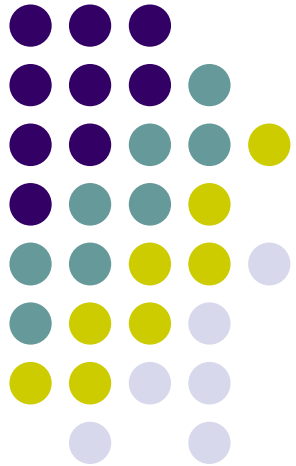
Chapter 3

Transport layer

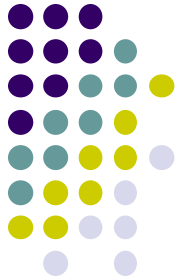
Liping Shen 申丽萍

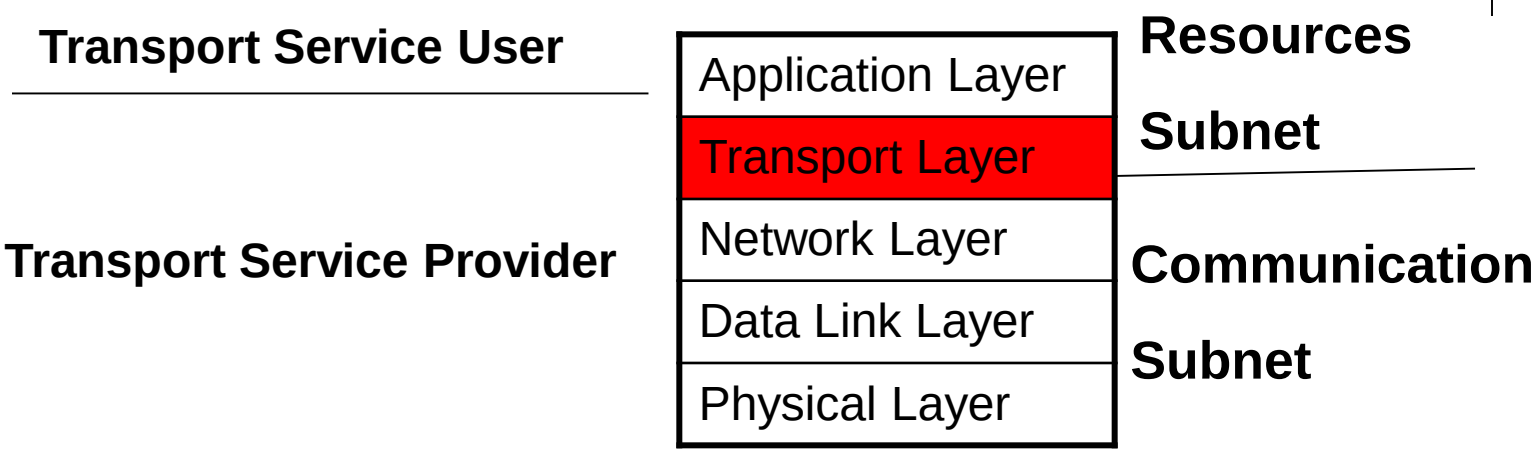
lpshen@sjtu.edu.cn



These slides are based upon the exceptional slides provided by Kurose and Ross

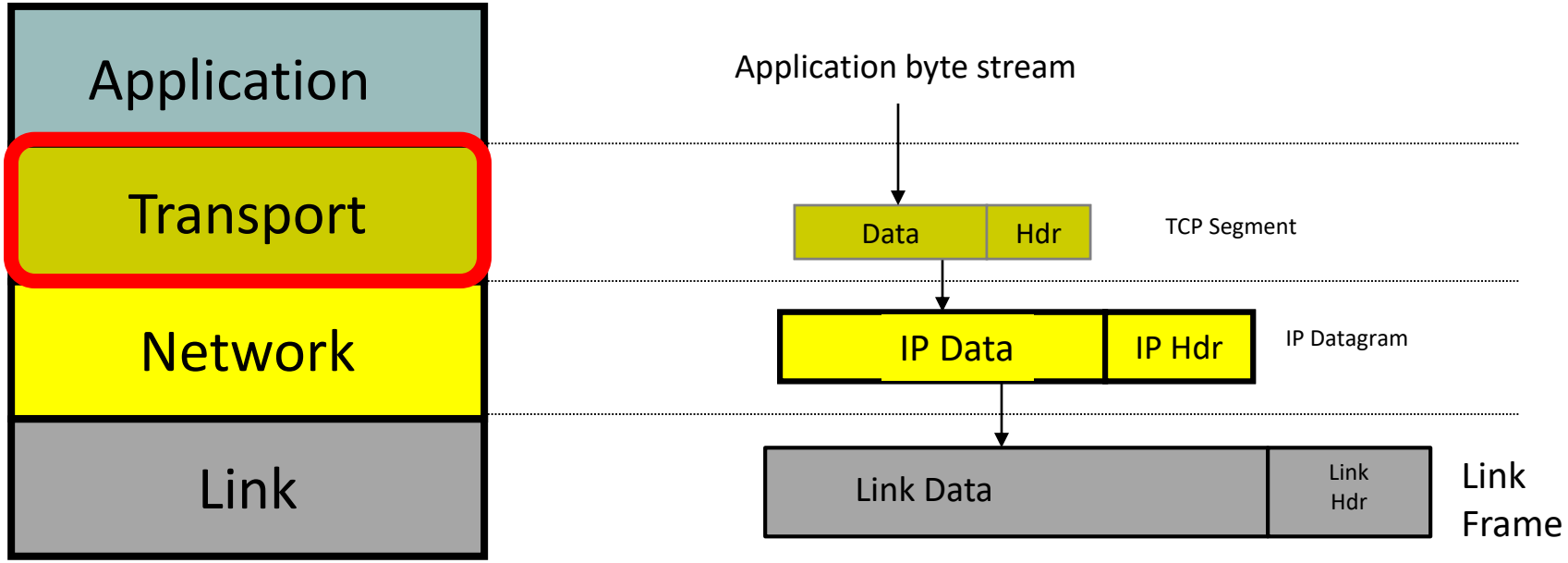
The Transport Layer in the Hybrid Model



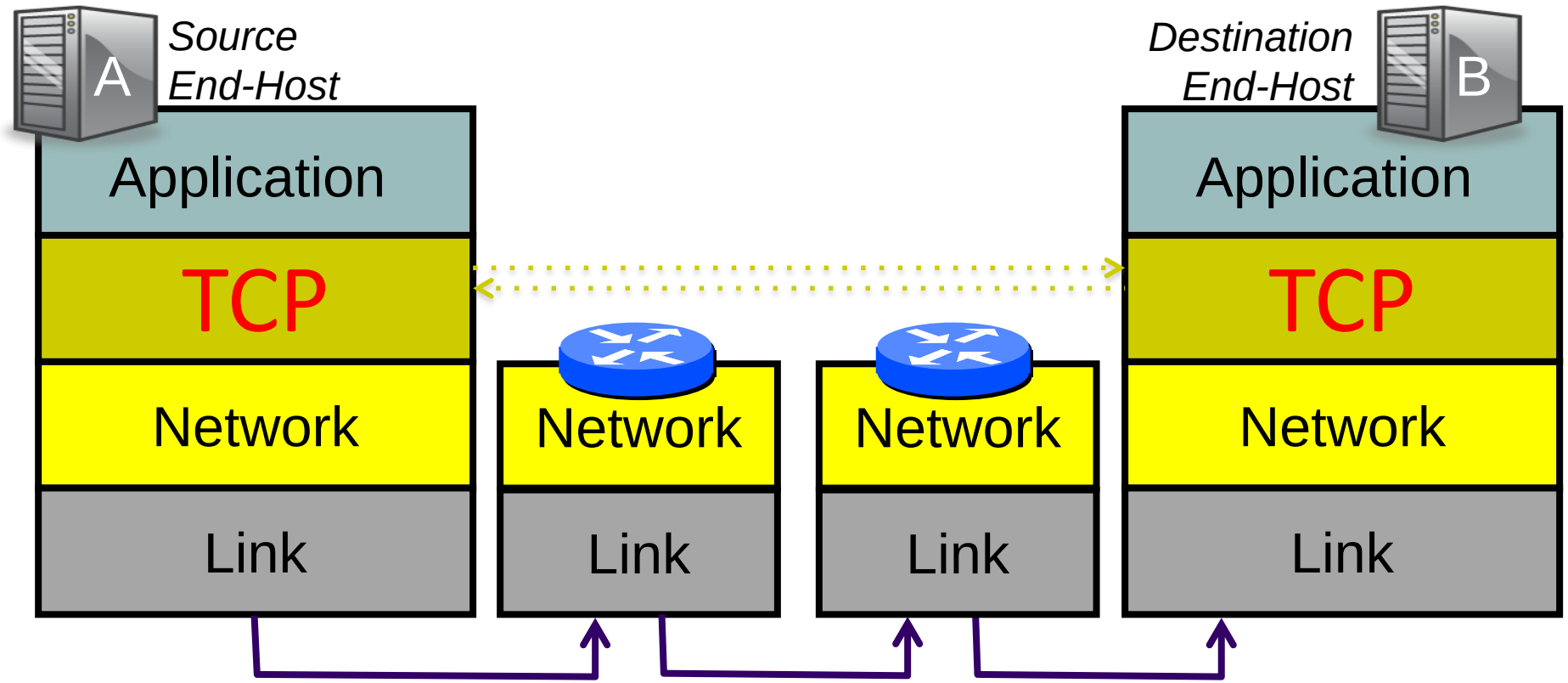


- service: builds on the network layer to provide data-delivery service for applications with the desired reliability or quality
- isolation: isolating the upper layers from the technology, design, and diversities of the subnet.
- complexity: depending on the services offered by the network layer

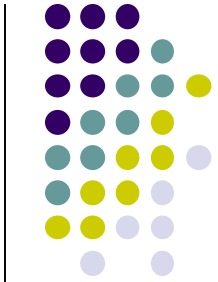
Transmission Control Protocol (TCP)



