

CMSC 332

Computer Networks

Transport Layer

Professor Szajda

Announcements

- Project 3 - I will be posted soon. Be on the lookout!
- Project 2 will be graded Friday. I'll look at
 - Source
 - Correct conventions
 - Correct source code title!
 - Documentation!!!!!!!!!!
 - Functionality



Chapter 3: Transport Layer

Our goals:

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

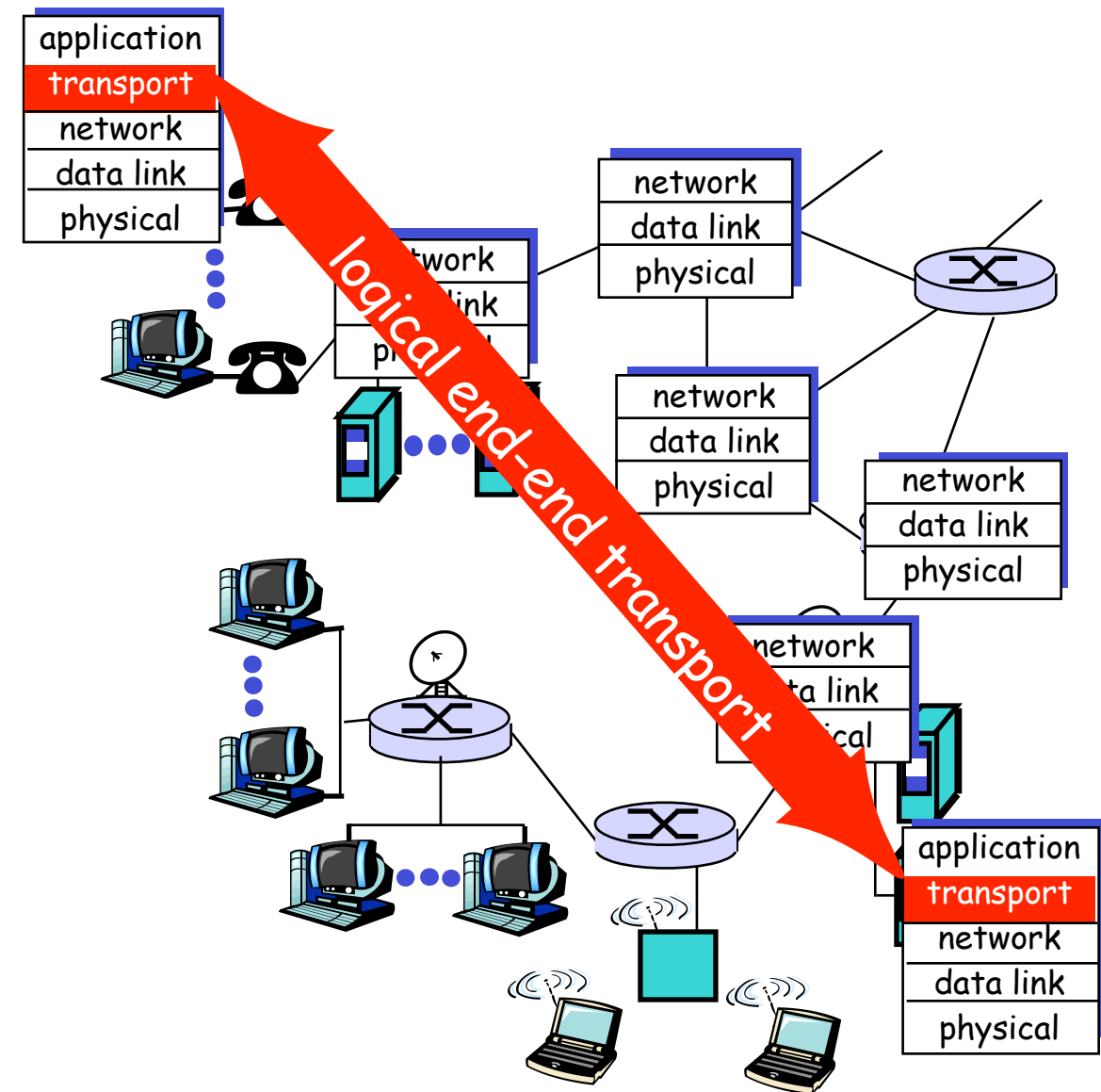


Chapter 3 Outline

- 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 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
 - rcv 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 sending letters to 12 kids

- processes = kids
- app messages = letters in envelopes
- hosts = houses
- transport protocol = Ann and Bill
- network-layer protocol = postal service

