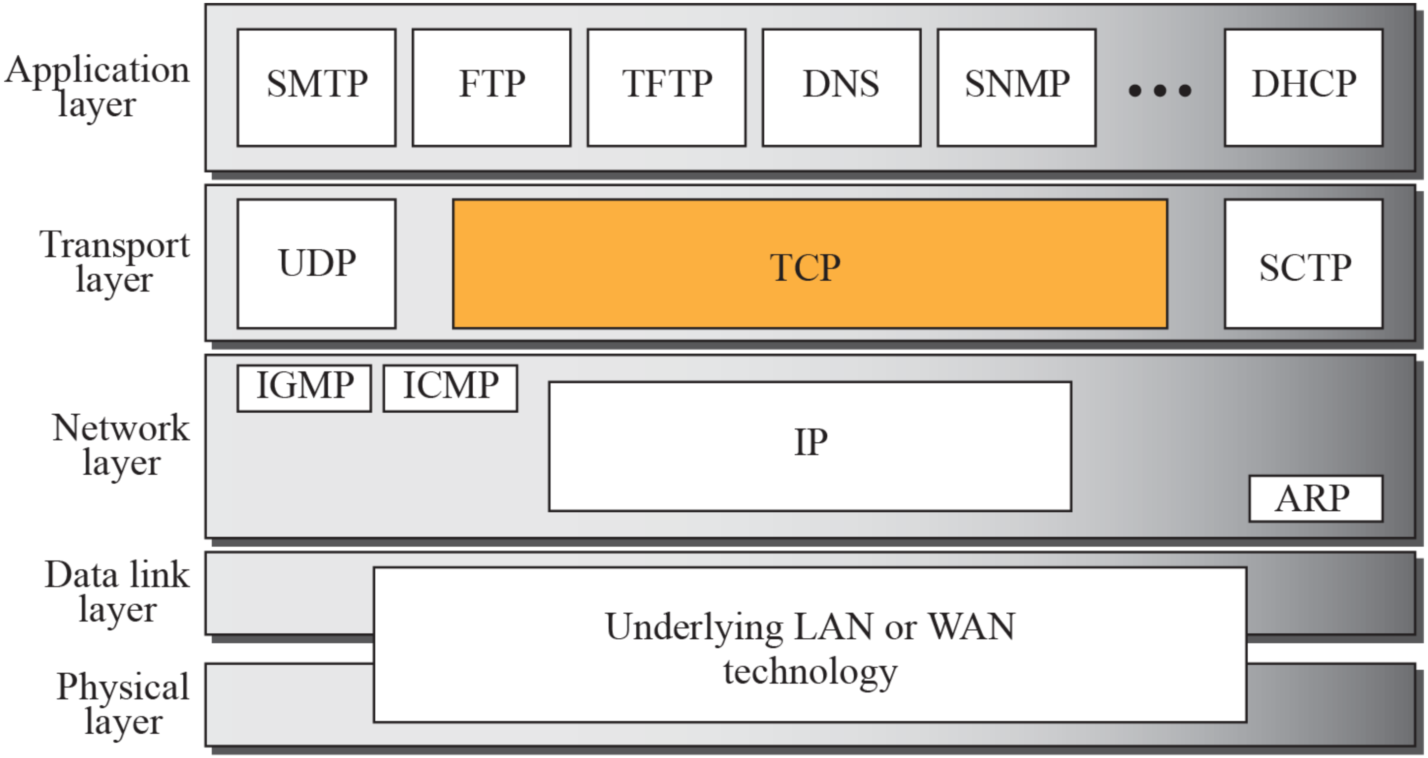
CPE348: Introduction to Computer Networks