Peer TCP layers communicate



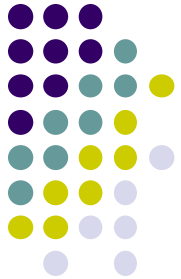
Chapter 3: Goals



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 and congestion control

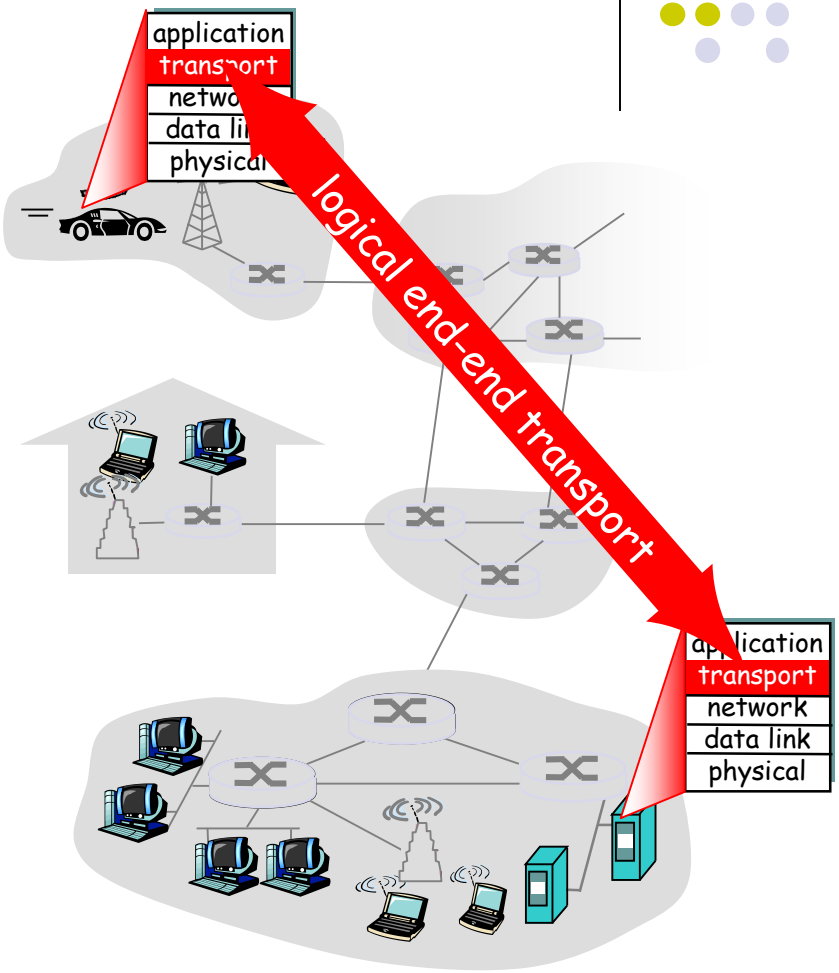
Chapter 3 Roadmap



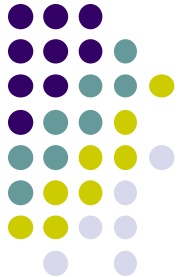
- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- 3.6 Principles of congestion control
- 3.7 TCP congestion control

Transport services and protocols

- Transport layer provides end-to-end connectivity across the network
- transport protocols run in end systems
- more than one transport protocol available to apps
  - TCP: reliable, in-order delivery
  - UDP: no-frills extension of “best-effort” IP