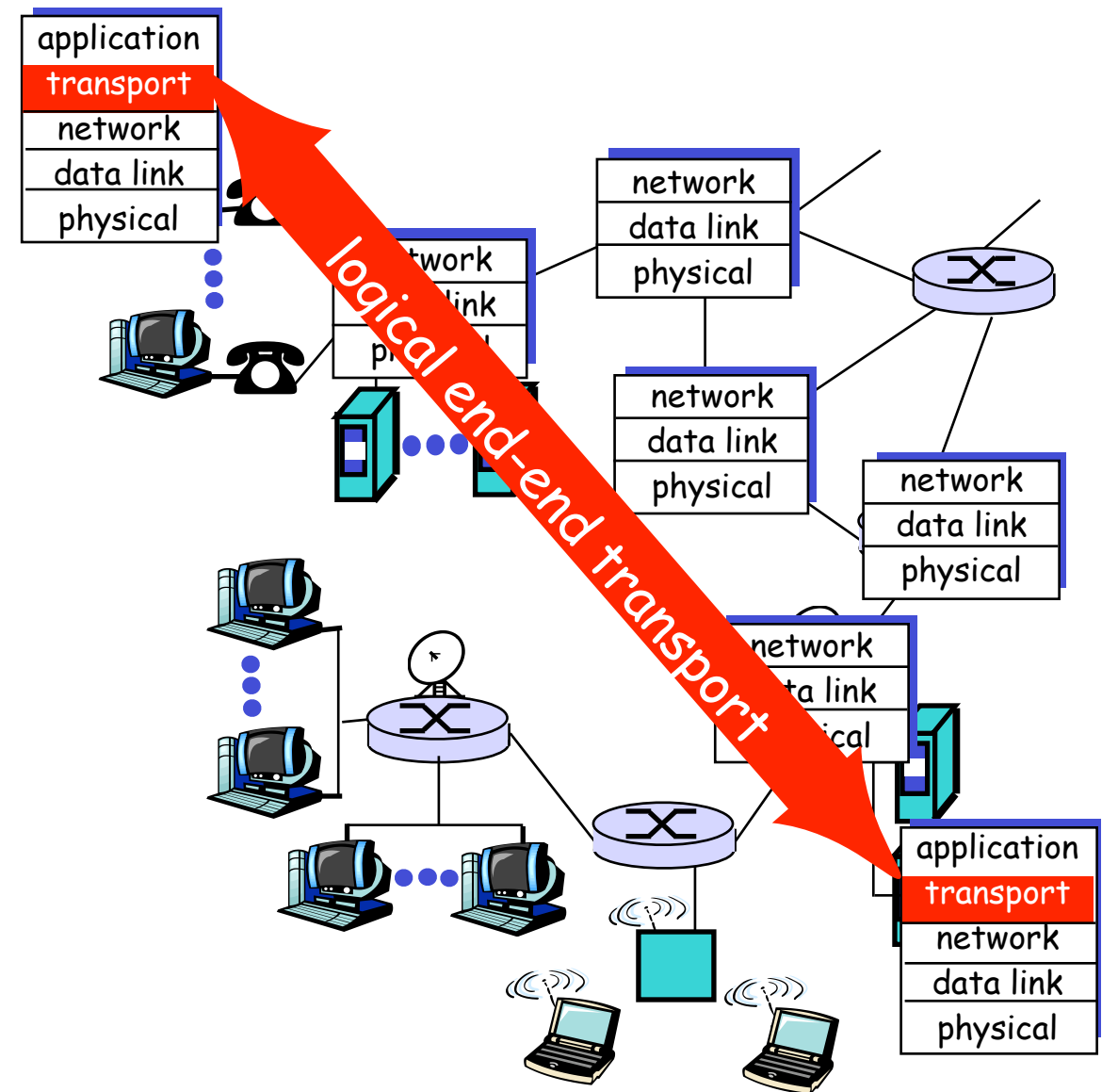
Layers of Networks?

- You can view each layer that we have discussed thus far as an abstract network:
 - Application Layer Networks: P2P, Social Networks, etc
 - Transport Layer Networks: Communicating processes
 - Network Layer Networks: Networks of Hosts
 - Link Layer Networks: One-Hop Networks
 - Physical Layer Networks: Wires



