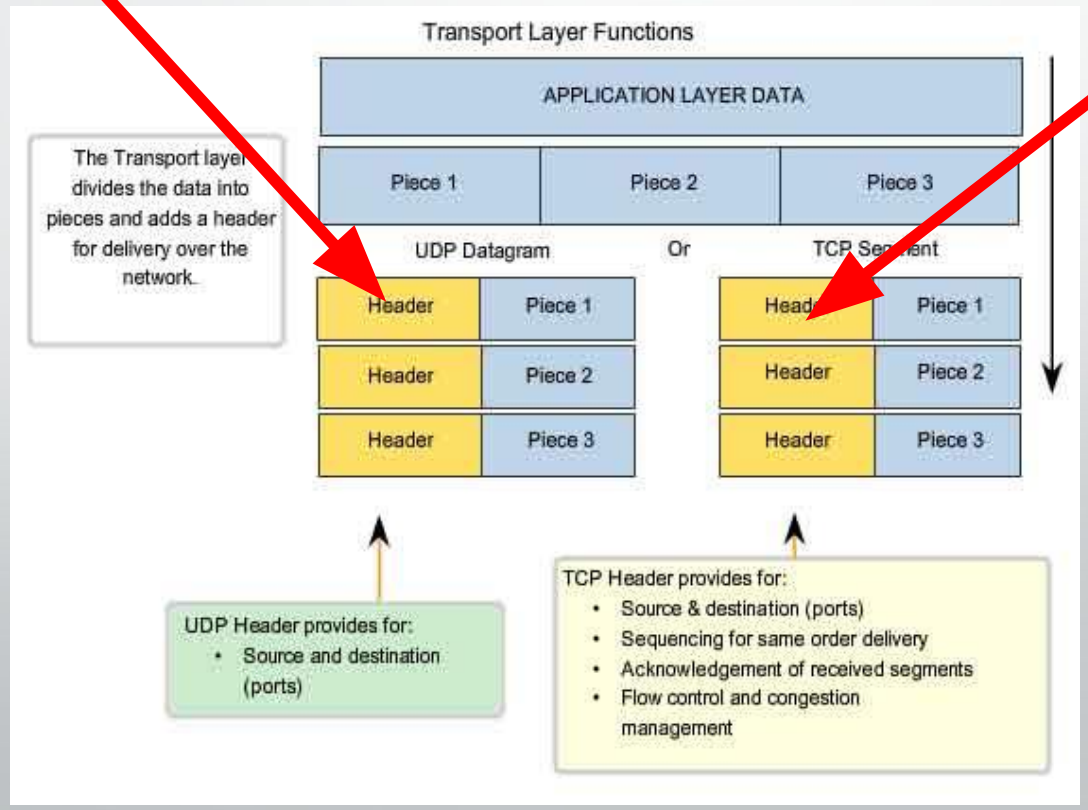
TCP and UDP Headers

UDP HEADER

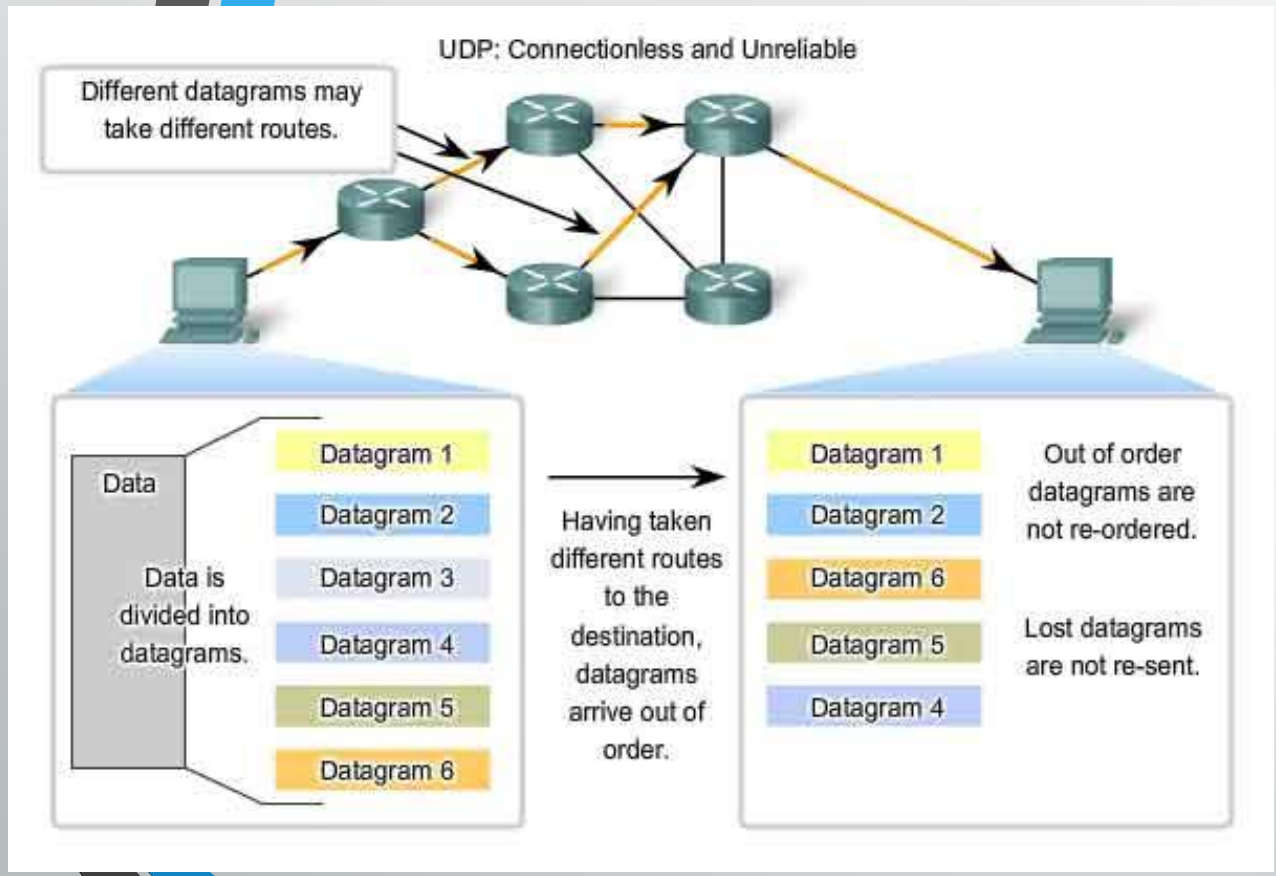
Source port number 16 bits	Destination port number 16 bits
Total length 16 bits	Checksum 16 bits