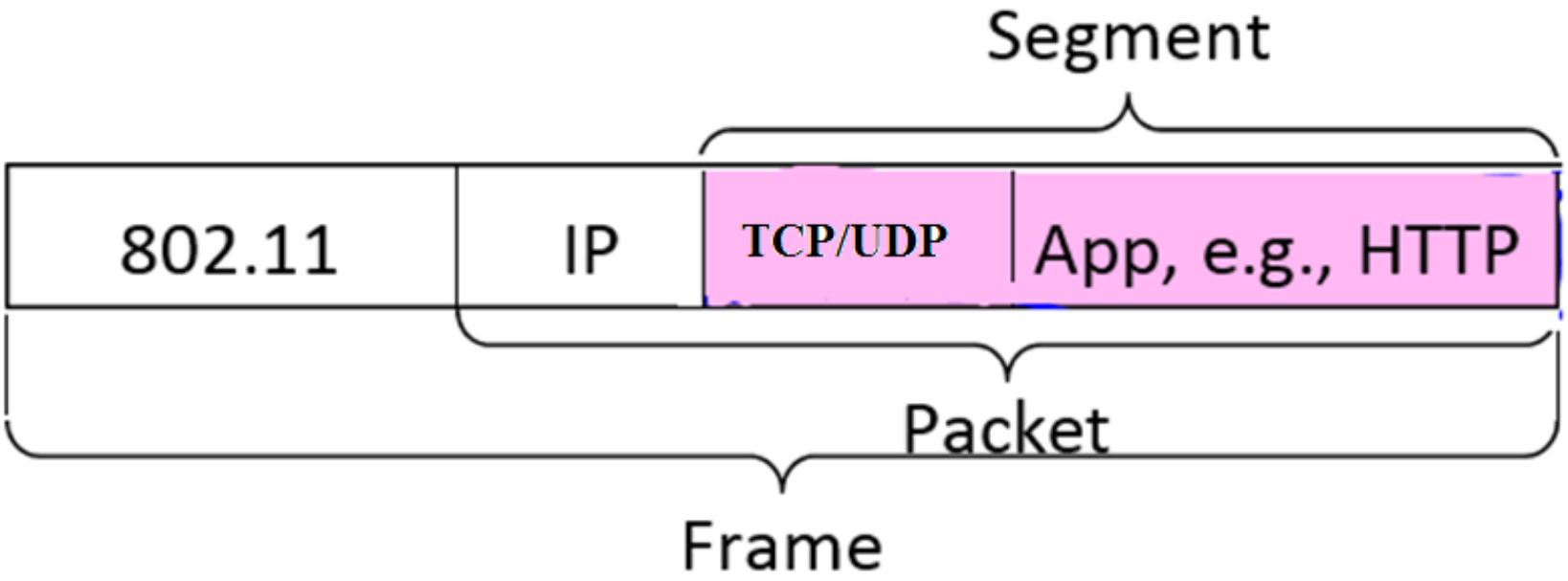
TCP and UDP



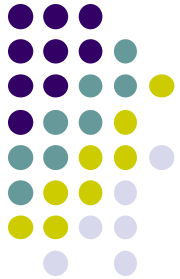
TCP (Streams)	UDP (Datagrams)
Connections	Datagrams
Bytes are delivered once, reliably, and in order	Messages may be lost, reordered, duplicated
Arbitrary length content	Limited message size
Flow control matches sender to receiver	Can send regardless of receiver state
Congestion control matches sender to network	Can send regardless of network state

Transport Segment

- Segments carry application data across the network.
- Segments are carried within packets within frames

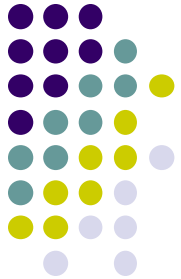


Chapter 3 outline



- 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

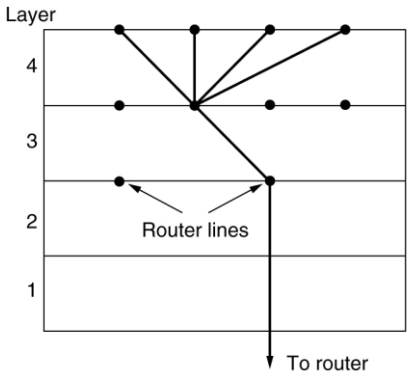
Ports

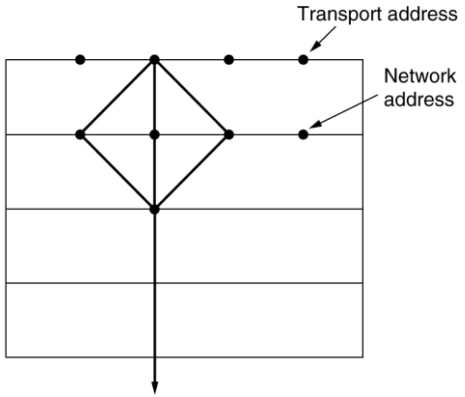


- Application process is identified by the two-tuple (IP address + port)
- Ports are 16-bit integers representing local addressing of a process.
- Servers often bind to “well-known ports”
  - <1024, used permanently and statically
  - e.g. FTP (20/21), SMTP (25), POP3 (110), IMAP (143), http (80), https(443).
- Clients often assigned “ephemeral” ports
  - Chosen by OS, used temporarily and dynamically

Multiplexing

- **Upward multiplexing**: if only one network address is available on a host, all transport connections on that machine have to use it.
- **Downward multiplexing**: If a user requires a lot of bandwidth that cannot be supported by a single network virtual circuit, use several circuits for a single connection.





(a) Upward multiplexing. (b) Downward multiplexing.