Lecture #15: Chapter 5.1

Jianqing Liu
Assistant Professor of Electrical and Computer Engineering, University of Alabama in Huntsville

jianqing.liu@uah.edu
http://jianqingliu.net

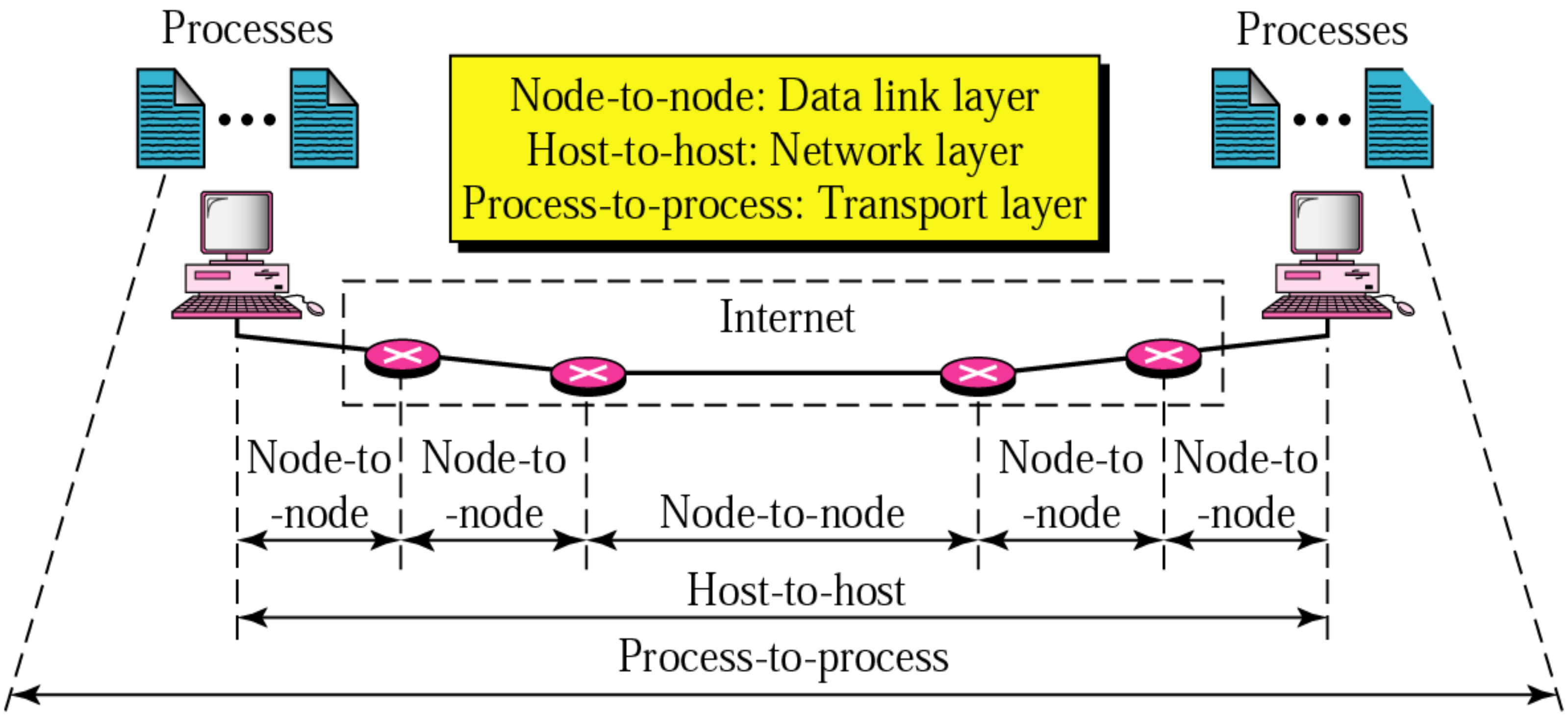
Chapter 5 – Overview



Chapter 5 – Outline

- End-to-end processes to consider:
  - Simple Demultiplexer (UDP)
  - Reliable Byte Stream (TCP)
  - Request/Reply Protocol (RPC)
  - Multimedia Specific Protocol (RTP)