TCP HEADER

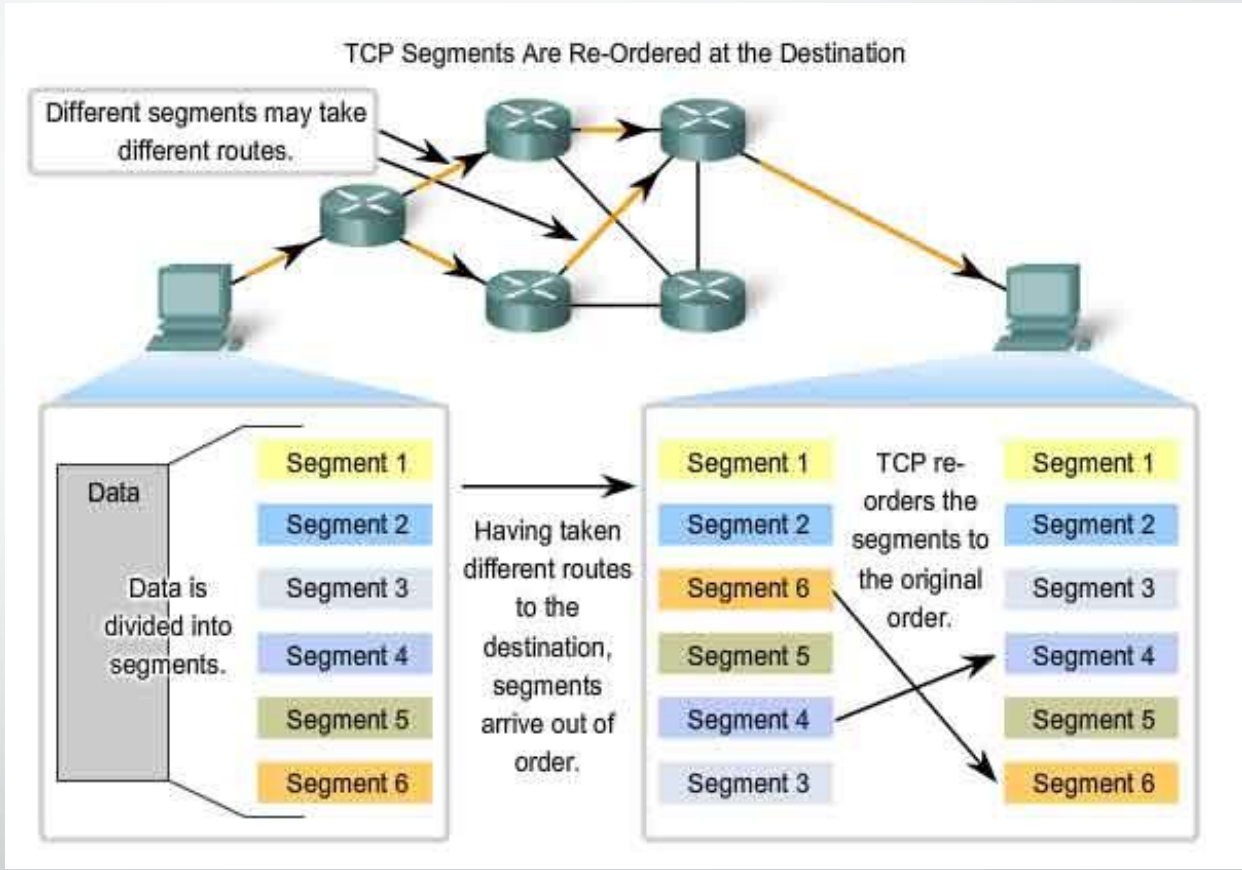
Source port address 16 bits								Destination port address 16 bits
Sequence number 32 bits								
Acknowledgment number 32 bits								
HLEN 4 bits	Reserved 6 bits	URG	ACK	PSH	RST	SYN	FIN	Window size 16 bits
Checksum 16 bits								Urgent pointer 16 bits
Options and Padding								



