

CMPE 344 Computer Networks

Spring 2012

End-to-End Protocols

Reading: Peterson and Davie, §5.1, 5.2.1-5.2.3

The transport layer

- Recall that IP provides **host-to-host** packet delivery service
- In general, one requires **process-to-process** communication channels
- Transport layer deals with process-to-process communication channels through end-to-end protocols (between “end” applications)

Transport protocols are expected to provide...

- Guaranteed message delivery
- In-order message delivery
- Delivery of at most one copy of the message
- Support for arbitrarily large messages
- Support for synchronization between sender and receiver
- Flow control
- Support for multiple application processes on each host

Best-effort networks

- Note that network layer below the transport layer may
 - Drop messages
 - Reorder messages
 - Deliver duplicate copies of a given message
 - Limit messages to some finite size
 - Delay messages for a long time
- Such networks (e.g., IP) are said to provide a best-effort service

UDP and TCP

- We will study the Internet's UDP and TCP protocols:
 - UDP: User Datagram Protocol
 - TCP: Transmission Control Protocol
- UDP provides a simple asynchronous demultiplexing service
- TCP provides a reliable byte-stream service

User Datagram Protocol

- UDP as a simple demultiplexer
 - Extends the host-to-host delivery service of IP into a process-to-process communication service
- There are usually many processes running on a host so the protocol needs to add a level of demultiplexing to allow multiple application processes on each host to share the network

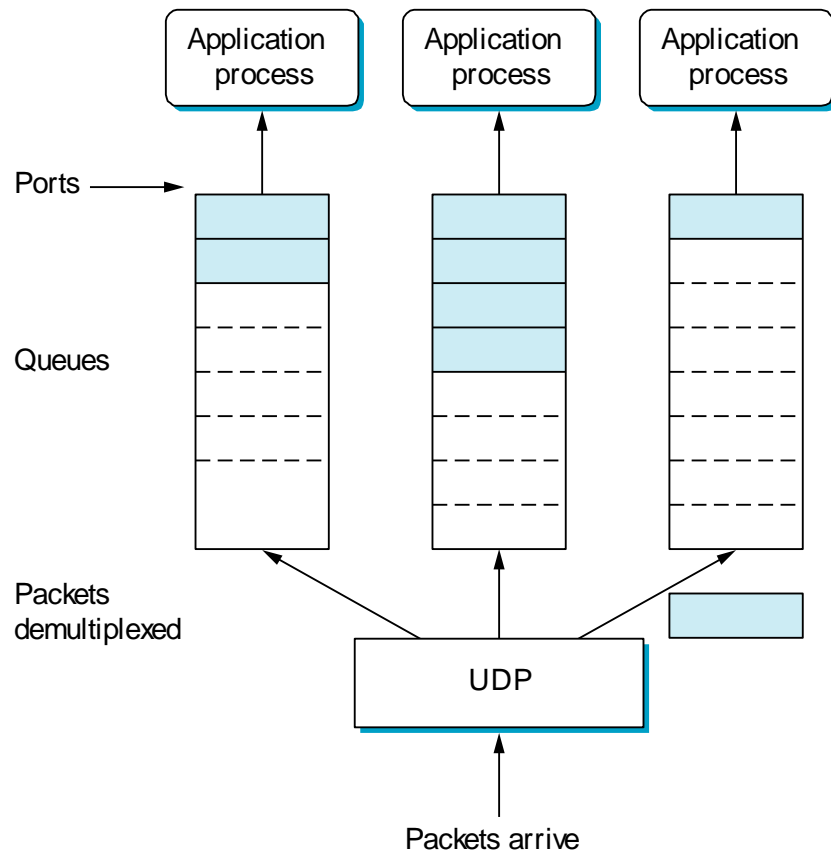
Ports

- Processes running on a host are identified by **ports** or mailboxes
- Source process sends a message to a port and the destination process receives the message from that port
- The header of a UDP packet contains port numbers for both the sender (source) and the receiver (destination) process