Multiplexing/demultiplexing

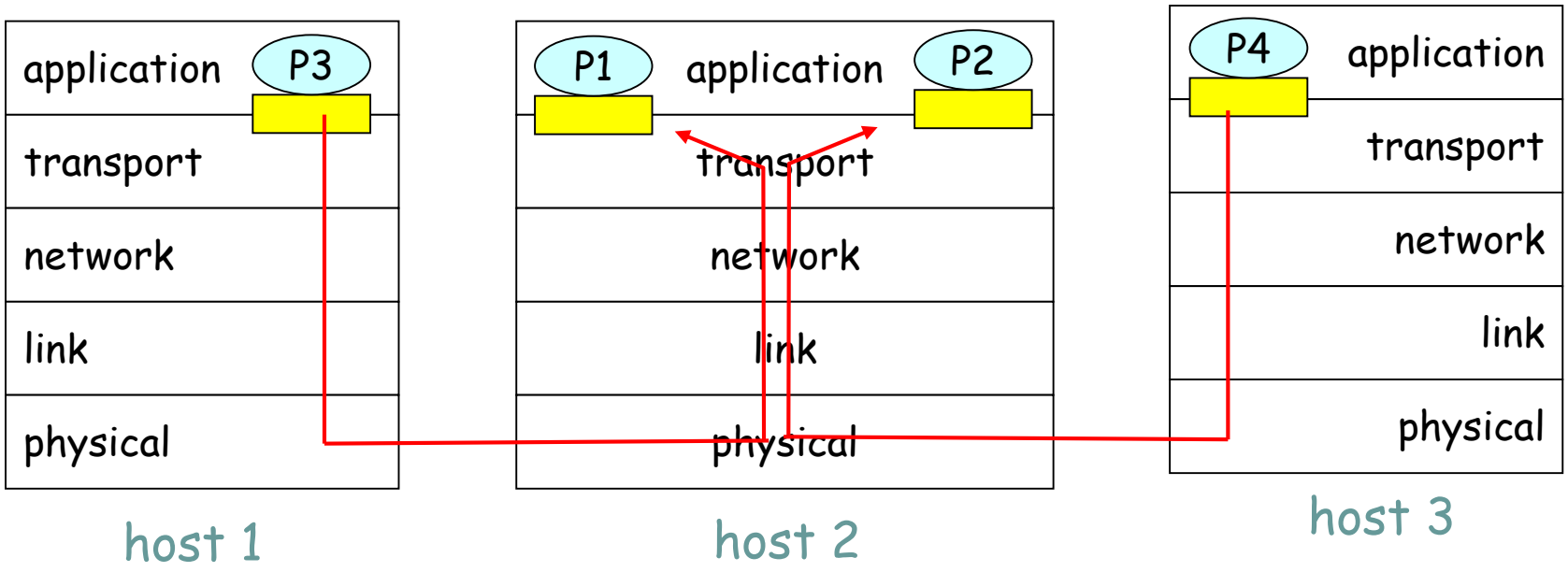
Demultiplexing at rcv host:

delivering received segments to correct socket

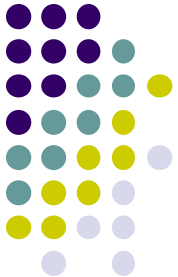
Multiplexing at send host:

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

= socket = process

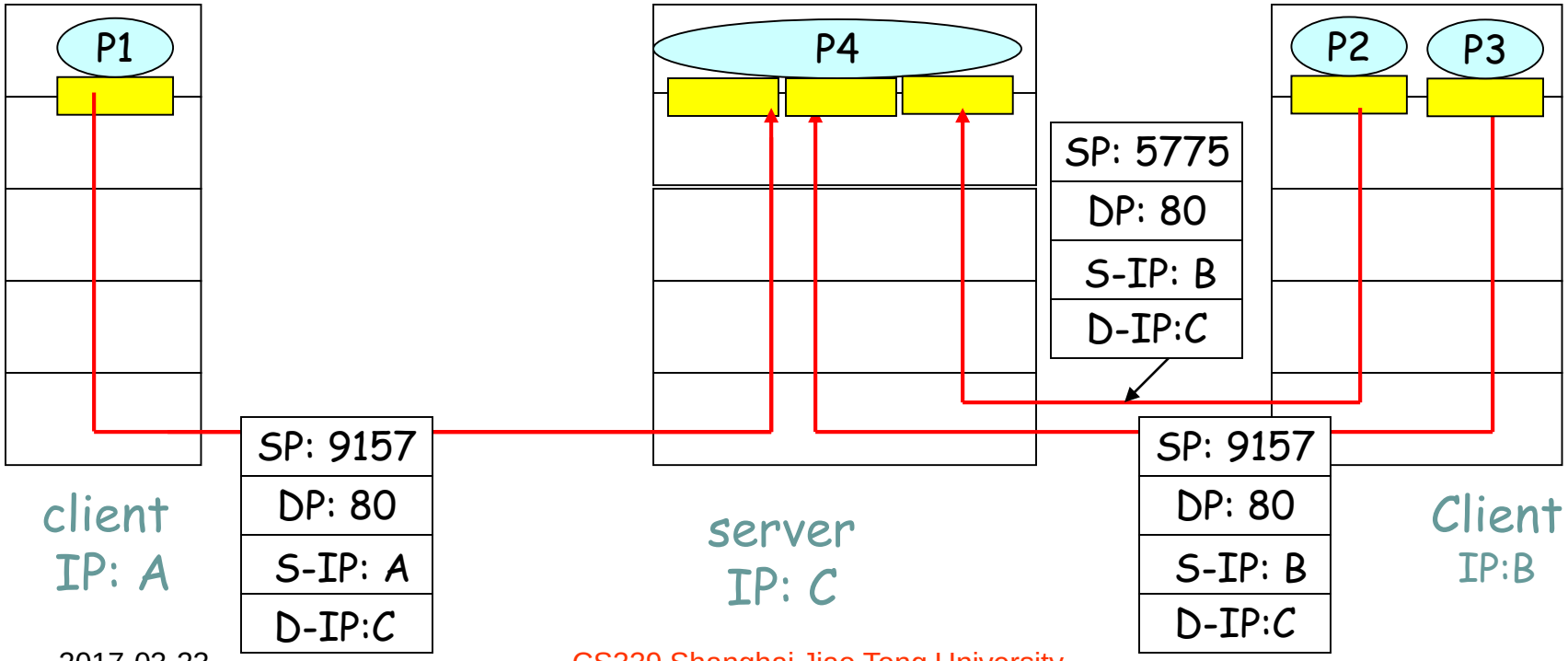


Connection-oriented demux: Threaded Web Server



- TCP socket identified by 4-tuple:

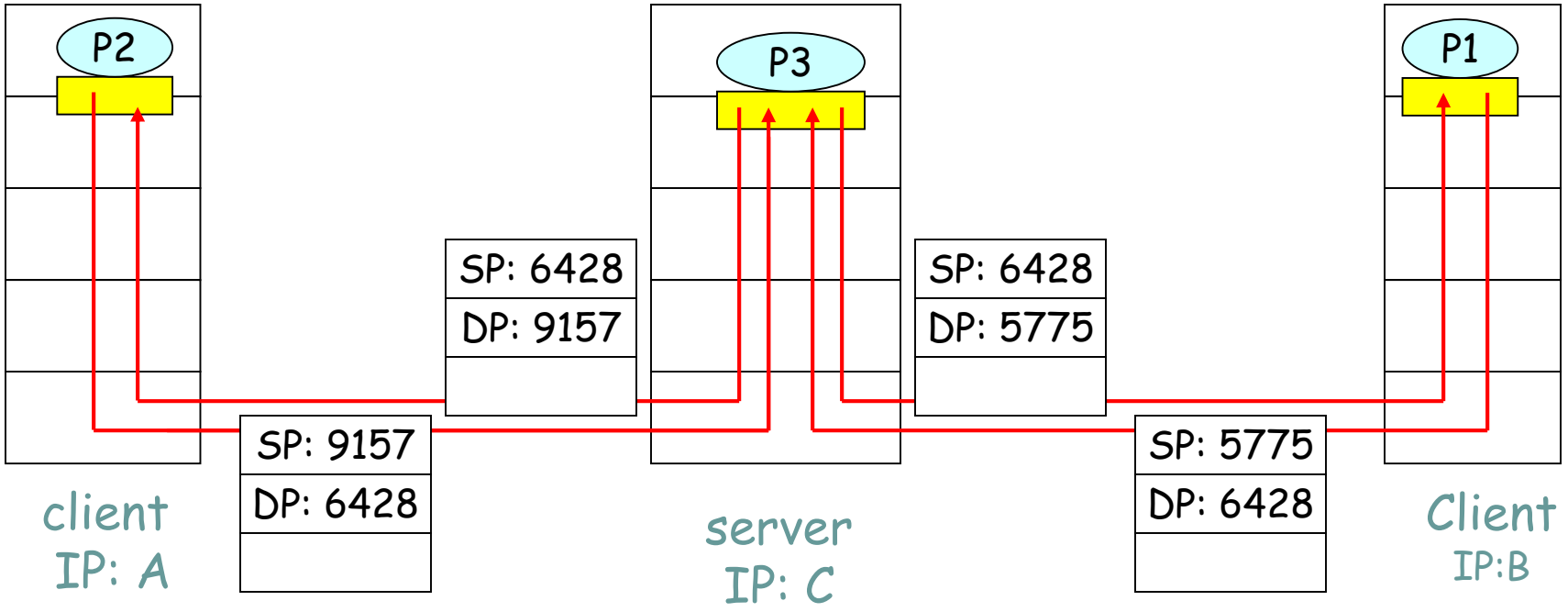
(source IP address, source port number, dest IP address, dest port number)



Connectionless demux

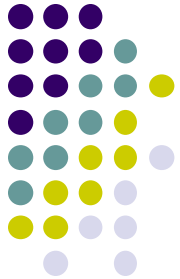
- UDP socket identified by two-tuple:

(dest IP address, dest port number)



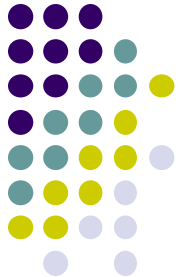
SP provides "return address"