Function 2 – Reassembly

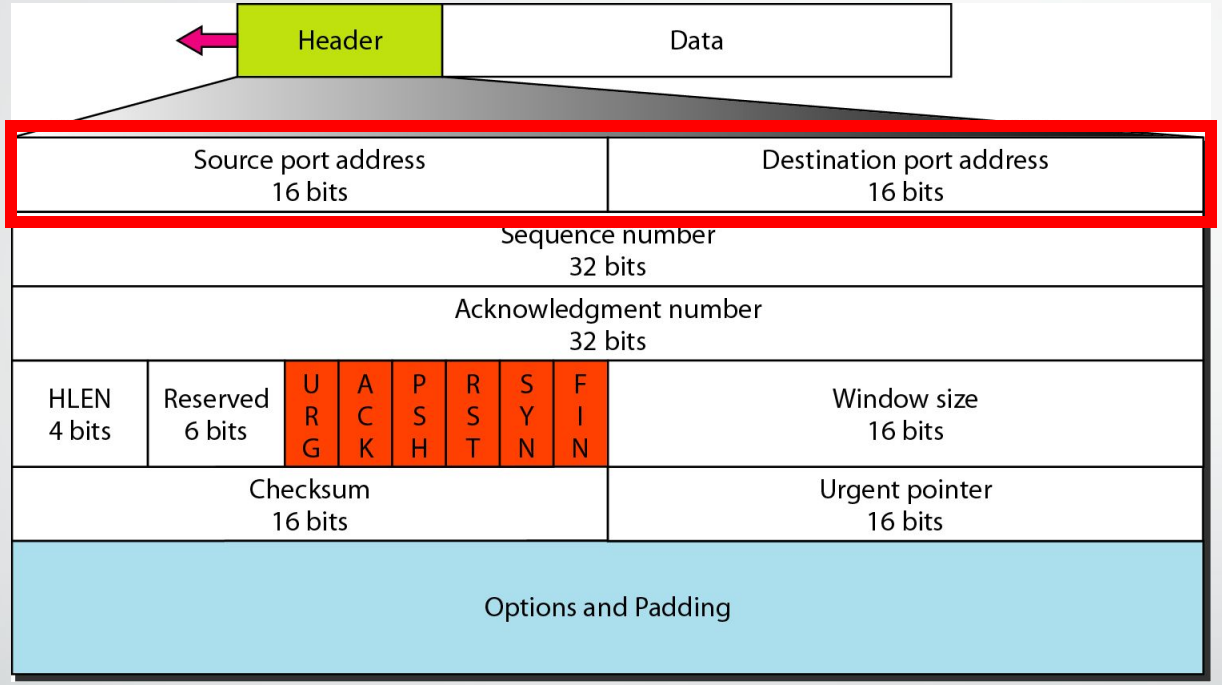


UDP

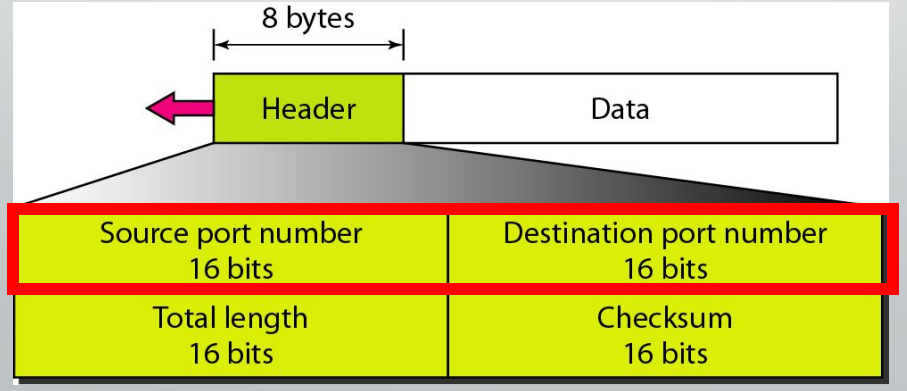


TCP

TCP and UDP Headers



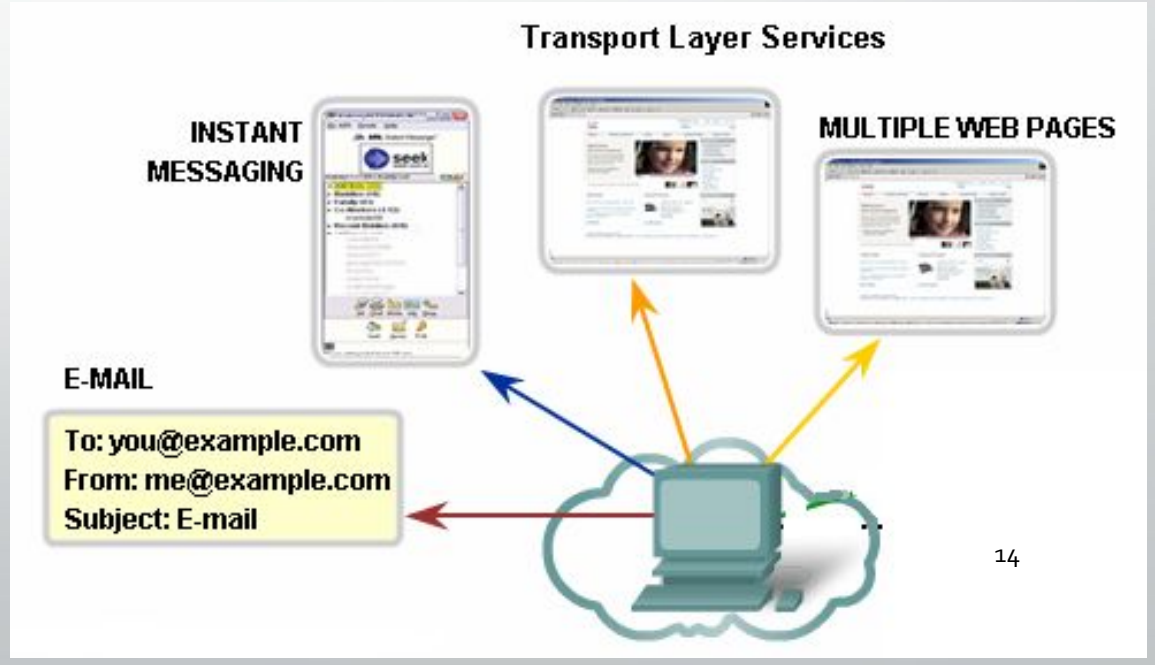
- TCP Header



- UDP Header

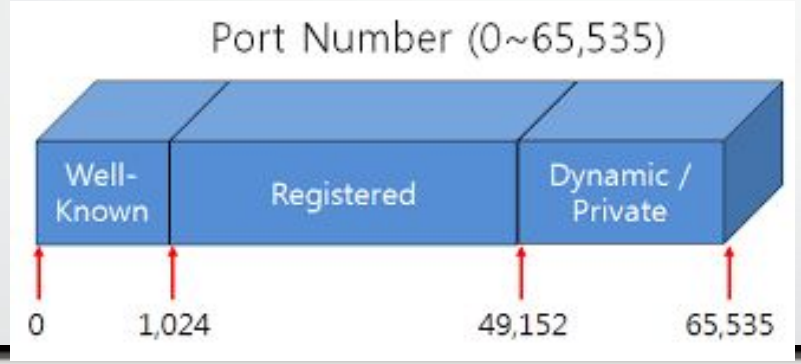
Function 3 – Identifying Different Applications

- Port Numbers/Addresses are used to identify different applications/processes running in a computer
- 16-bits in length
  - Represented as one single decimal number
  - Range **0 - 65535**
  - e.g. **80 – Web**; **25 – SMTP**
  - **110 – POP3**, **531 – Instant Messaging**



Port Numbers

- Internet Corporation for Assigned Names and Numbers (ICANN) assigns port numbers.
- **Three** categories:



Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

Port Number Types

- **Well-Known Ports:**
  - Reserved for common services and applications.
  - Assigned by IANA and controlled

Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

20&21 – FTP

25 – SMTP

443 – HTTPS

143 – IMAP

80 – HTTP

110 – POP3

23 – Telnet

53 - DNS

Port Number Types

- **Registered Ports:**
  - Not assigned or controlled by IANA
  - Can be registered as the default port for a lot of not-so-well-known, especially corporate/proprietary protocols.
  - Must request IANA

Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

8008 – Alternate HTTP

1863 – MSN Messenger

8080 – Alternate HTTP

5004 – RTP

5060 – SIP (VoIP)

Examples of Registered Ports

Table 10.2. Selected Registered UDP and TCP Ports with Service and Brief Description of Meaning

Port Number	Service	Brief Description of Use
1024	Reserved	Reserved for future use
1025	Blackjack	Network version of blackjack
1026	CAP	Calendar access protocol
1027	Exosee	ExoSee
1029	Solidmux	Solid Mux Server
1102	Adobe 1	Adobe Server 1
1103	Adobe 2	Adobe Server 2
44553	Rbr-debug	REALBasic Remote Debug
46999	Mediabox	MediaBox Server
47557	Dbbrowse	Databeam Corporation
48620–49150	Unassigned	These ports have not been registered
49151	Reserved	Reserved for future use

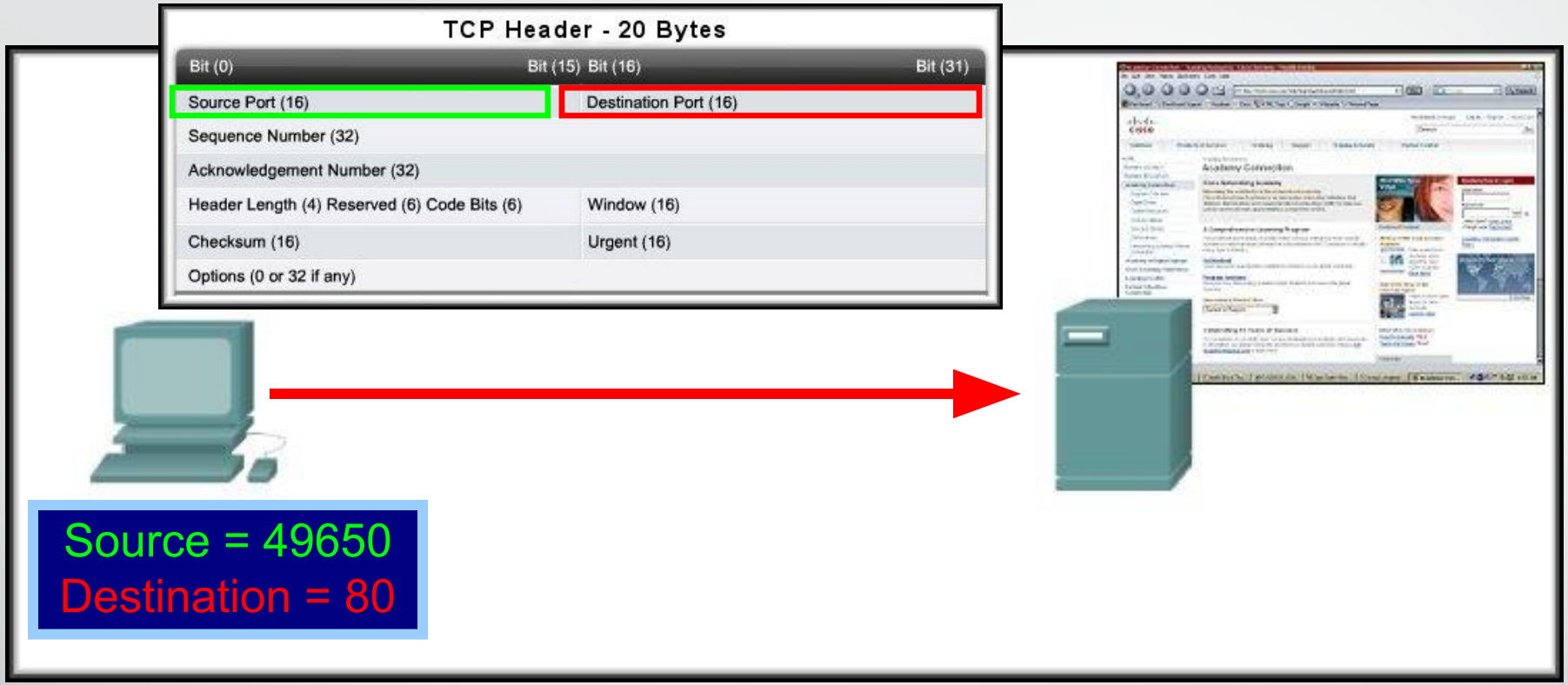
Port Number Types

- **Dynamic Ports:**
  - Also known as private or ephemeral ports
  - Never assigned or controlled by IANA.

Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

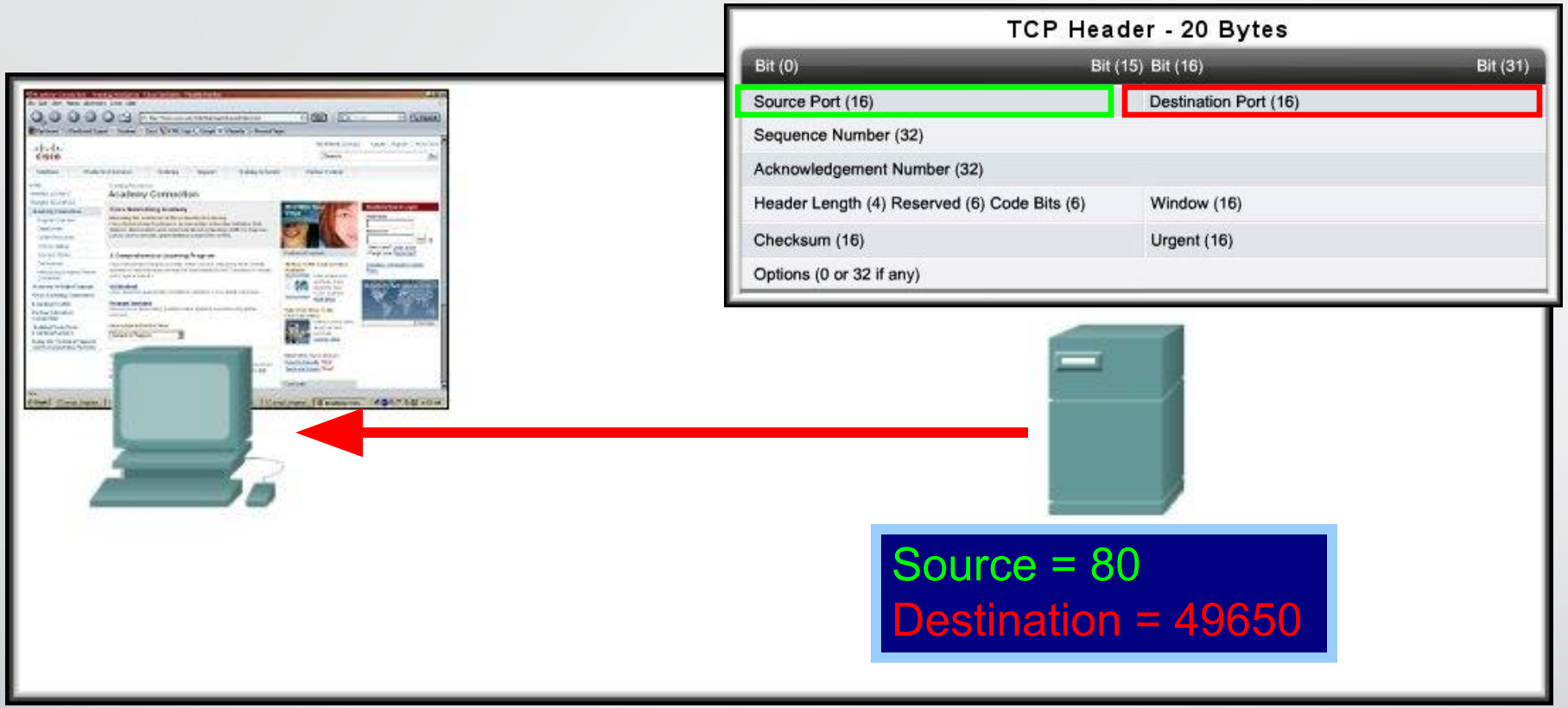
Dynamic port usage will become clearer as we move through the material.

More on Port Numbers



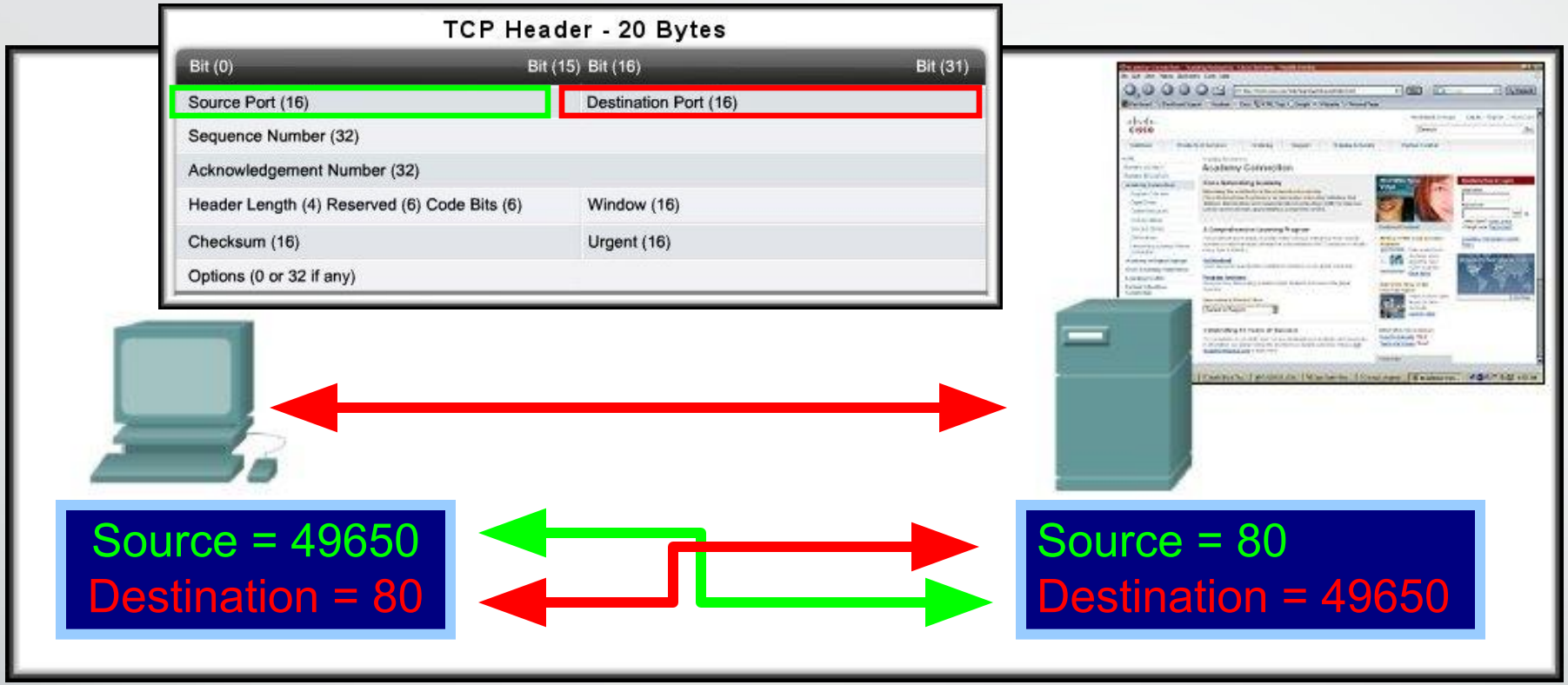
- Server is listening on Port 80 for HTTP connections.
- The client sets the destination port to 80 and uses a dynamic port as its source.