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

- 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

Multiplexing/demultiplexing

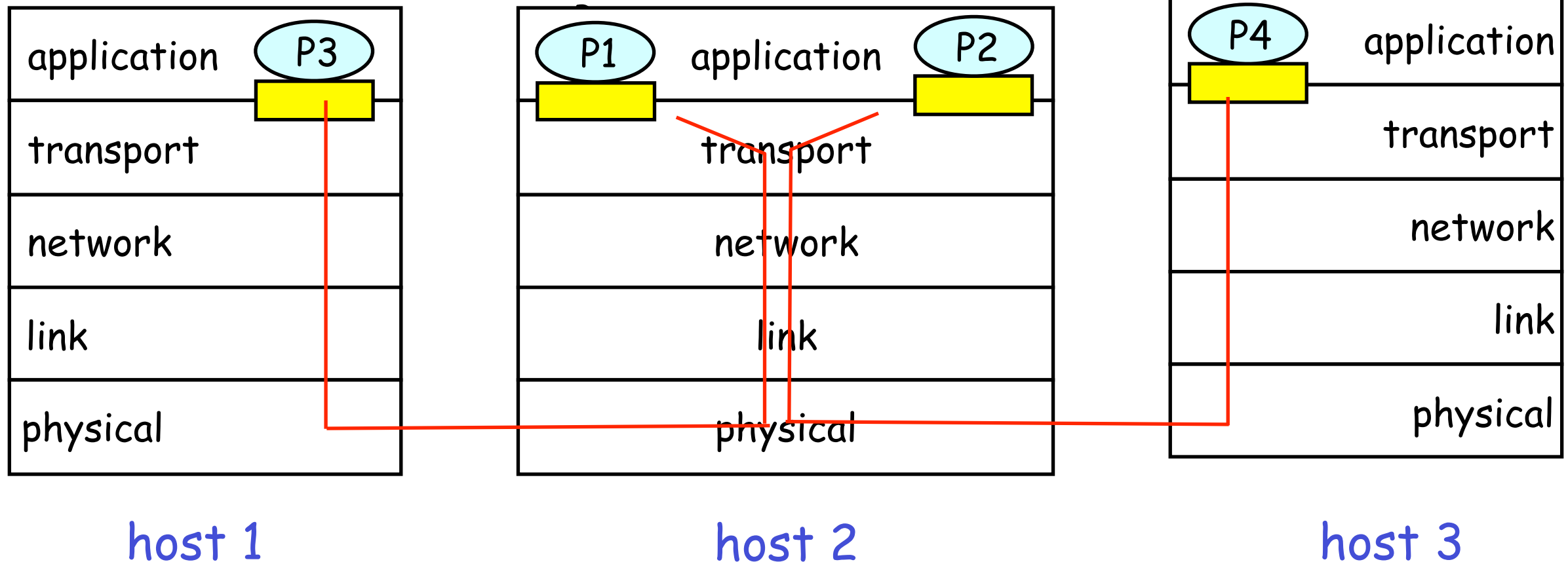
Demultiplexing at rcv host:

delivering received segments
to correct socket

Multiplexing at send host:

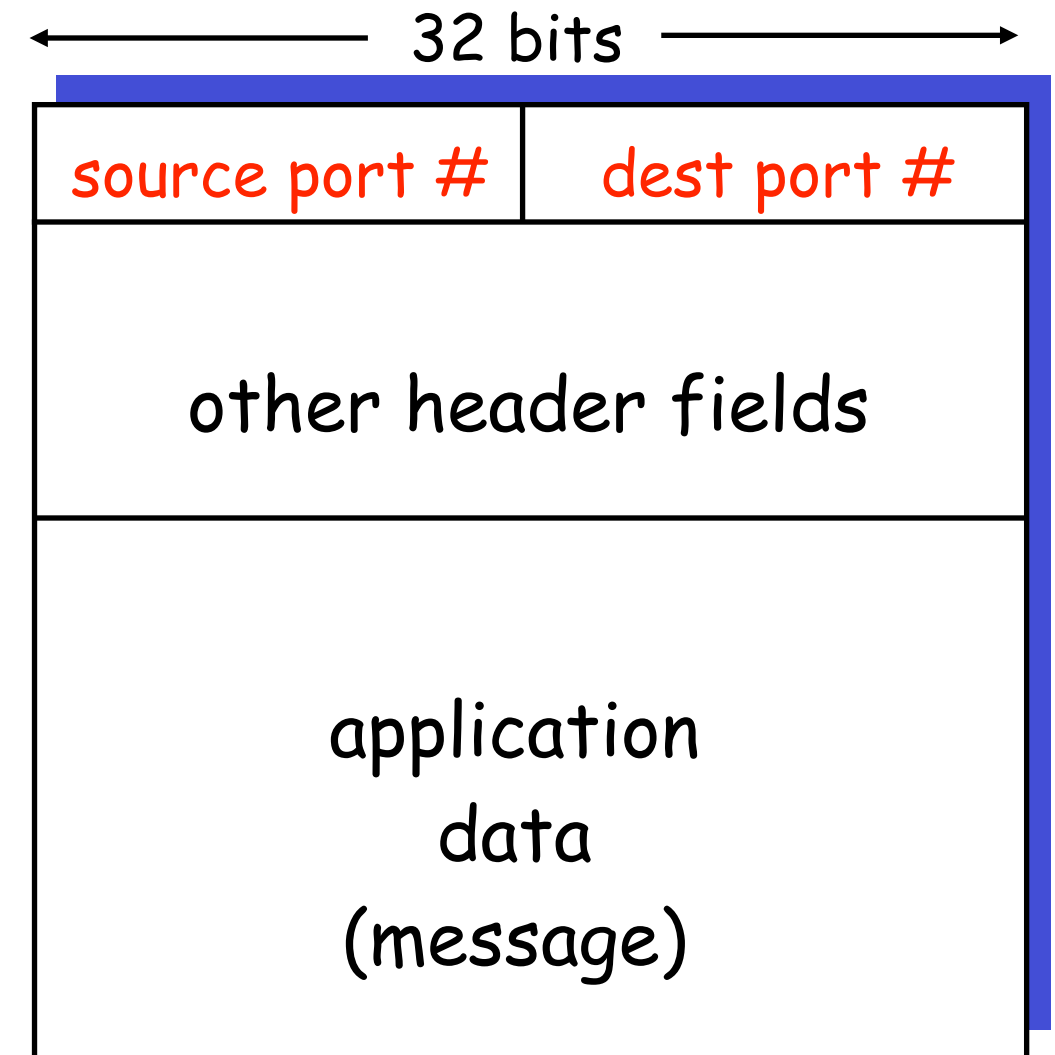
gathering data from multiple
sockets, enveloping data with
header (later used for
demultiplexing)

