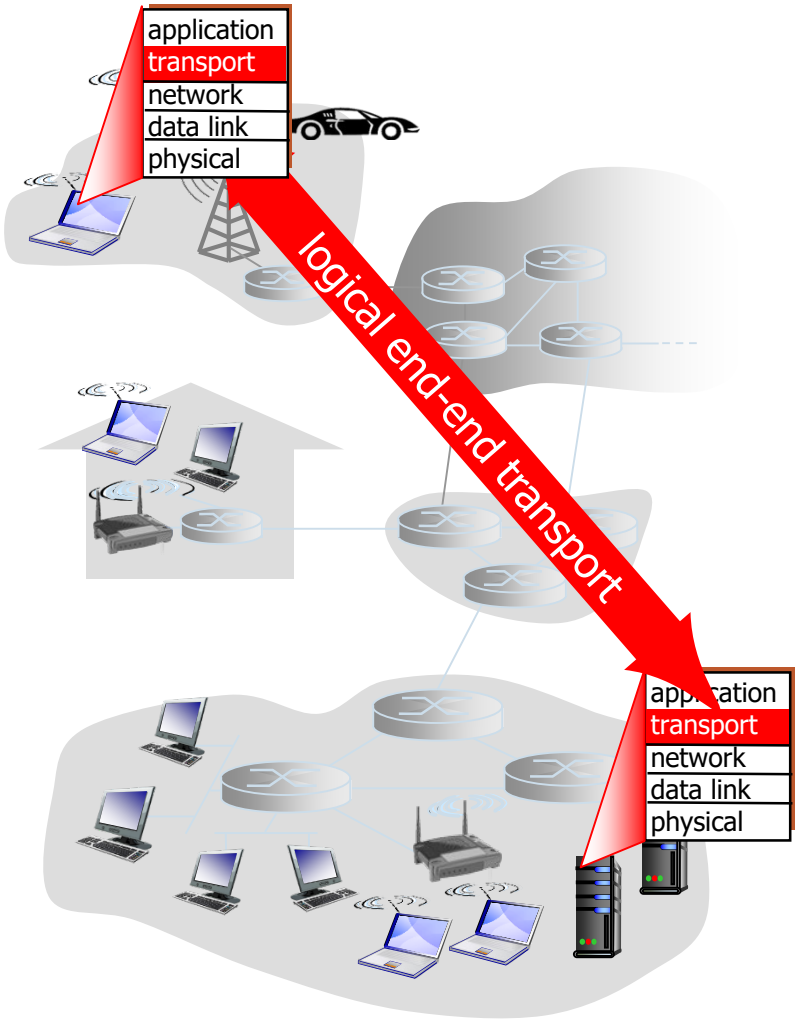
ECCS-3631
Networks and Data Communications

Module 5: Transport Layer and Reliable Communication
Module 5-1: Transport Layer, Multiplexing and Demultiplexing, Connection-less and Connection-oriented

Dr. Ajmal Khan

Introduction of Transport-Layer

- A transport-layer protocol provides for logical communication between application processes running on different hosts.
- Logical communication means that it is as if the hosts running the processes were directly connected. In reality, the hosts may be on opposite sides of the planet, connected via numerous routers and a wide range of link types.
- Transport-layer protocols are implemented in the end systems but not in network routers.



Introduction of Transport-Layer

- On the sending side, the transport layer converts the application-layer messages it receives from a sending application processes into transport-layer packets (known as transport-layer segments).
- It breaks the application messages into smaller chunks and adds a transport-layer header to each chunk to create the transport-layer segment. Then, it passes the segment to the network-layer.
- Internet has two transport-layer protocols; TCP and UDP

Internet Transport-Layer Protocol

- TCP/IP Network has two distinct transport-layer protocols;

TCP (Transmission Control Protocol) provide following services

- Reliable
- Connection-oriented service
- Connection Setup
- Flow control
- Congestion Control

UDP (User Datagram Protocol) provide following services

- Unreliable, and
- Connectionless

Services not available at Transport-Layer

- Delay guarantees
- Bandwidth guarantees

Multiplexing and Demultiplexing

At the destination host, the transport layer receives segments from the network layer. The transport layer has the responsibility of delivering the data in these segments to the appropriate application process running in the host.