Port Numbers in Action



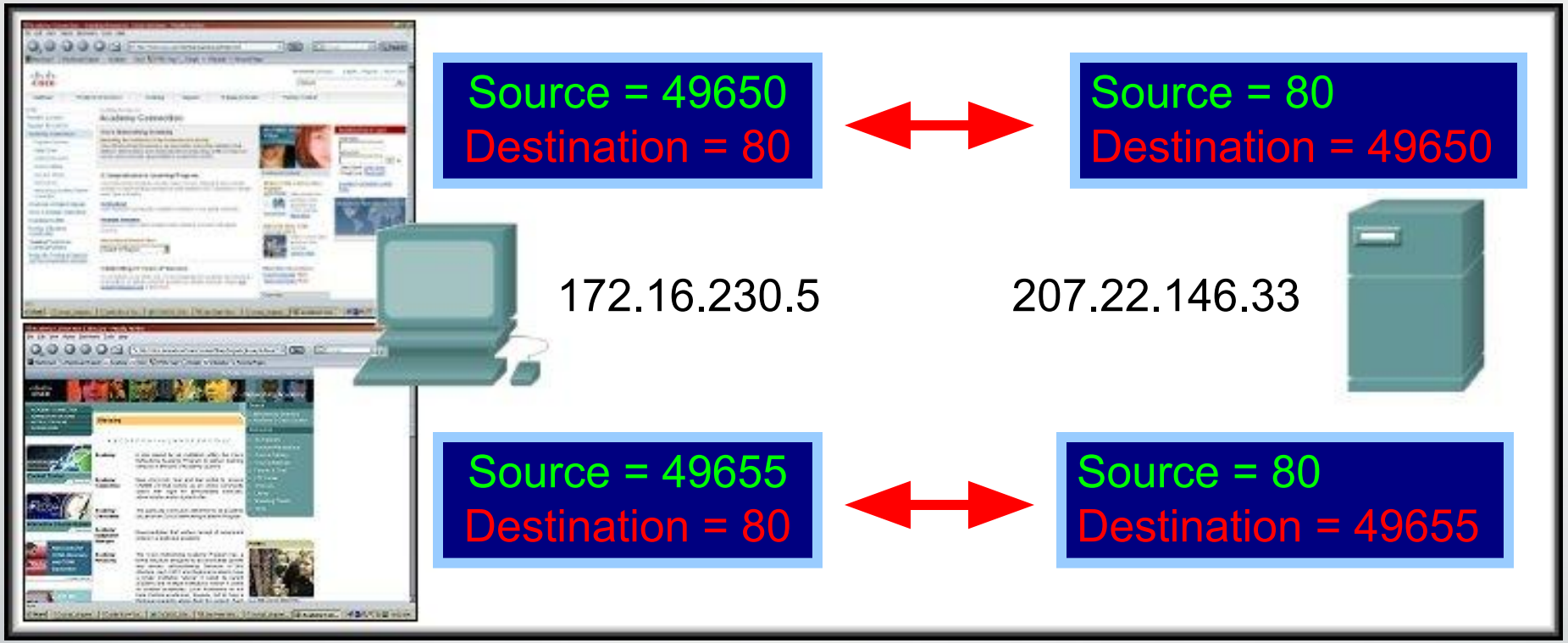
- Server replies with the web page.
  - Sets the source port to 80 and uses the client's source port as the destination.

Port Numbers in Action



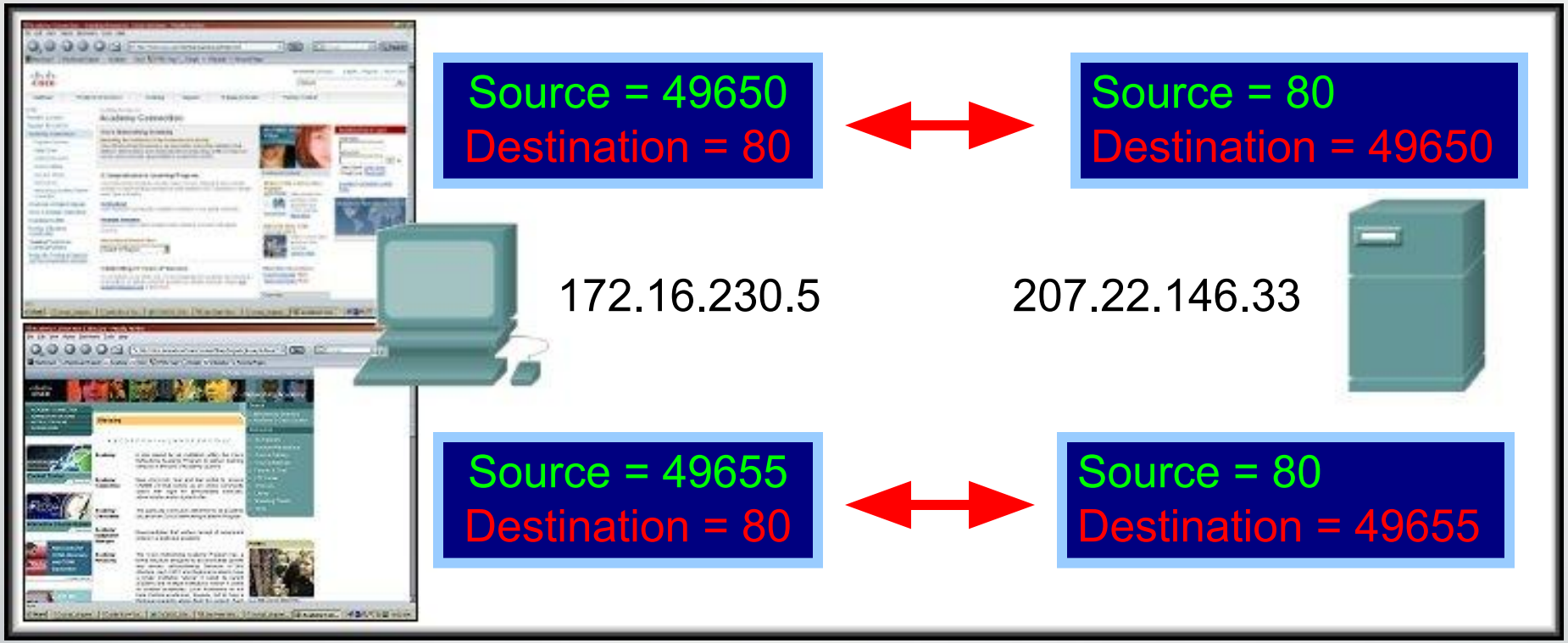
- Notice how the source and destination ports are used.
- Clients can use any random port number, servers can't.
  - Because clients won't be able to identify server process otherwise
- Servers, however, cannot use any random port number
  - Use of **well-known port numbers**!

Port Numbers in Action



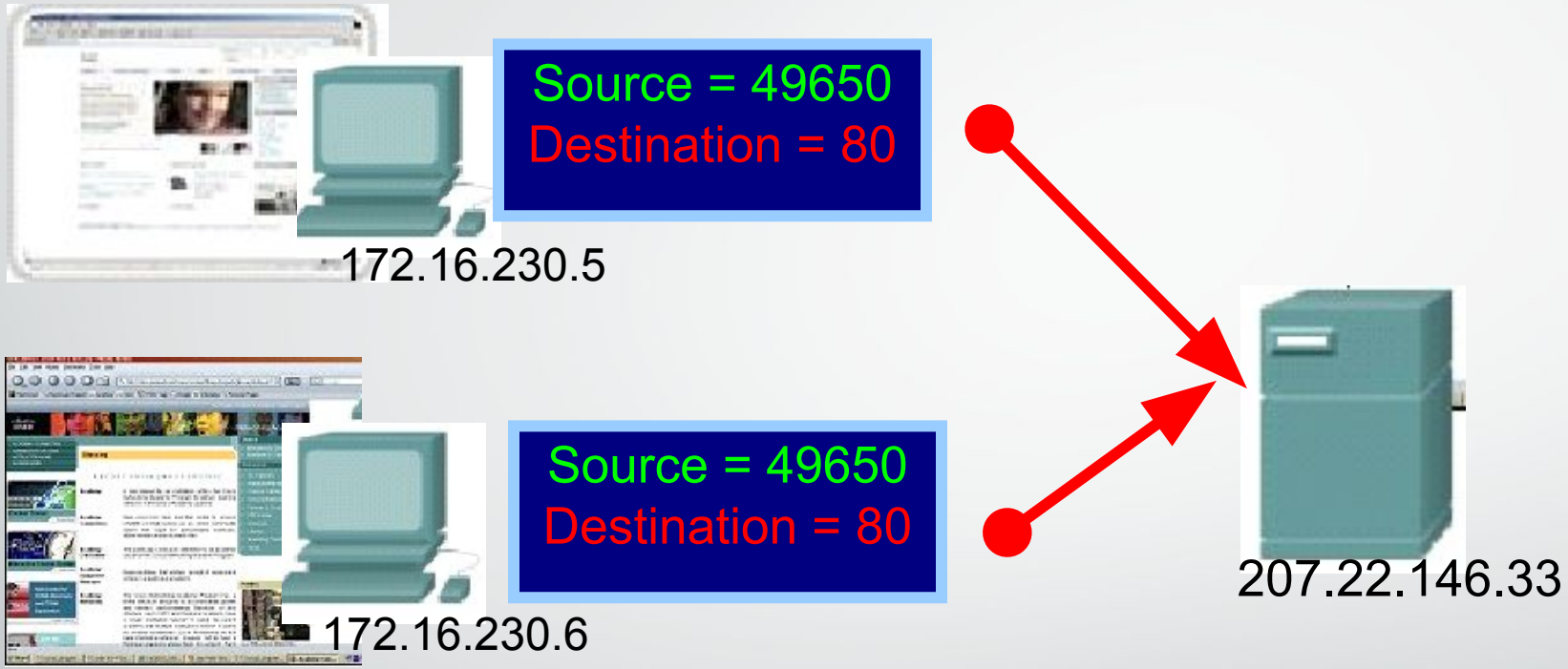
- What if there are two sessions to the same server?
  - The client uses another dynamic port as its source and the destination is still port 80.
  - Different source ports keep the sessions unique on the server.