 = socket  = process



How demultiplexing works

- **host receives IP datagrams**
 - each datagram has source IP address, destination IP address
 - each datagram carries 1 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

- Create sockets with port numbers:

```
addr1.sin_port = htons(12534);
```

```
addr2.sin_port = htons(12535);
```

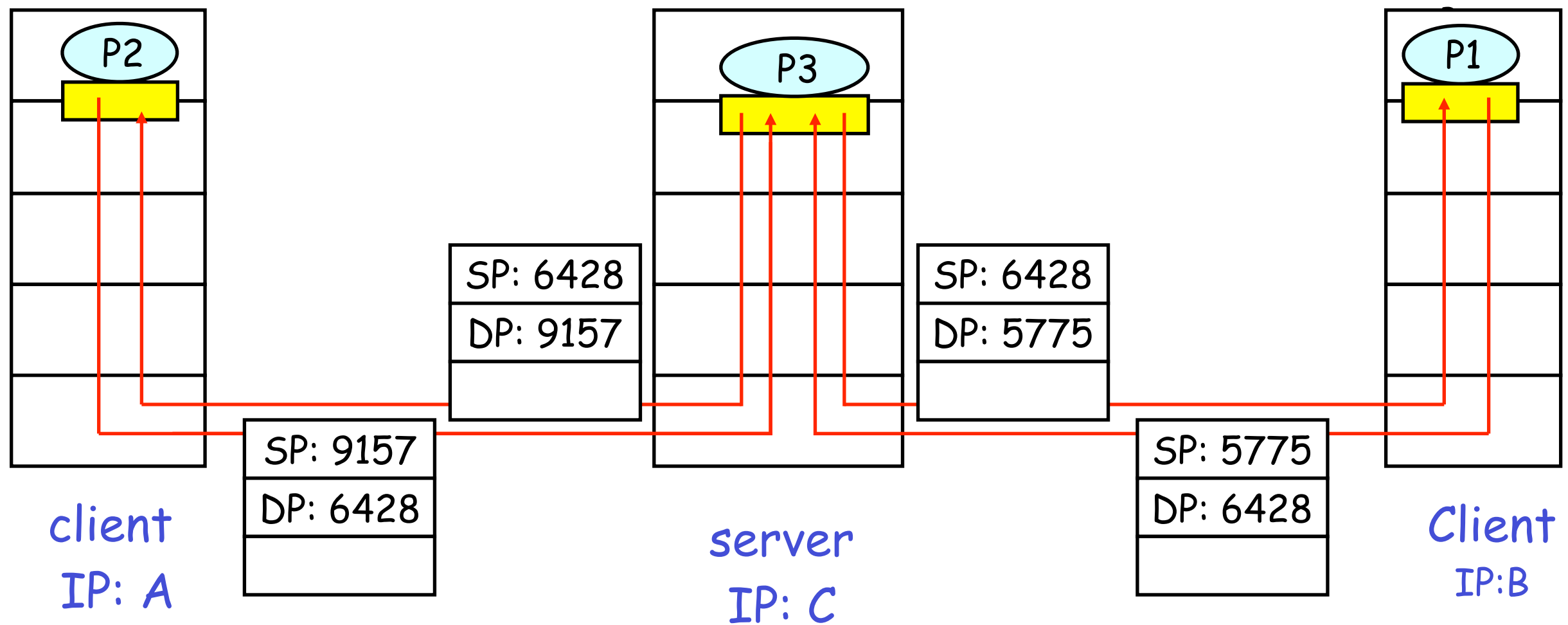
- UDP socket identified by two-tuple:

(dest IP address, dest port number)

- When host receives UDP segment:
 - ▶ checks destination port number in segment
 - ▶ directs UDP segment to socket with that port number
- IP datagrams with different source IP addresses and/or source port numbers directed to same socket



Connectionless demux (cont)

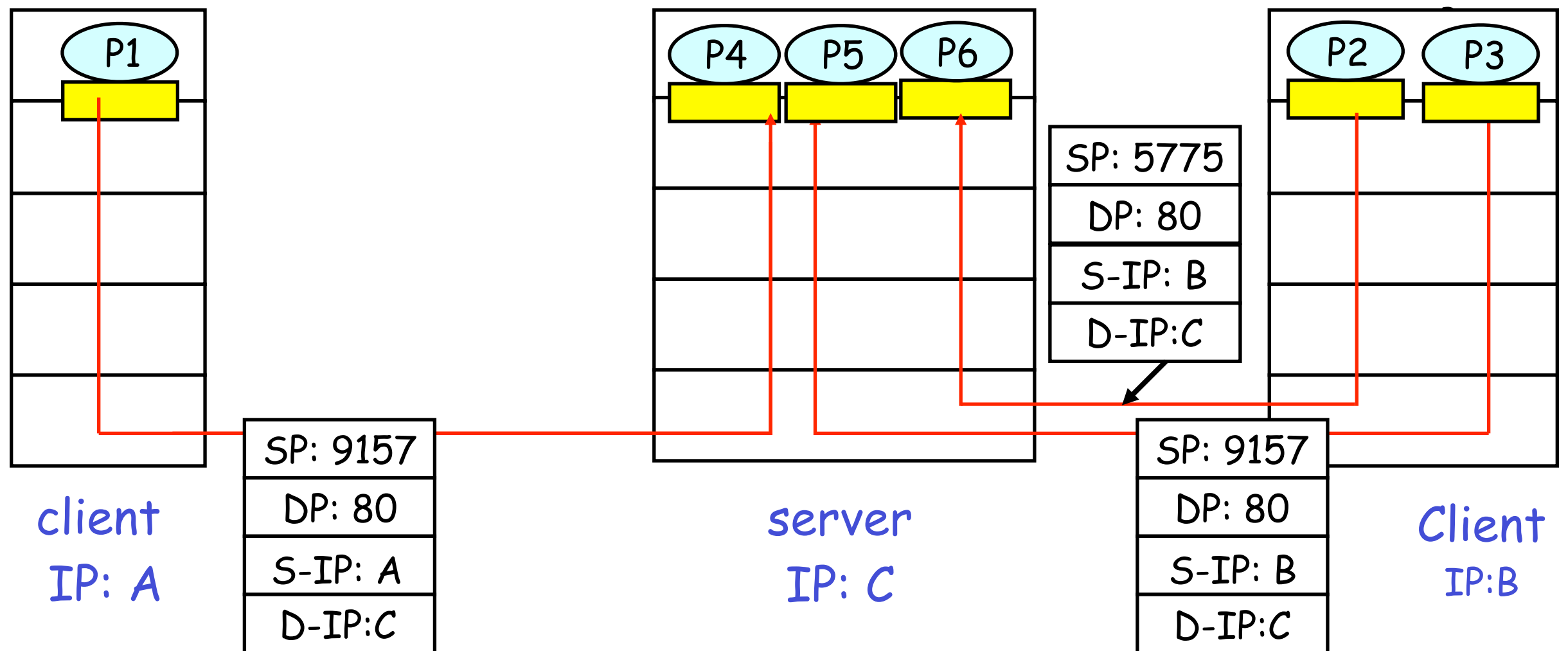


SP provides "return address"