More on ports

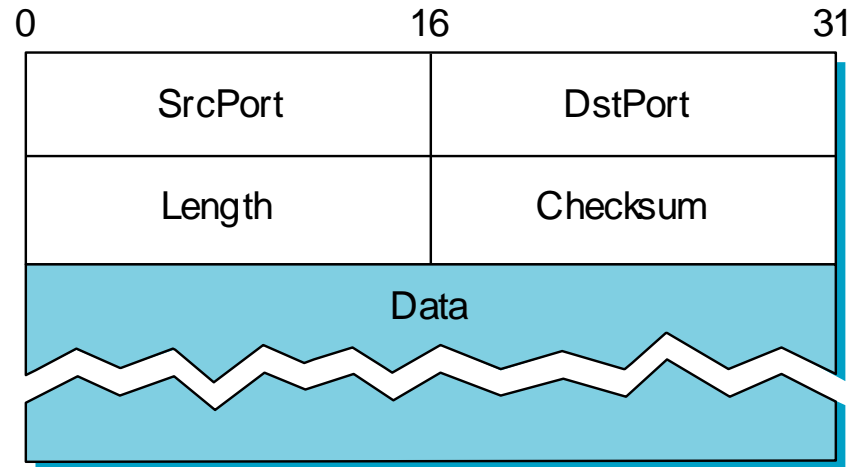
- Ports are interpreted only on a single host
- That is, a process is identified by a port on a particular host using
 <port, host> pair
- This pair is the demultiplexing key for the UDP

Implementation by message queues



Note that UDP does not implement flow control to tell the sender to slow down. If a queue becomes full because the messages are arriving too fast, they will be discarded.

UDP header format



SrcPort and DstPort: Source and destination port numbers

Length: Length of UDP packet in bytes

Checksum: See next slide

UDP error checking

- UDP is not reliable, no guarantee for in-order delivery
- However, it can ensure the correctness of the message by the use of a checksum over the following:
 - The UDP header +
 - The message body +
 - The pseudo header: 3 fields from IP header (protocol no, src IP addr, dst IP addr) plus UDP length field
- Contrast UDP's checksumming with IP checksumming and ATM HEC!

UDP client and servers

- How does a client learn a server's port number?
 - Servers accept messages at **well-known ports** (port numbers < 1024)
 - e.g., Echo: Port 7, Time: Port 37,
DNS: Port 53
- The server already knows the client's port number from client's contact to the server