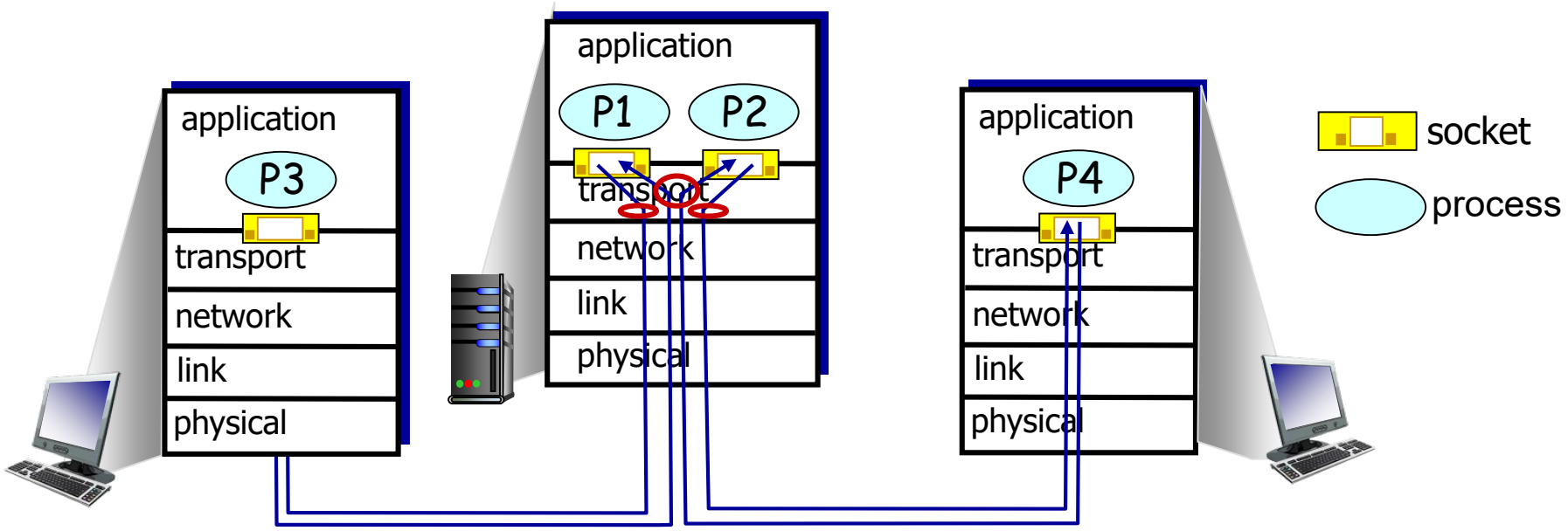
Suppose you are sitting in front of your computer, and you are downloading Web pages while running one FTP session and two Telnet sessions. You therefore have four network application processes running—two Telnet processes, one FTP process, and one HTTP process. When the transport layer in your computer receives data from the network layer, it needs to direct the received data to one of these four processes.

A process can have one or more sockets, doors through which data passes from the network to the process and through which data passes from the process to the network. Thus, the transport layer in the receiving host does not actually deliver data directly to a process, but instead to an intermediary socket. Because at any given time there can be more than one socket in the receiving host.

Multiplexing and Demultiplexing

Each transport layer segment has a set of fields in the segment. At the receiving end, the transport layer examines these fields to identify the receiving socket and then directs the segment to that socket. This job of delivering the data in a transport-layer segment to the correct socket is called **demultiplexing**.