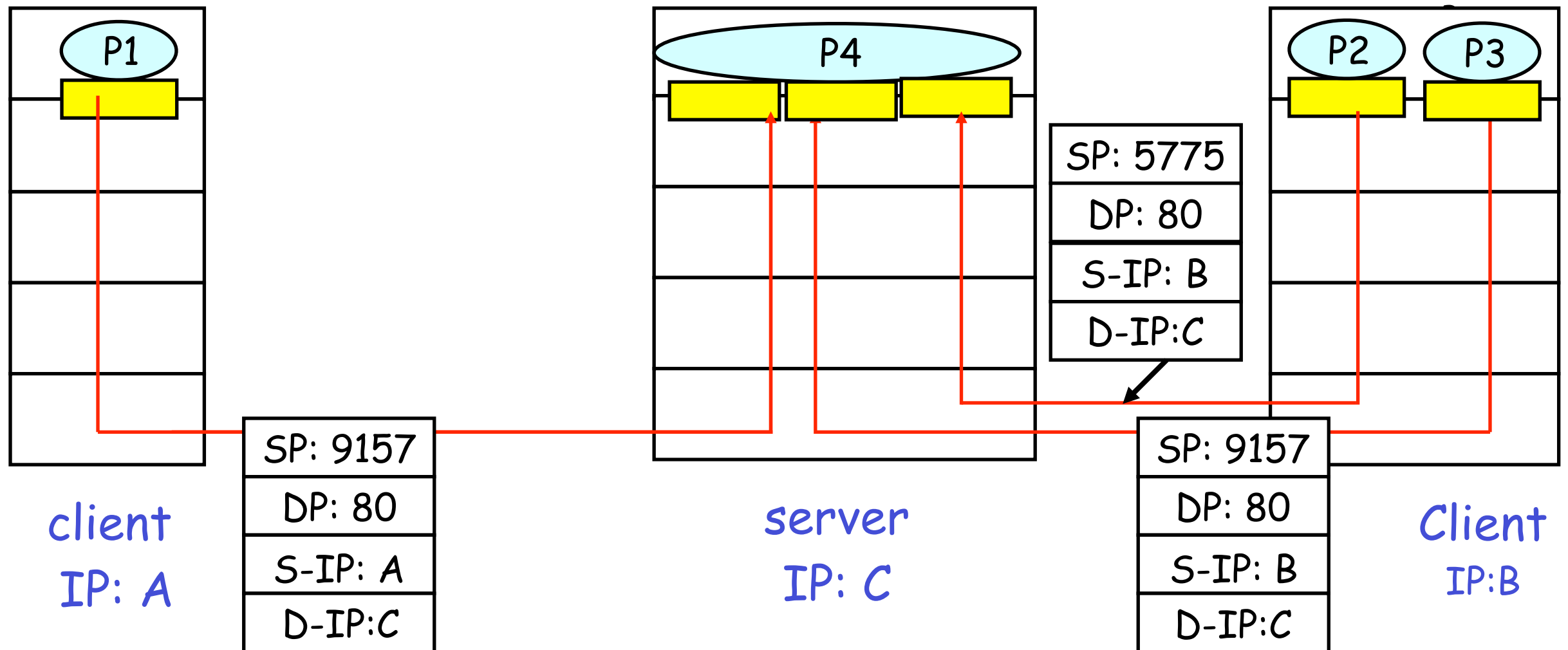
Connection-oriented demux

- TCP socket identified by 4-tuple:
 - source IP address
 - source port number
 - dest IP address
 - dest port number
- recv host 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 (cont)



Connection-oriented demux: Threaded Web Server



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

UDP: User Datagram Protocol [RFC 768]