Port Numbers in Action



- What if there are two tabs in the same PC?
  - The client uses another dynamic port as its source and the destination is still port 80.
  - Different source ports keep the sessions unique.

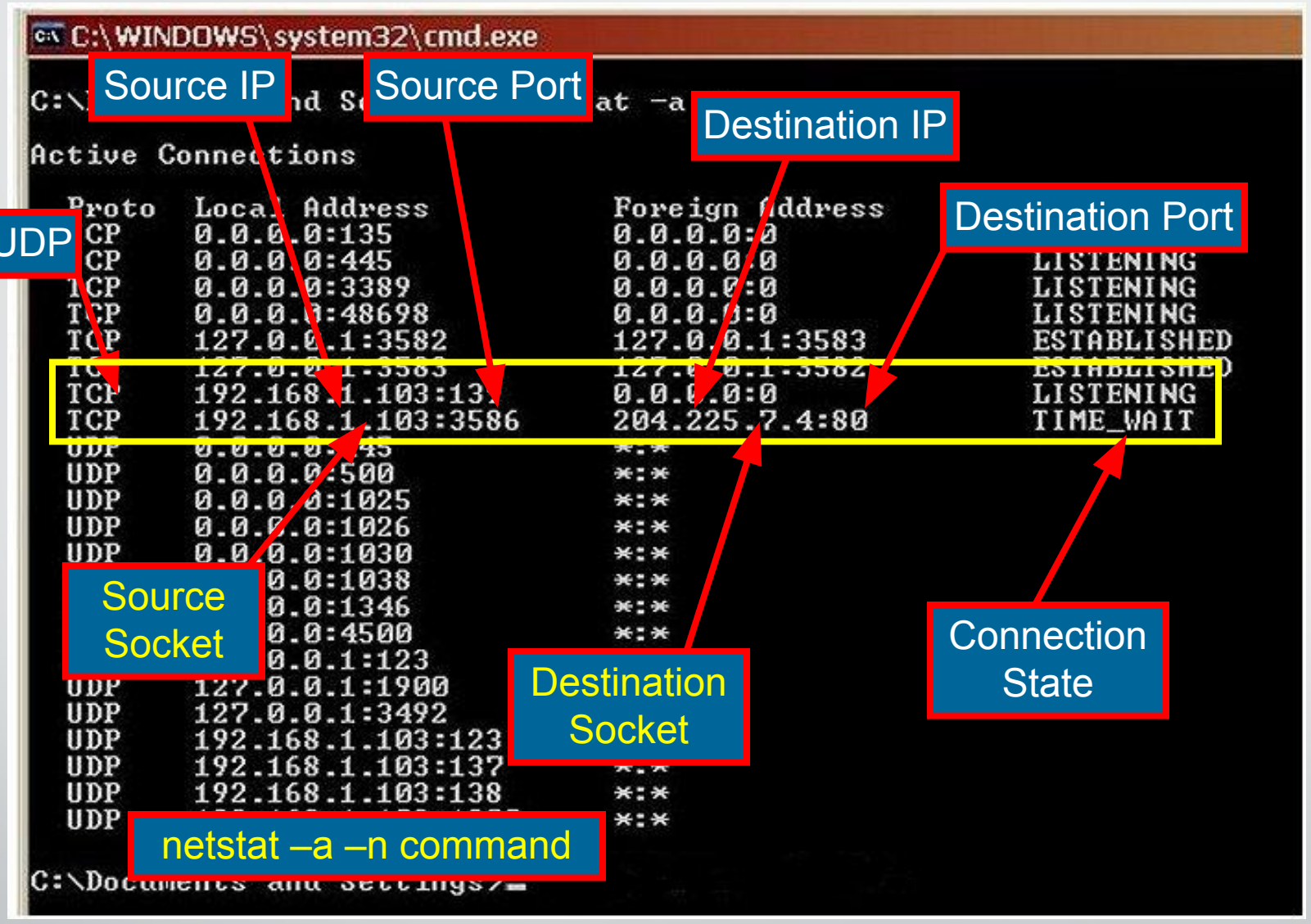
More on Port Numbers in Action



- How does the Server's Transport Layer keep them separate?
  - The socket (IP Address:Port)

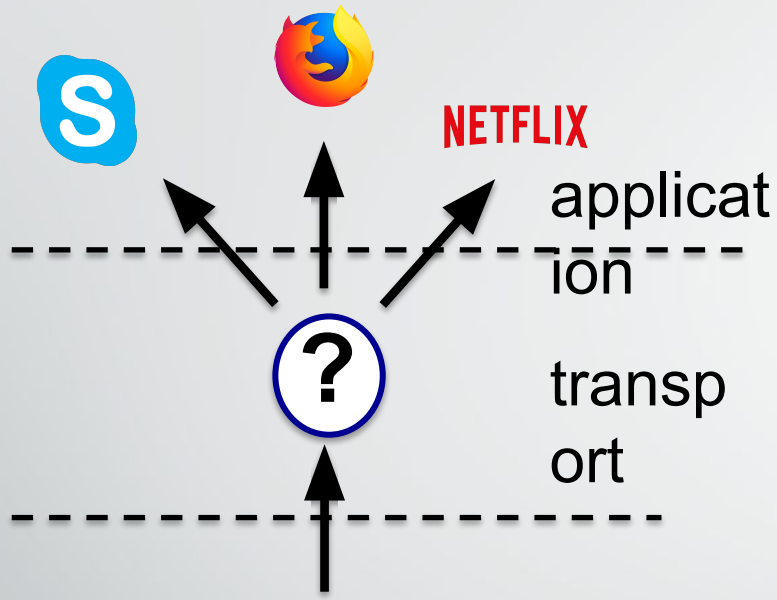
172.16.230.5:49650 ⟷ **207.22.146.33:80**