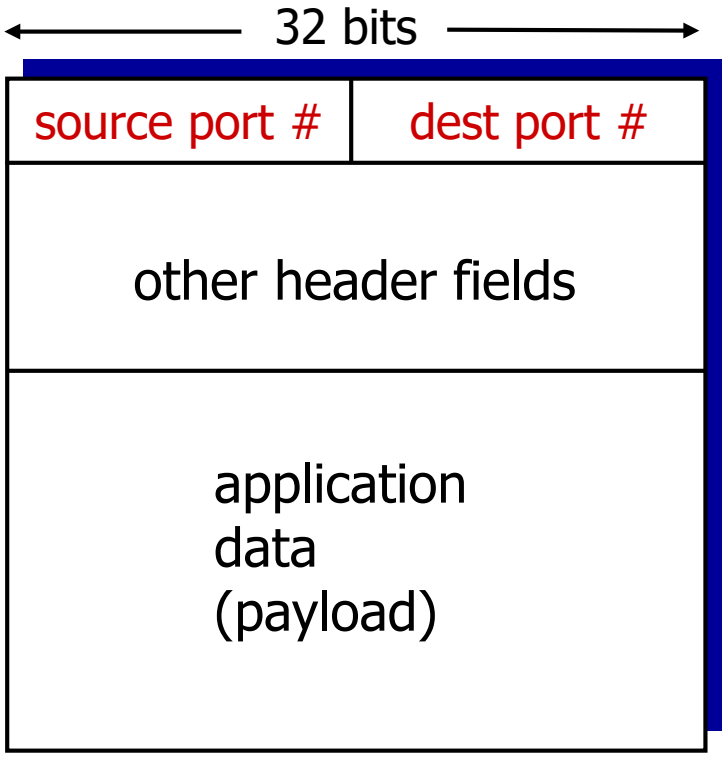
The job of gathering data chunks at the source host from different sockets, encapsulating each data chunk with header information to create segments, and passing the segments to the network layer is called **multiplexing**.



Multiplexing and Demultiplexing

Transport-layer multiplexing requires (1) that sockets have unique identifiers, and (2) that each segment have special fields that indicate the socket to which the segment is to be delivered. These