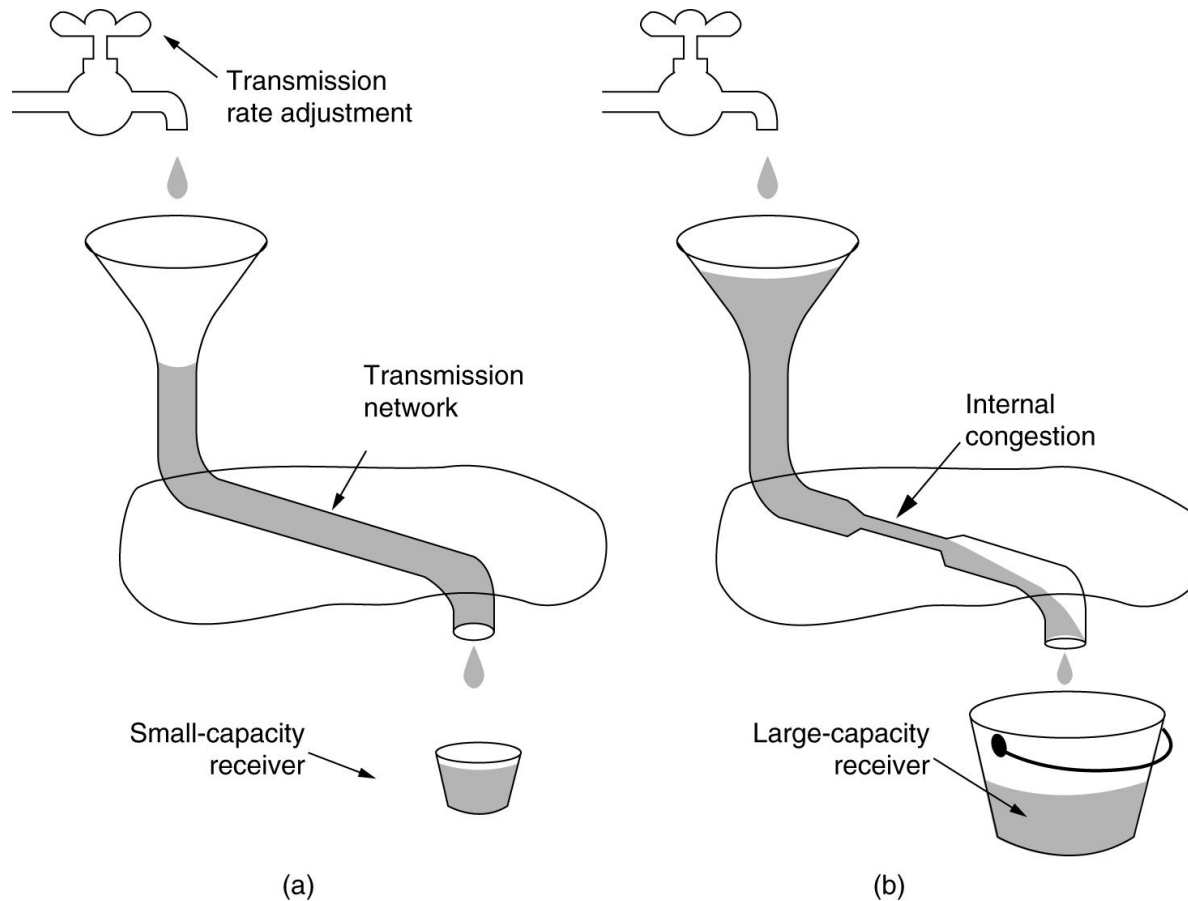
Transmission Control Protocol

- TCP provides a reliable, connection-oriented byte-stream service
- TCP is a full-duplex protocol
 - Each TCP connection supports a pair of byte streams, one for each direction
- TCP provides **flow control**
 - Receiver can limit how much data the sender can transmit at a given time
 - This mechanism prevents sender from overrunning receiver

More on TCP

- Like UDP, TCP supports a demultiplexing mechanism
 - Multiple application processes on a host simultaneously communicate with their peers
- TCP provides **congestion control**
 - Throttle how fast TCP sends data to prevent sender from overloading network elements (switches and routers)
 - Note that TCP **flow control is an end-to-end** issue whereas congestion control is concerned with interaction of hosts and network elements

Flow vs. congestion control



(a) A fast network feeding a low capacity receiver

(b) A slow network feeding a high-capacity receiver

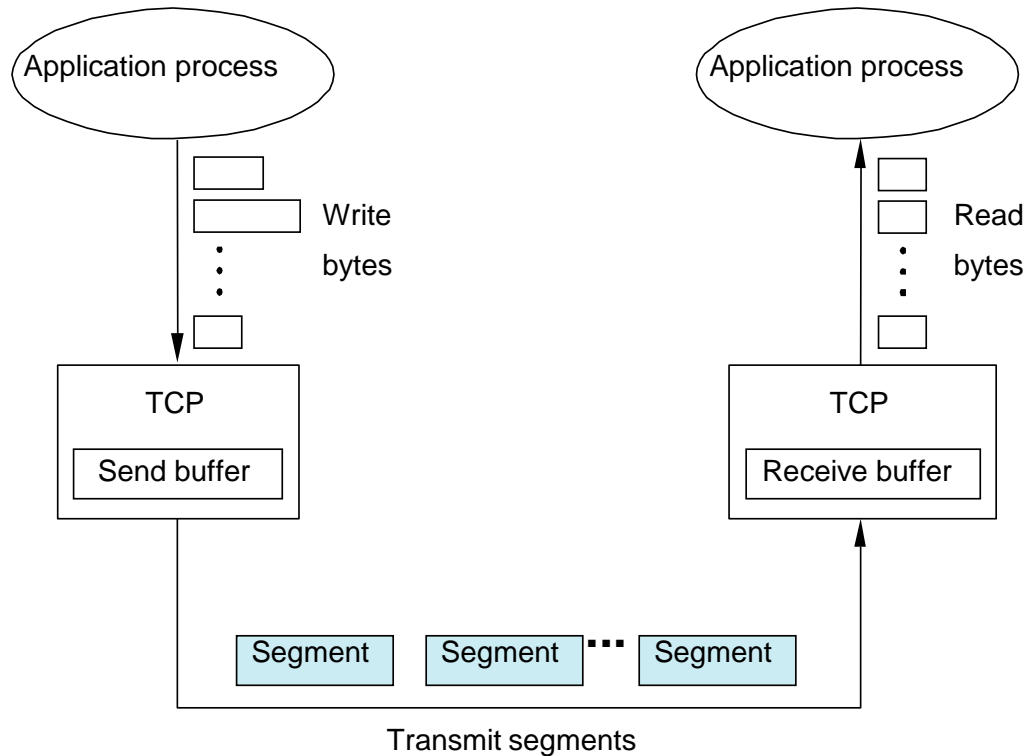
End-to-end sliding window

- TCP uses the sliding window algorithm to provide reliable in-order delivery of messages
- However, in TCP, sliding window runs over logical connection between processes as opposed to sliding window running over a single physical link between two nodes (at layer 2)
- Note that in TCP, RTT will be variable!
 - TCP has mechanisms to estimate RTT