172.16.230.6:49650 ⟷ **207.22.146.33:80**



Netstat - Network Utility Tool

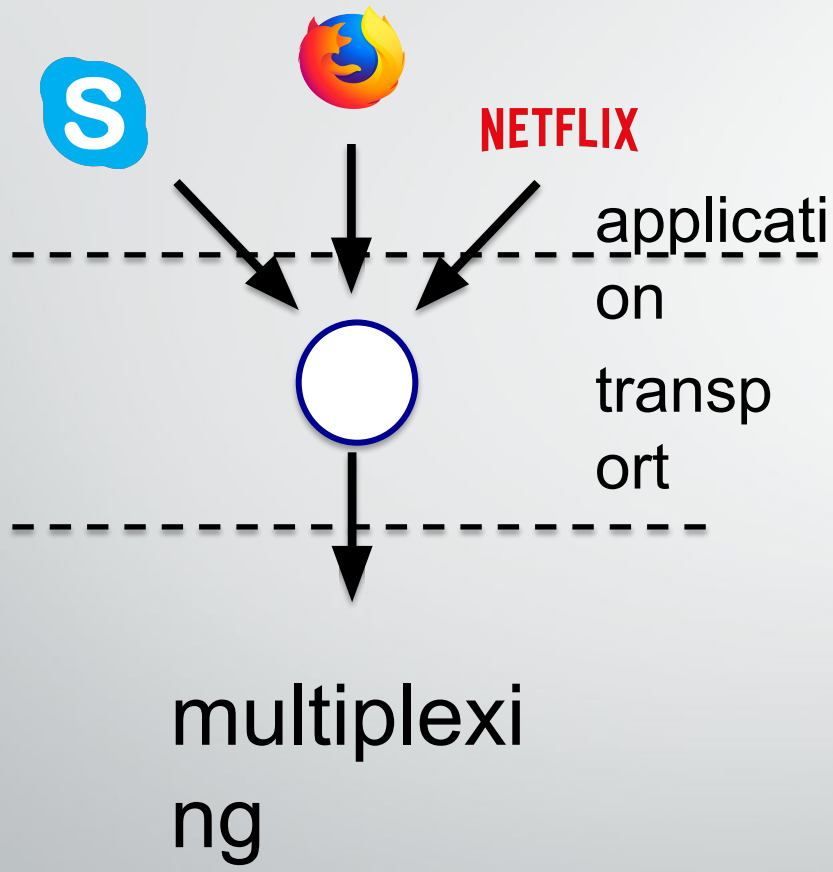
Function 4 –DeMultiplexing/ Multiplexing

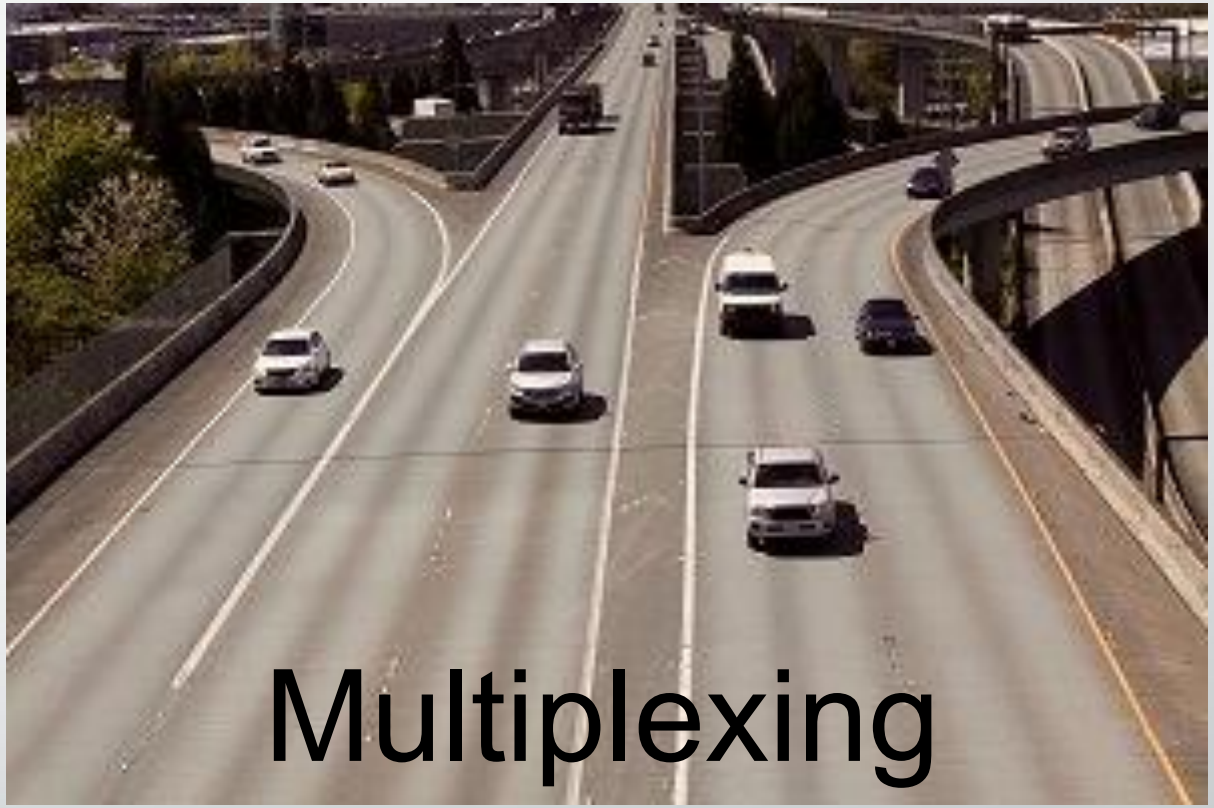


Demultiplexing

Demultiplexing

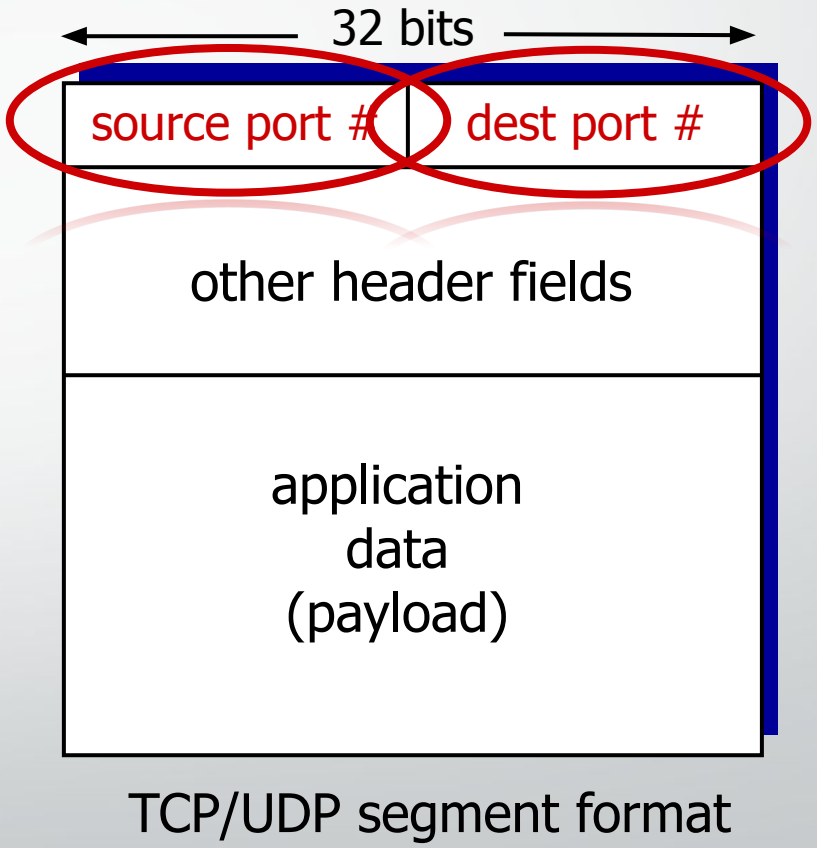
Function 4 –DeMultiplexing/ Multiplexing