Transport layer – Overview



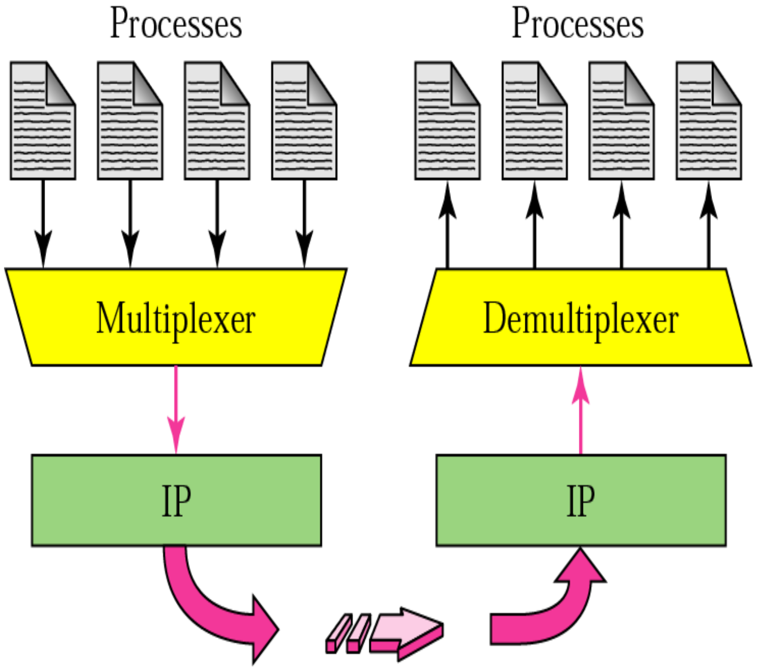
Transport layer – Overview

Multiplexing

Sender may have several processes (e.g., Youtube, WhatsApp) that need to send packets

Demultiplexing

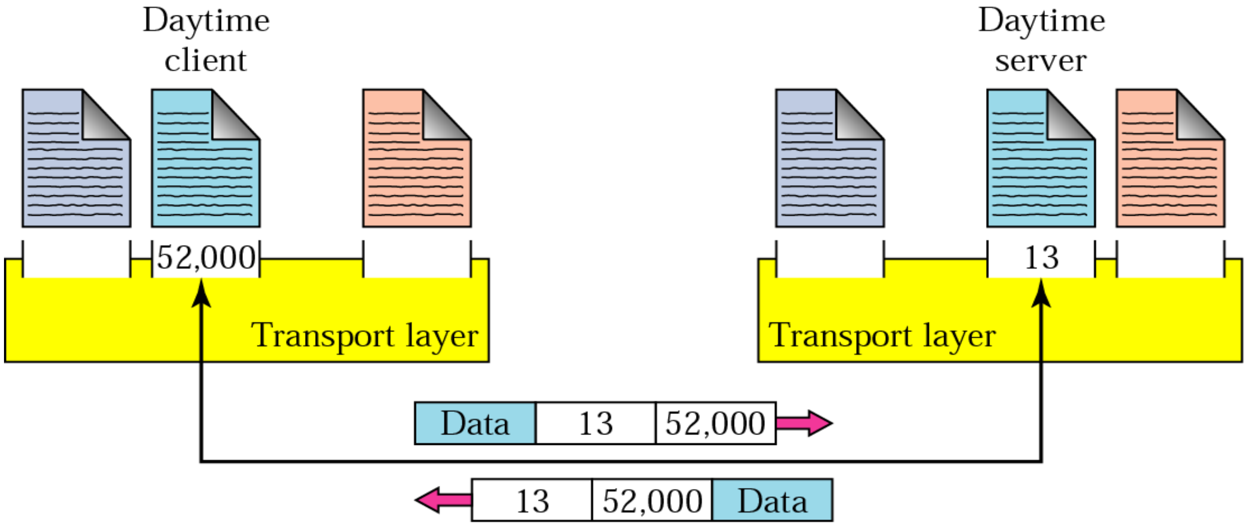
At receiver side, after error checking and header dropping, transport-layer delivers each message to appropriate process