special fields are the source port number field and the destination port number field.

Each port number is a 16-bit number, ranging from 0 to 65535. The port numbers ranging from 0 to 1023 are called well-known port numbers and are restricted, which means that they are reserved for use by well-known application protocols such as HTTP (uses port number 80) and FTP (uses port number 21).



TCP/UDP segment format

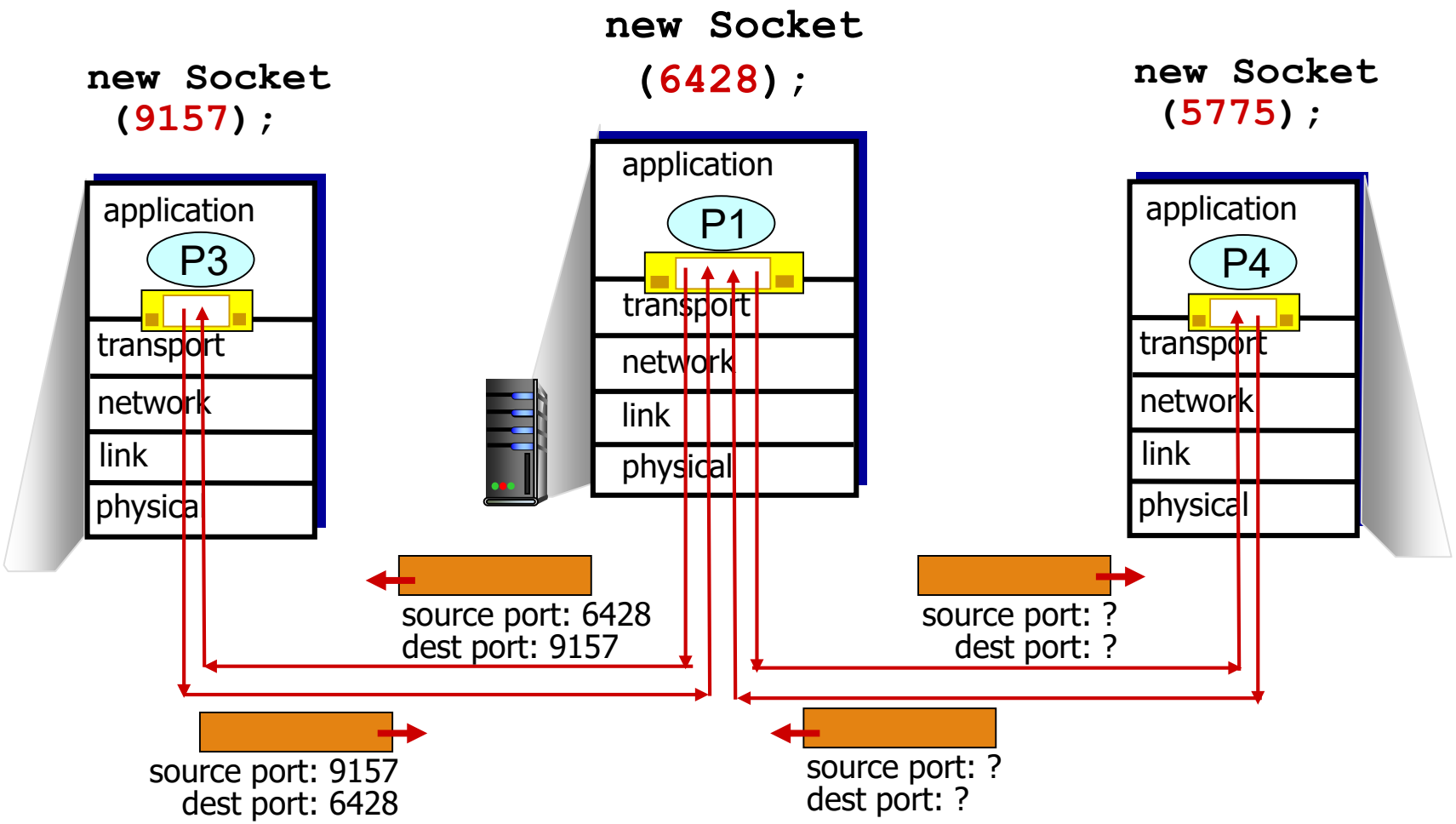
Connectionless Multiplexing and Demultiplexing