TCP's sliding window and flow control

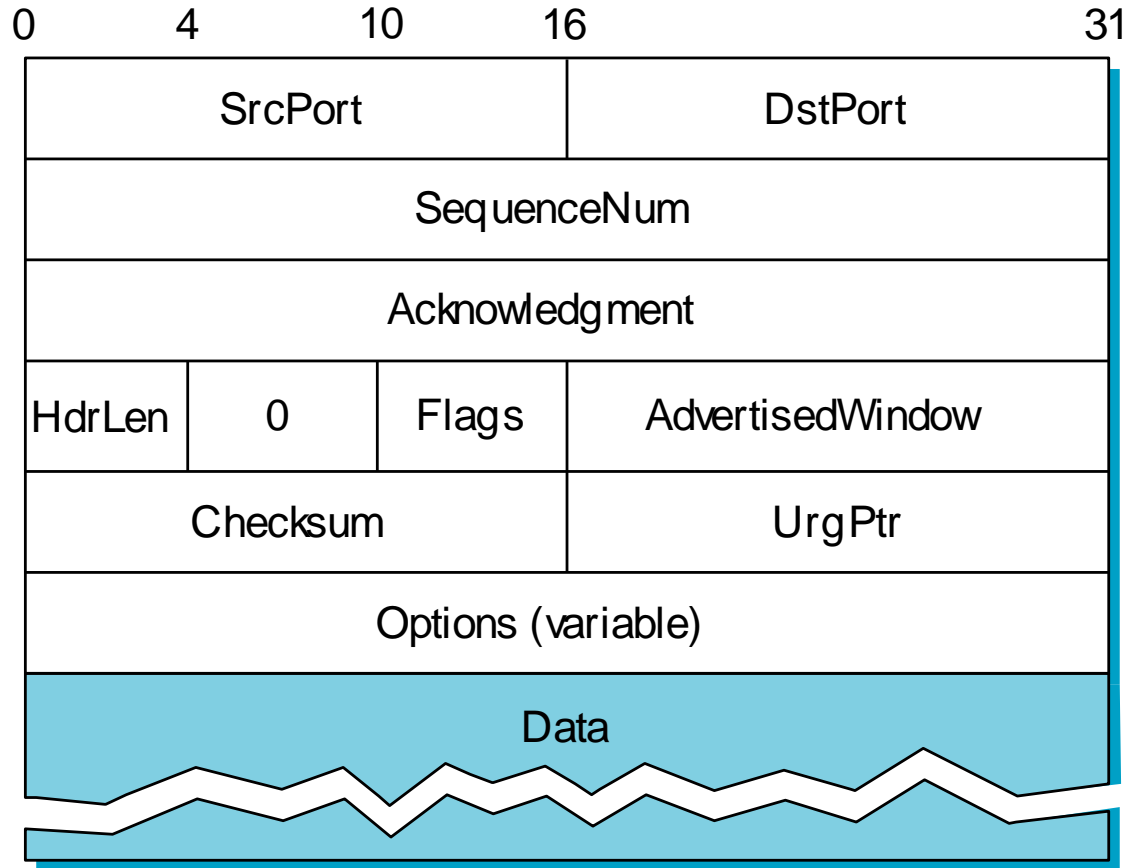
- Flow control: TCP has a mechanism to “learn” how much resources (e.g., buffer space) the other side can allocate to the connection
- TCP's end-to-end approach can be contrasted to the hop-by-hop approach taken by some protocols such as ITU's packet switching protocol X.25

Byte streams and segments



TCP “packets” are called segments

TCP header format



TCP header fields

- SrcPort and DstPort: Source and destination port numbers
- SequenceNum: Position in sender's byte stream of data in segment
- AcknowledgementNum: Number of next byte expected
- HdrLen: Number of 32-bit words in TCP header
- Flags: URG, ACK, PSH, RST, SYN, FIN
- AdvertisedWindow: How much data TCP is willing to accept
- Checksum: Checksum over, header, data, and pseudo header
- UrgPtr: Pointer to urgent data in segment