Transport layer – Overview

Addressing

- Data link layer → MAC address
- Network layer → IP address
- Transport layer → Port number (choose among multiple processes running on destination host)



Transport layer – Overview

- Port numbers are 16-bit integers (0→65,535)
- Servers use well known ports.
  - DNS – port 53 (UDP)
  - FTP – port 21 (TCP)
  - HTTP – port 80 (TCP)
  - Mail service SMTP – port 25 (TCP)
  - SSH – port 22 (TCP)
- Clients use short-lived ports
- Server and Client can agree on a new port

If no transport layer protocols

- Typical limitations of the network:
    - Drop messages
    - Reorder messages
    - Limit messages to some finite size
    - Deliver messages after an arbitrarily long delay

- Network is providing a best-effort level of service – IP is an example

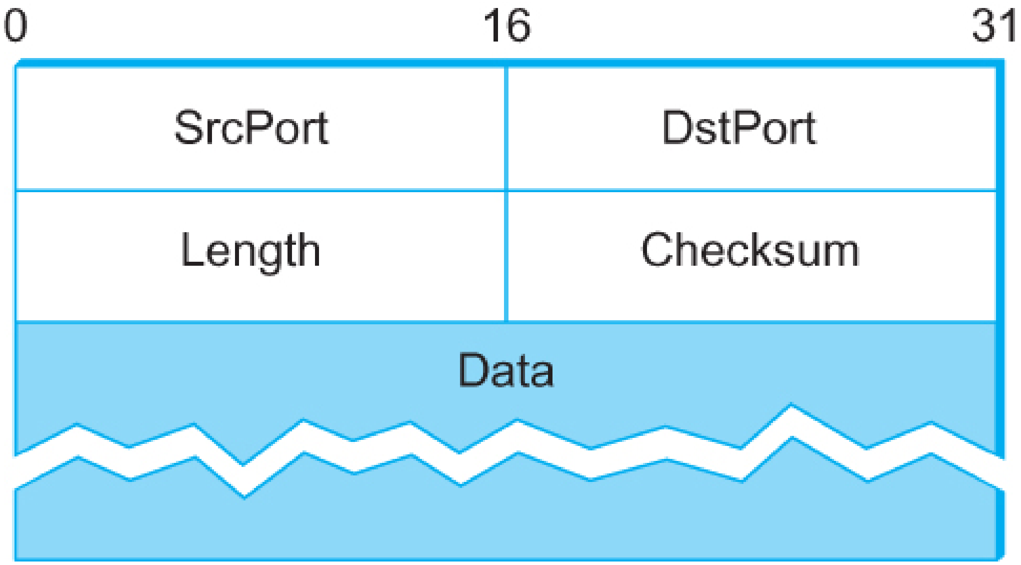
Transport layer – Capability

- A transport protocol promises to
  - Guarantee message delivery
  - Deliver messages in the correct order
  - Support arbitrarily large messages; multiple application processes on each host
  - Allow flow control, congestion control and QoS provisioning

Transport layer – Goal

Develop protocols that turn the less-than-desirable underlying network into the high level of service required by application programs!

User Datagram Protocol (UDP)