When a UDP socket is created by opening an application, the transport-layer automatically assigns a port number to the socket. In particular, the transport-layer assigns a port number in the range 1024 to 65535 that is currently not being used by any other UDP port in the host.

If the application developer writing the code were implementing the server side of a well-known protocol, then the developer would have to assign the corresponding well-known port number. Typically, the client side of the application lets the transport layer automatically (and transparently) assign the port number, whereas the server side of the application assigns a specific port number.

It is important to note that a UDP socket is fully identified by a **two-tuple** consisting of a destination IP address and a destination port number. As a consequence, if two UDP segments have different source IP addresses and/or source port numbers, but have the same destination IP address and destination port number, then the two segments will be directed to the same destination process via the same destination socket.

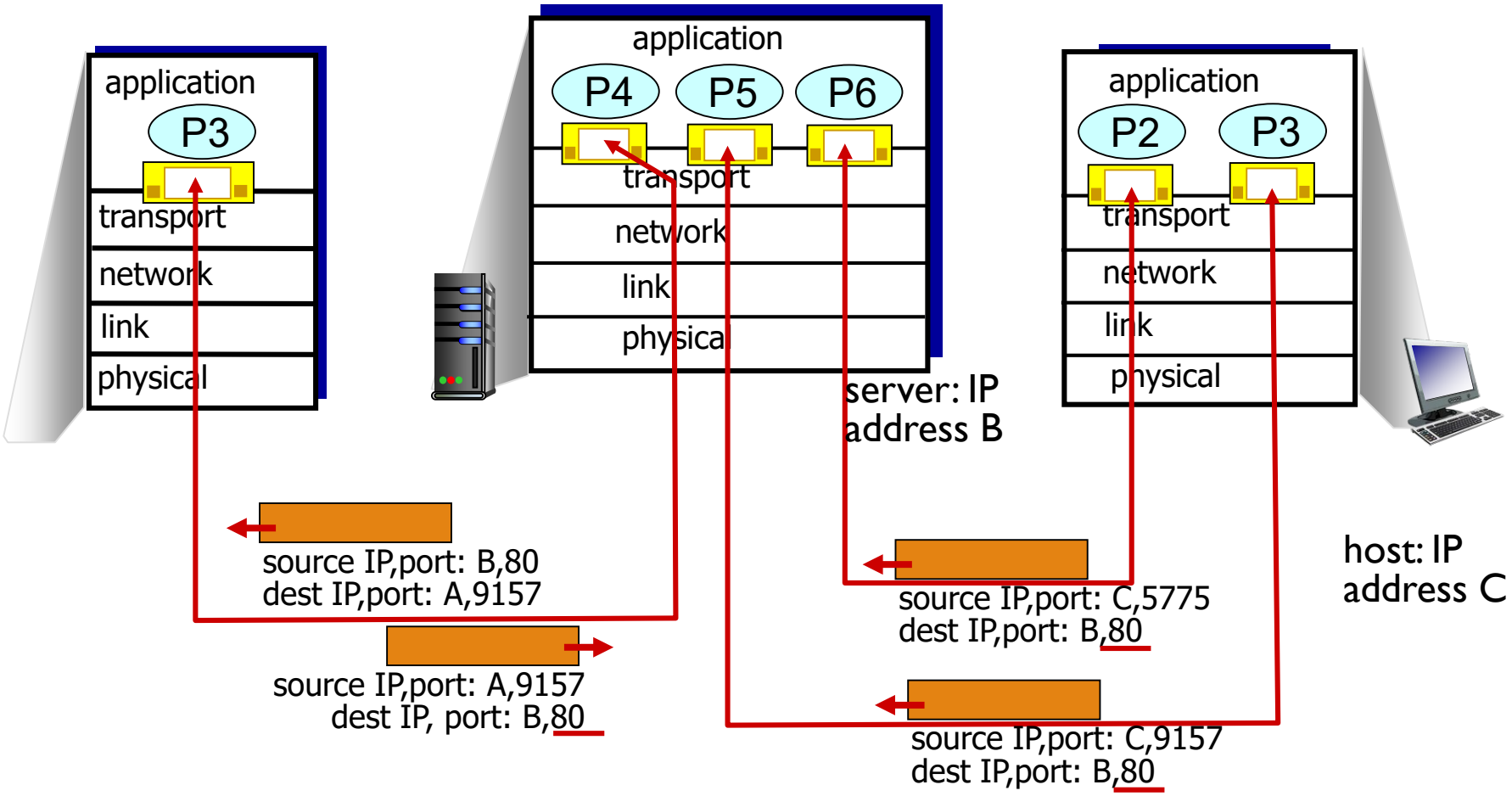
Connectionless Multiplexing and Demultiplexing



Connection-Oriented Multiplexing and Demultiplexing

Difference between a TCP socket and a UDP socket is that a TCP socket is identified by a **four-tuple**: (source IP address, source port number, destination IP address, destination port number). Thus, when a TCP segment arrives from the network to a host, the host uses all four values to direct (demultiplex) the segment to the appropriate socket. In particular, and in contrast with UDP, two arriving TCP segments with different source IP addresses or source port numbers will (with the exception of a TCP segment carrying the original connection-establishment request) be directed to two different sockets.

Connection-Oriented Multiplexing and Demultiplexing



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

List of Applications that use UDP or TCP

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

Practice Problem 1

Suppose Client A initiates a Telnet session with Server S. At about the same time, Client B also initiates a Telnet session with Server S. Provide possible source and destination port numbers for