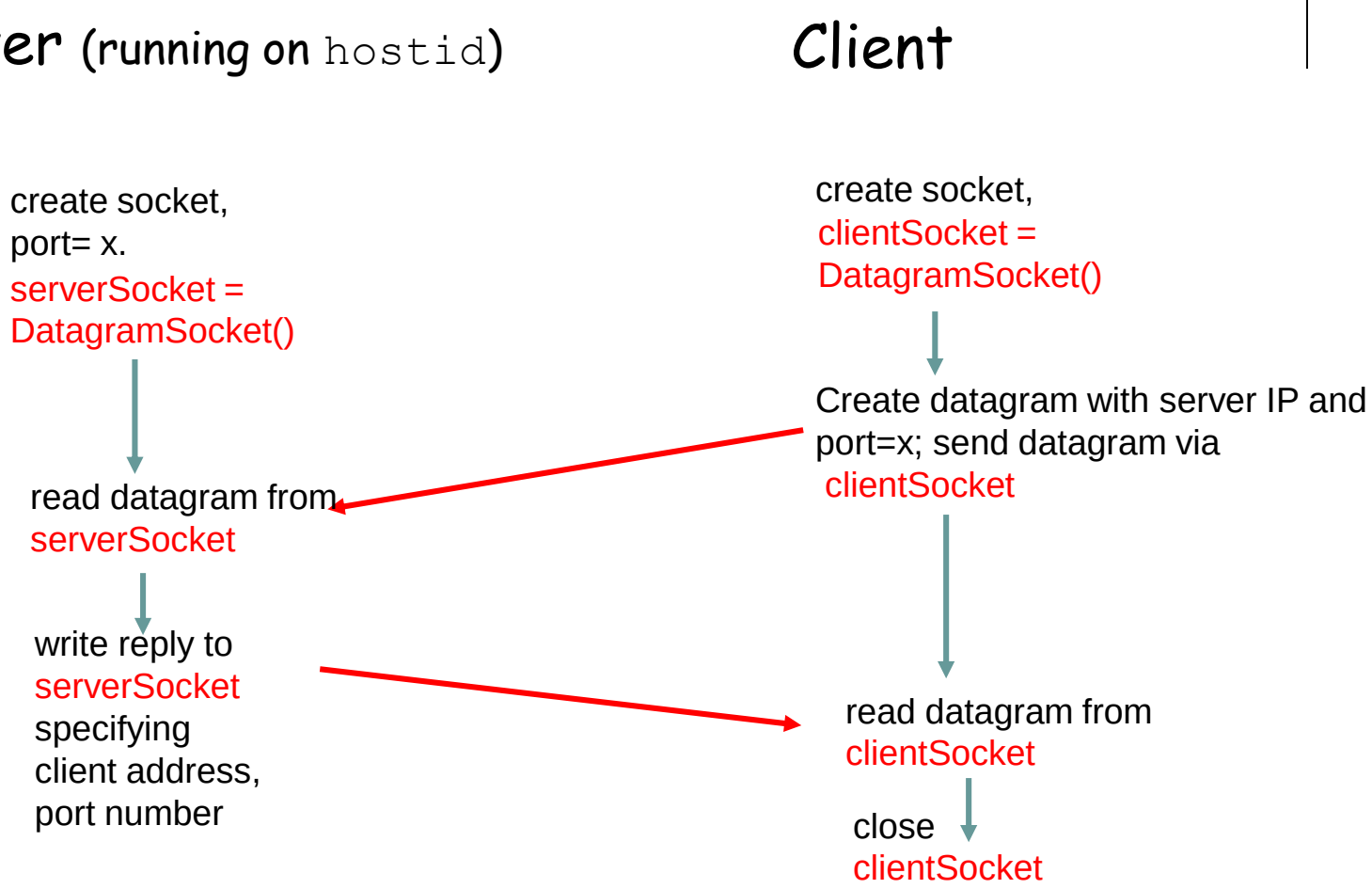
Chapter 3 outline



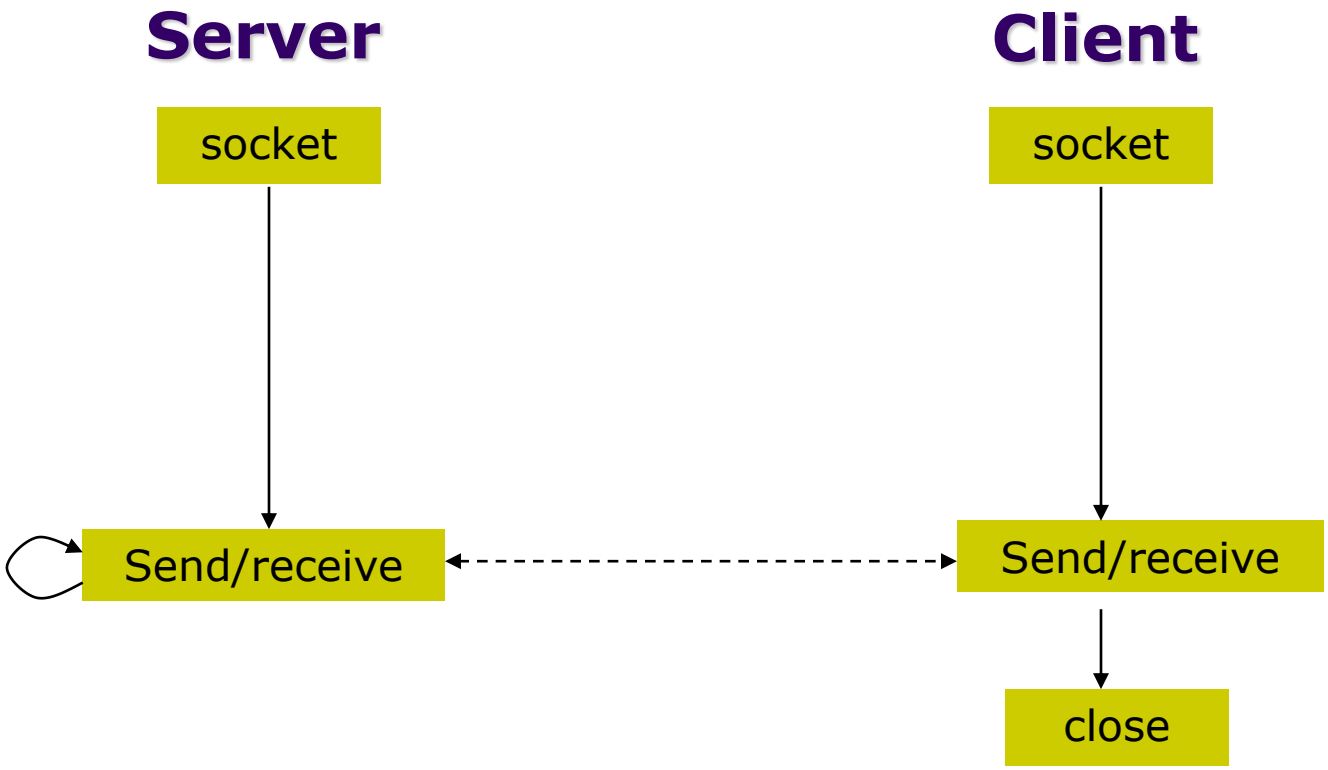
- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- 3.6 Principles of congestion control
- 3.7 TCP congestion control