0 16 31

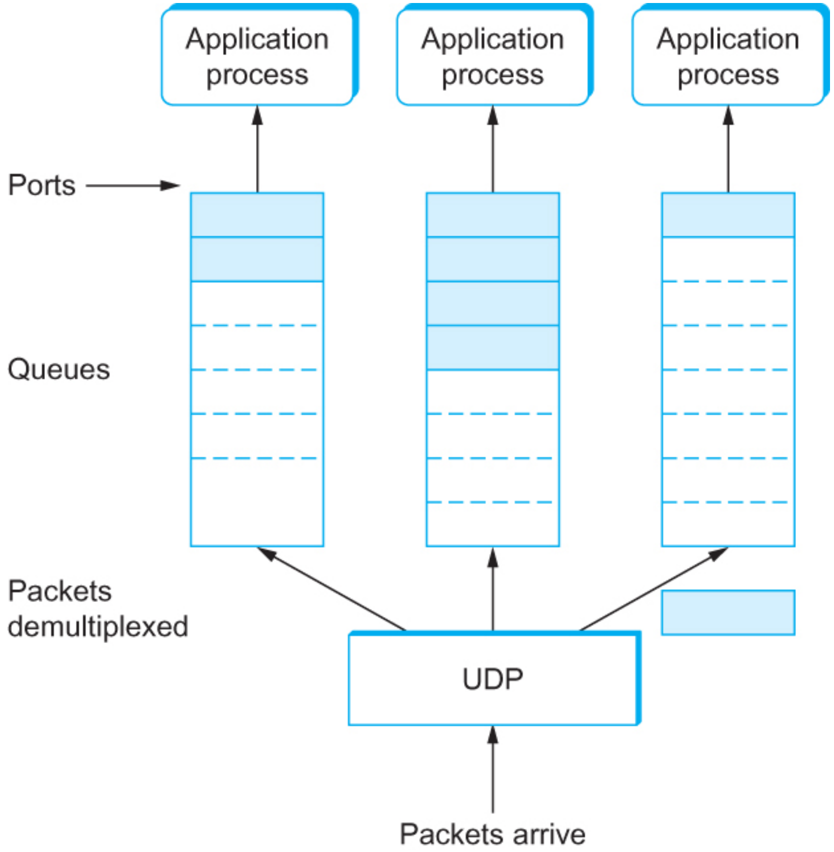
SrcPort DstPort

Length Checksum

Data

Format for UDP header

UDP – Simple Demultiplexer



UDP Message Queue

UDP – Properties

- UDP does not have a flow-control mechanism
- UDP does not implement reliable delivery
- But, good for burst data transfer, low-latency, (e.g., live-streaming), loss-tolerating applications

Transmission Control Protocol (TCP)

- TCP must perform functions:
  - Segmentation → breaks message into packets
  - End-to-end error control → since IP is an unreliable Service
  - End-to-end flow control → to avoid host buffer overflow
  - End-to-end congestion control → to avoid network congestion
  - Multiplexing and demultiplexing sessions

- TCP promises to be:
  - Reliable
  - Connection-oriented → virtual circuit
  - Stream-oriented → users exchange streams of data
  - Full duplex → concurrent transfers can take place in both directions
  - Buffered → TCP accepts data and transmits when appropriate

Flow Control vs Congestion Control

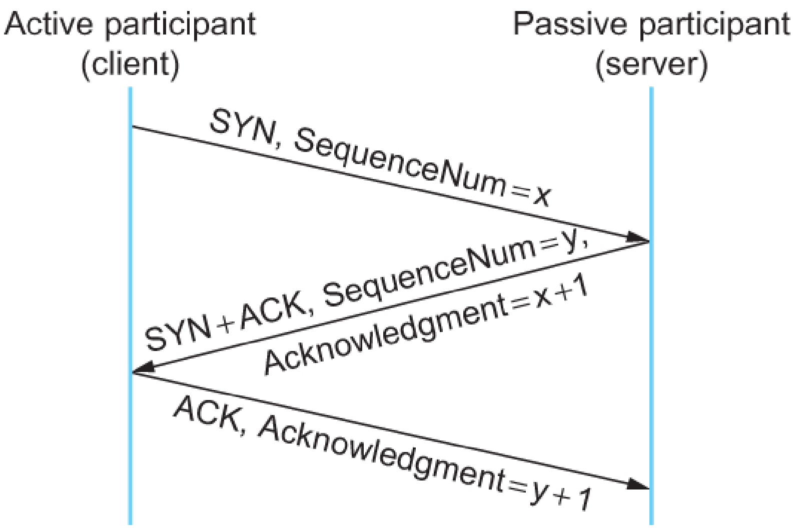
- Flow control - prevent senders from overrunning the capacity of the receivers
- Congestion control - prevent too much data from being injected into the network, thereby causing switches or links to become overloaded