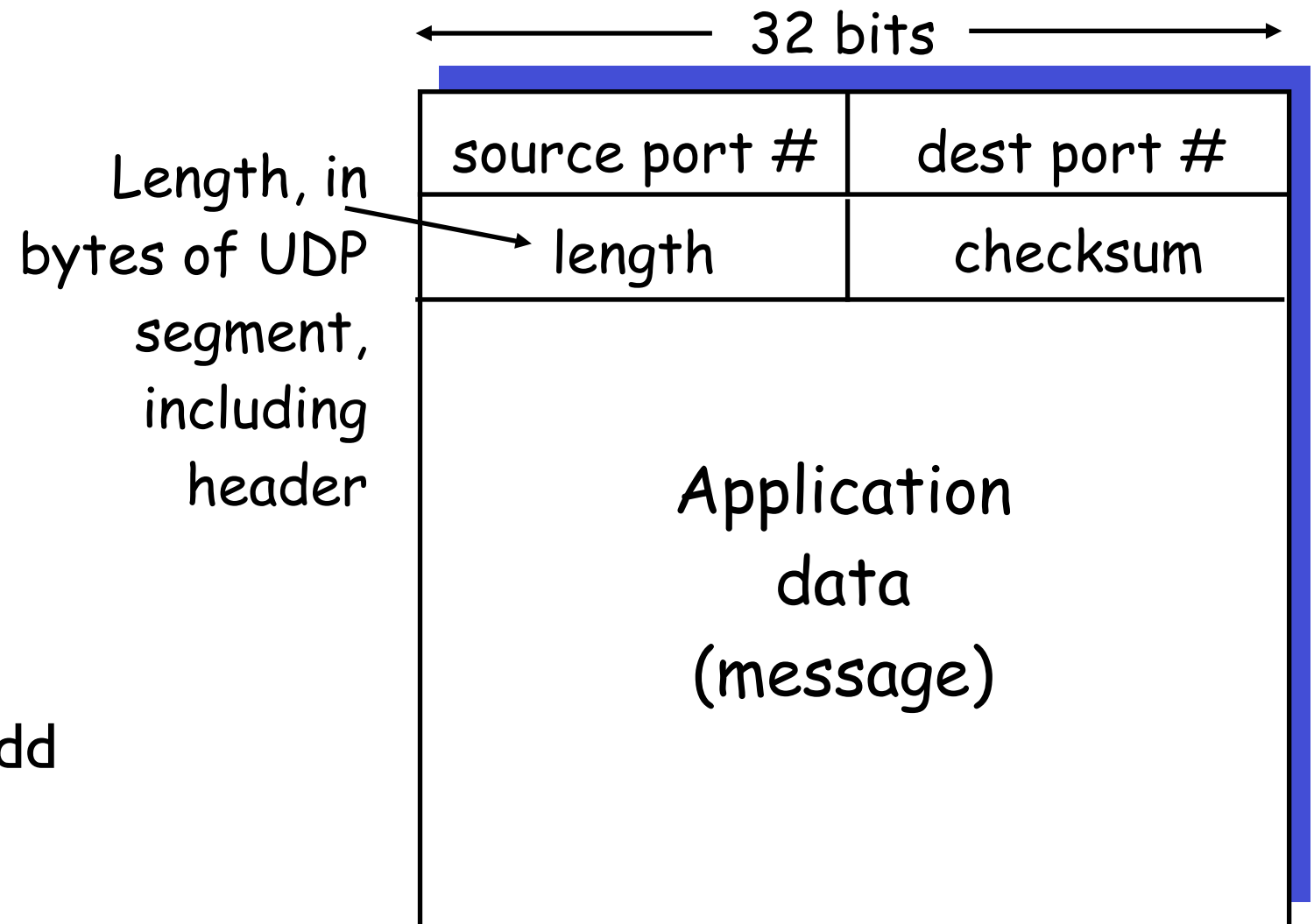
- “no frills,” “bare bones” Internet transport protocol
- “best effort” service, UDP segments may be:
 - lost
 - delivered out of order to app
- **connectionless:**
 - no handshaking between UDP sender, receiver
 - each UDP segment handled independently of others

Why is there a UDP?

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

UDP: more

- often used for streaming multimedia apps
 - loss tolerant
 - rate sensitive
- other UDP uses
 - DNS
 - SNMP
- reliable transfer over UDP: add reliability at application layer
 - application-specific error recovery!



UDP segment format

UDP checksum

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

Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
 - How is this different than 2's complement?
- 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. But maybe errors nonetheless? More later

Internet Checksum Example

- Note
 - When adding numbers, a carryout from the most significant bit needs to be added to the result
- 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
<hr/>																	
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
<hr/>																	
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