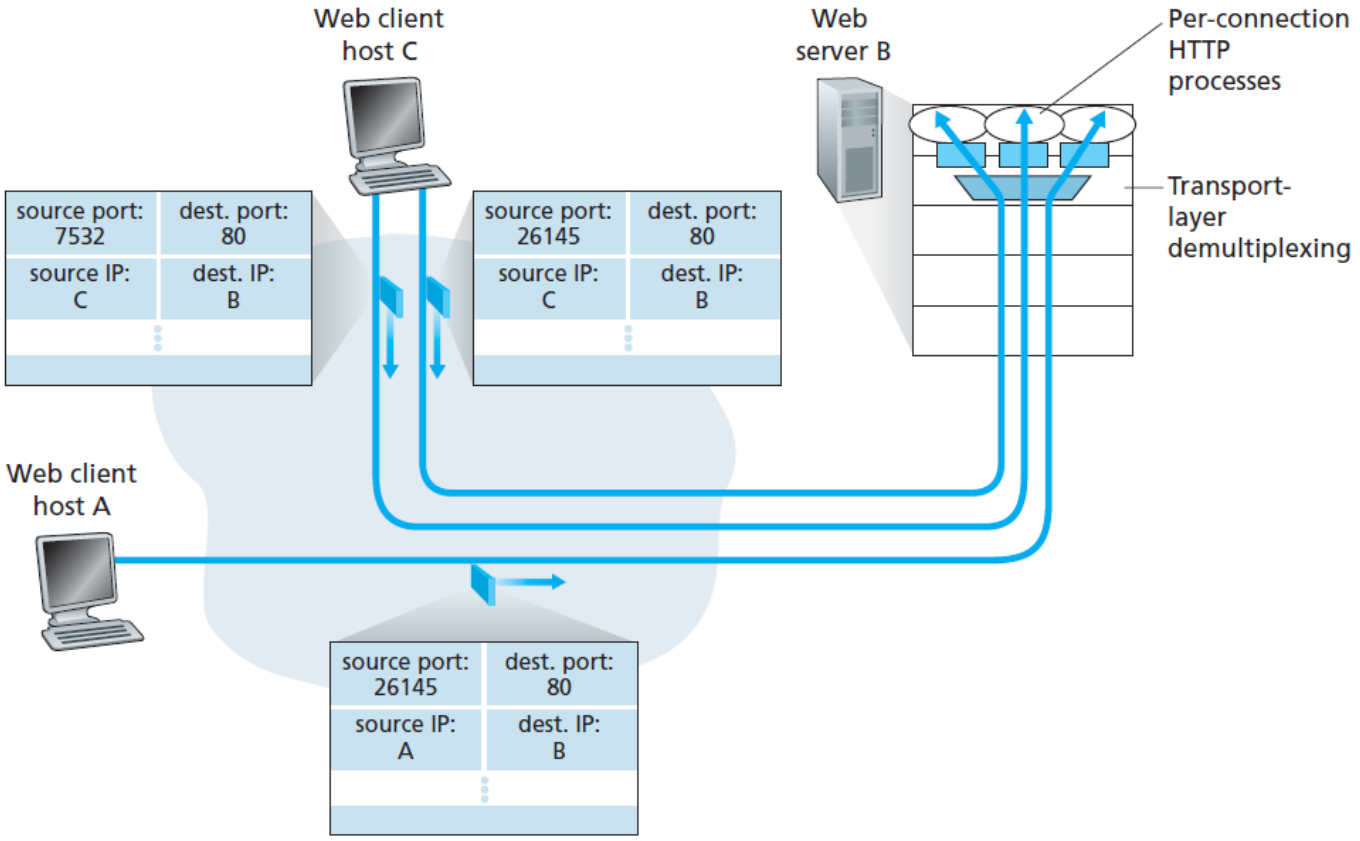
(a) The segments sent from A to S.
(b) The segments sent from B to S.
(c) The segments sent from S to A.
(d) The segments sent from S to B.
(e) If A and B are different hosts, is it possible that the source port number in the segments from A to S is the same as that from B to S?
(f) How about if they are the same host?

	source port numbers	destination port numbers
a) A → S	467	23
b) B → S	513	23
c) S → A	23	467
d) S → B	23	513

e) Yes.
f) No.

Practice Problem 2

Consider the following Figure. What are the source and destination port values in the segments flowing from the server back to the clients' processes? What are the IP addresses in the network-layer datagrams carrying the transport-layer segments?



Two clients, using the same destination port number (80) to communicate with the same Web server application

Practice Problem 2

Suppose the IP addresses of the hosts A, B, and C are a, b, c, respectively.

To host A: Source port =80, source IP address = b, dest port = 26145, dest IP address = a

To host C, left process: Source port =80, source IP address = b, dest port = 7532, dest IP address = c

To host C, right process: Source port =80, source IP address = b, dest port = 26145, dest IP address = c