How demultiplexing works

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



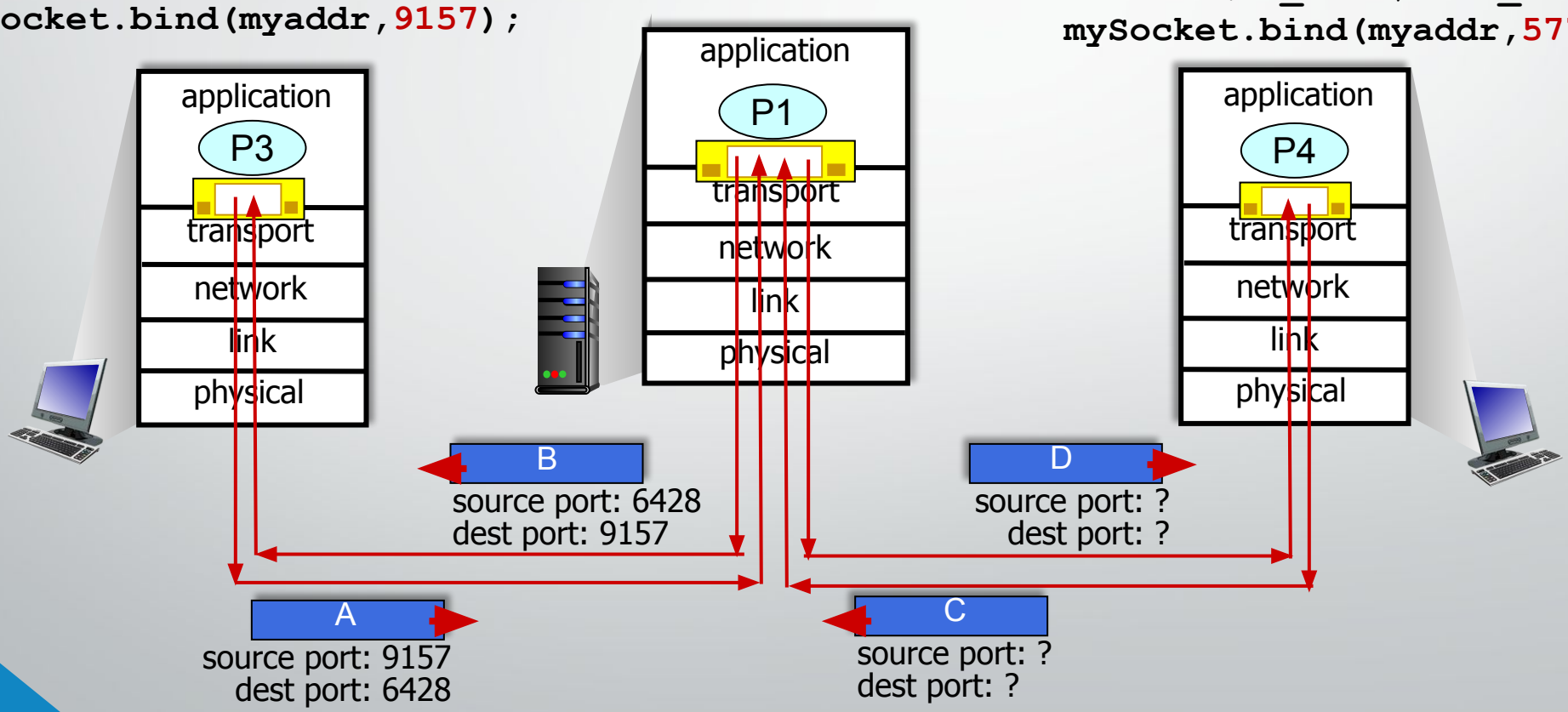
TCP/UDP segment format

UDP demultiplexing: an example

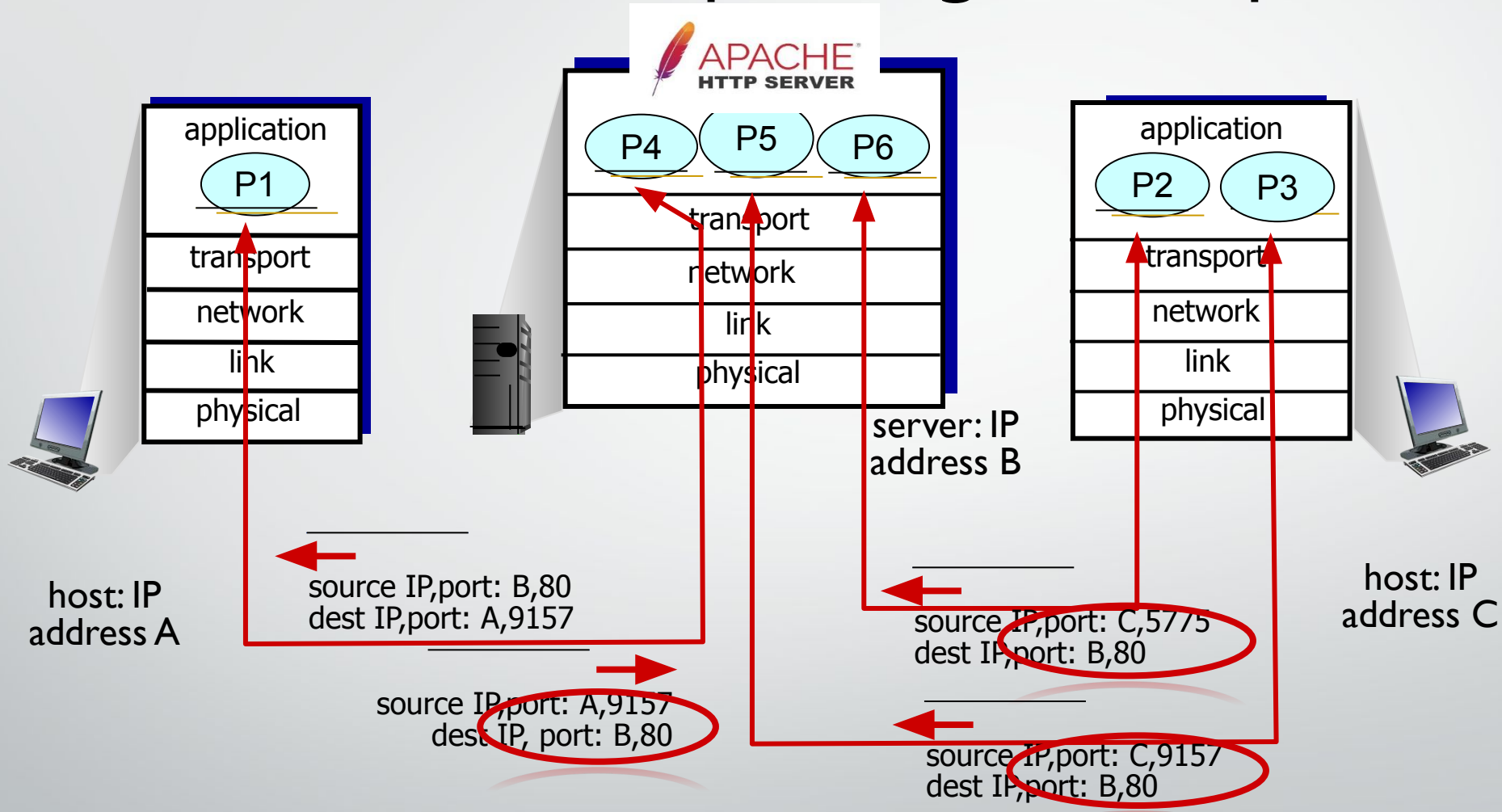
```
mySocket =
 socket(AF_INET,SOCK_DGRAM)
mySocket.bind(myaddr,6428);
```

```
mySocket =
 socket(AF_INET,SOCK_STREAM)
mySocket.bind(myaddr,9157);
```

```
mySocket =
 socket(AF_INET,SOCK_STREAM)
mySocket.bind(myaddr,5775);
```



TCP demultiplexing: example



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

DeMultiplexing/ Multiplexing

- Multiplexing, demultiplexing: based on segment, datagram header field values
- **UDP:** demultiplexing using destination port number (only)
- **TCP:** demultiplexing using 4-tuple: source and destination IP addresses, and port numbers
- Multiplexing/demultiplexing happen at *all* layers

And Now more on TCP!