Recap： UDP socket interaction

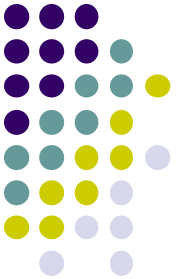




Recap： UDP socket interaction

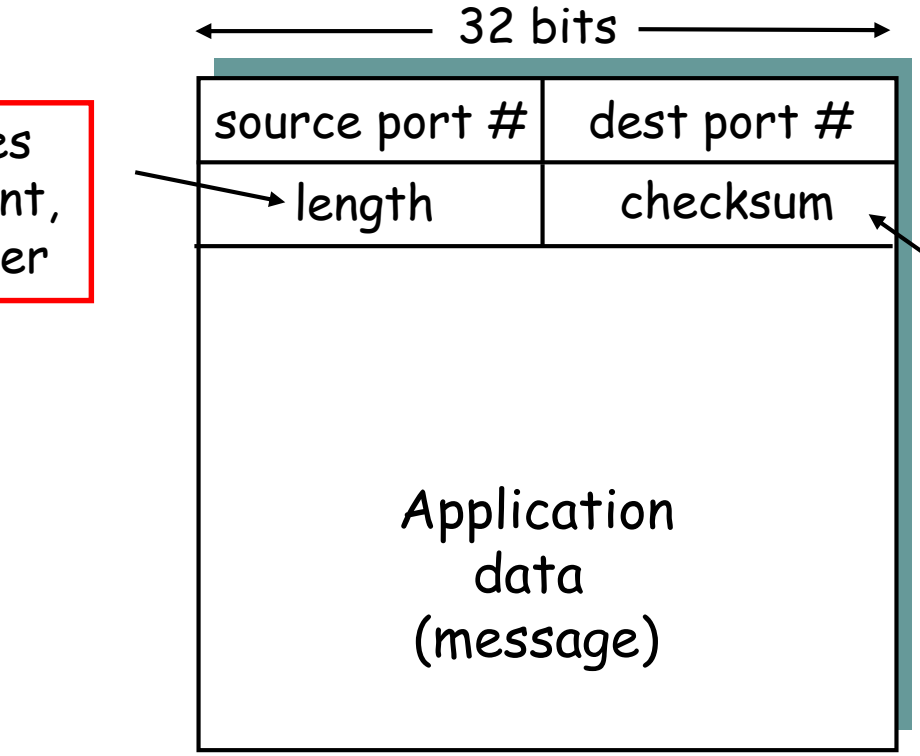


UDP: User Datagram Protocol [RFC 768]



- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently
- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - Lost
  - error
  - out of order
- reliable transfer over UDP: add reliability at application layer

UDP segment format

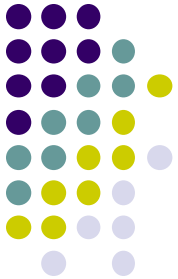


Length in bytes of UDP segment, including header

detect error, discard segment when error detected

Why "UDP has better control over what data is sent and when" ?

UDP checksum



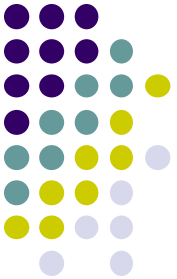
- Goal: detect “errors” in transmitted segment
- calculation: addition (1’s complement sum) of segment contents
- Example: add two 16-bit integers

	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

UDP Service Model

Property	Behavior
Connectionless Datagram Service	No connection established. Packets may show up in any order.
Self contained datagrams	Everything needed for transmission is included in datagram (e.g. the destination address)
Unreliable delivery	1. No acknowledgments. 2. No mechanism to detect missing or mis-sequenced datagrams. 3. No flow control.

Why UDP



- no connection establishment (which can add delay)
- no retransmission (which can add delay)
- simple: no connection state at sender, receiver
- small segment header: 8 bytes
- no flow control and congestion control: UDP can blast away as fast as desired
- Examples:
  - short message interaction apps: DNS, SNMP, PRC
  - loss tolerant and delay sensitive streaming multimedia apps: ipPhone, SKYPE