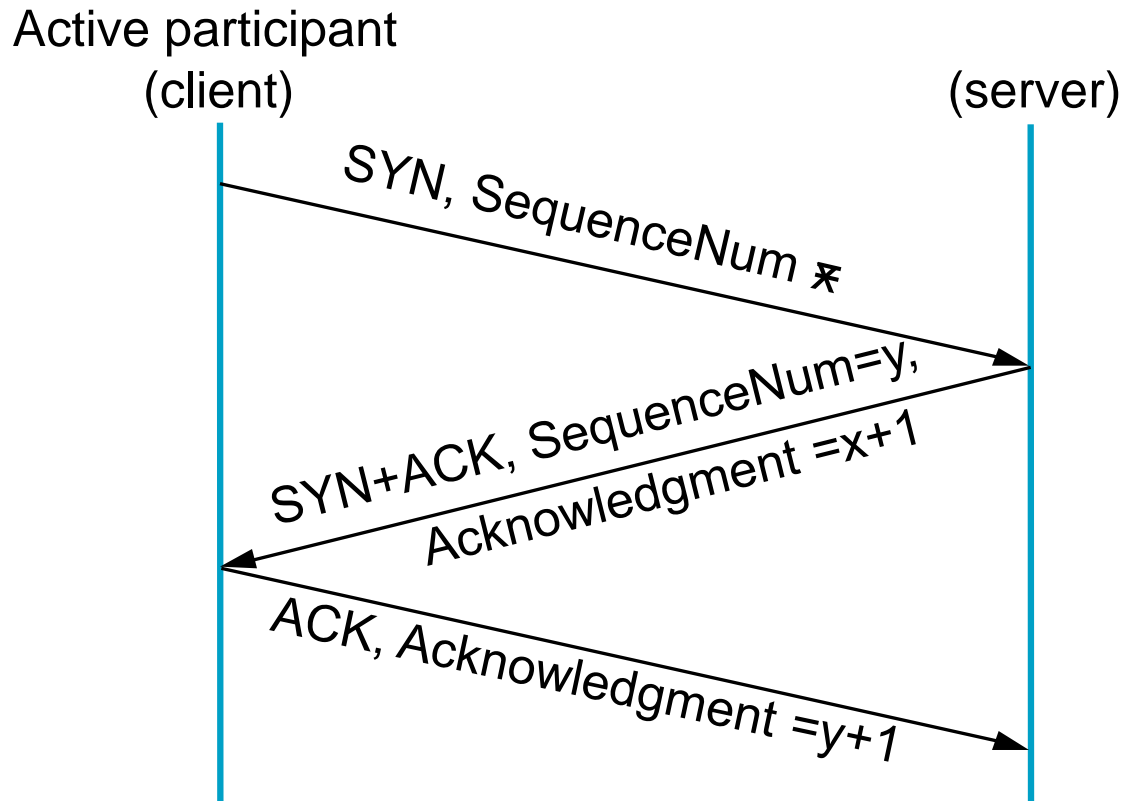
TCP demultiplexing

- The 4-tuple
<src port, src IP addr, dst port, dst IP addr>
uniquely identify each TCP connection
- This 4-tuple is used as the demultiplexing key
- Because TCP identifies a connection by a 4-tuple, a given TCP port number can be shared by multiple connections on the same host
 - Multiple connections can exist simultaneously on one local port

Passive and active open in TCP

- TCP is connection-oriented; both ends of the connection must agree that a connection is desired
- Application program on one end (server) performs a **passive open** indicating to the OS that it will accept an incoming connection
- Application program on the other end (client) must then contact its OS with an **active open** request to establish a connection

TCP three-way handshake for connection establishment



Assume server executed a passive open.
SYN bit is used to establish a connection.

Some well-known TCP ports

- Mail service: Port 25
- FTP: Port 21
- Telnet: Port 23
- Web service: Port 80
- etc.

- For example,
<18.26.3.36, 1069, 128.10.2.3, 21>
might correspond to an FTP connection

Network traffic composition

- TCP dominates the current Internet traffic
 - > ~80% of total Mbytes
 - TCP mediated Web applications make up
 - > ~50% of total Mbytes
- (Based on measurements on Sprint IP backbone network)