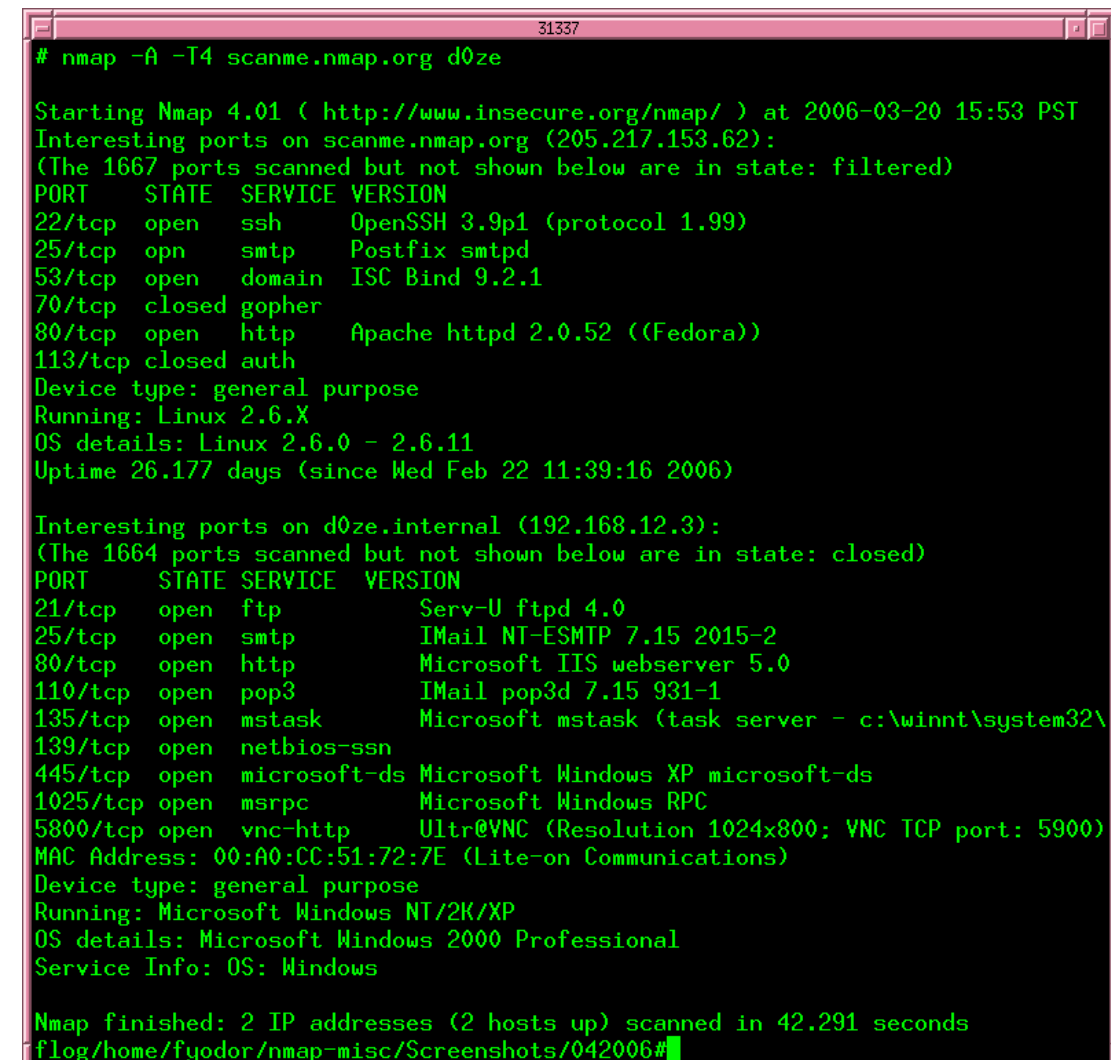
Port Scanning

- Technique used by black- and white-hat communities alike.
- Attempts to connect to a large number (usually all) of ports on a machine.
 - Successful responses mean that a process is running.
 - If you know what processes are running, you will be able to select the right exploit to launch.
 - Most firewalls offer some protection against this.
- This is happening *all the time* on the Internet.
 - The bad guys are constantly looking for a way in...



Port Scanning Tools

- *nmap* is the most popular tool for port scanning.
 - ...and it is free...
- By seeing which ports are active, nmap can tell a lot about your machine.
 - For instance, what OS you are running...
- Be careful to check with admins before running this!
 - Most admins will automatically shut you down if you run it...



```
# nmap -A -T4 scanme.nmap.org d0ze

Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at 2006-03-20 15:53 PST
Interesting ports on scanme.nmap.org (205.217.153.62):
(The 1667 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 3.9p1 (protocol 1.99)
25/tcp    open  smtp      Postfix smtpd
53/tcp    open  domain    ISC Bind 9.2.1
70/tcp    closed gopher
80/tcp    open  http      Apache httpd 2.0.52 ((Fedora))
113/tcp   closed auth
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.0 - 2.6.11
Uptime 26.177 days (since Wed Feb 22 11:39:16 2006)

Interesting ports on d0ze.internal (192.168.12.3):
(The 1664 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp       Serv-U ftpd 4.0
25/tcp    open  smtp      IMail NT-ESMTP 7.15 2015-2
80/tcp    open  http      Microsoft IIS webserver 5.0
110/tcp   open  pop3      IMail pop3d 7.15 931-1
135/tcp   open  mstask    Microsoft mstask (task server - c:\winnt\system32\
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp  open  msrpc     Microsoft Windows RPC
5800/tcp  open  vnc-http  Ultr@VNC (Resolution 1024x800; VNC TCP port: 5900)
MAC Address: 00:A0:CC:51:72:7E (Lite-on Communications)
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows 2000 Professional
Service Info: OS: Windows

Nmap finished: 2 IP addresses (2 hosts up) scanned in 42.291 seconds
flog/home/fyodor/nmap-misc/Screenshots/042006#
```