TCP Design – Overview

- Reliable
  - Requires ACK and performs retransmission;
  - If ACK not received, retransmit;
  - After a number of retransmissions, give up;
  - How long to wait for ACK? (next lecture)

TCP Design – Overview

- Connection-Oriented : Connection Establishment



Timeline for three-way handshake algorithm

x is starting sequence number for client – selected at random

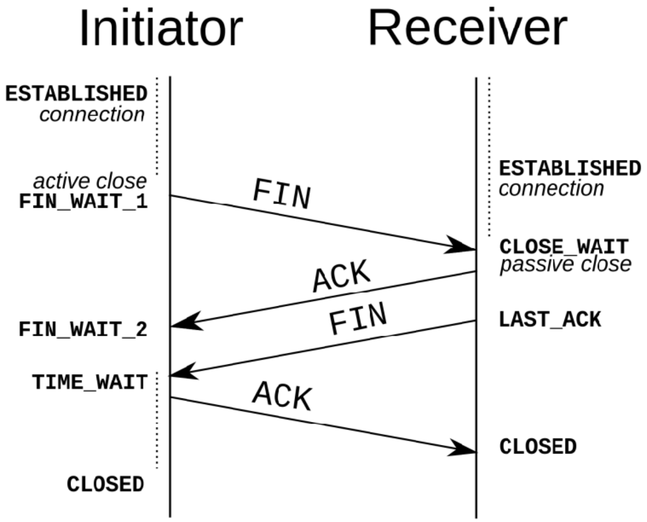
y is starting sequence number for server – selected at random

(want to avoid reusing same sequence numbers to soon)

Ack value identifies **next sequence number expected**

TCP Design – Overview

- Connection-Oriented : Connection Termination



Timeline for four-way handshake algorithm

Fin: Finish (data with SeqNum = x)

ACK: SeqNum = x+1

FIN: Finish (data with SeqNum = y)

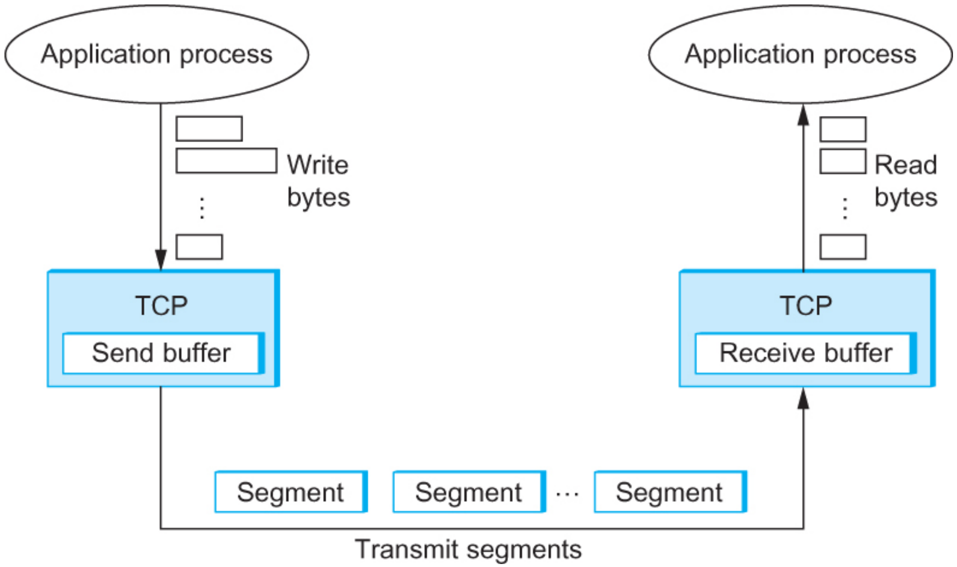
ACK: SeqNum = y+1

TCP Design – Overview

- Correct order
  - Use sequence numbers (next lecture)
  - Associated with every byte that it sends
  - To detect packet loss, reordering and duplicate removal
  - To protect replay attack

TCP Design – Overview

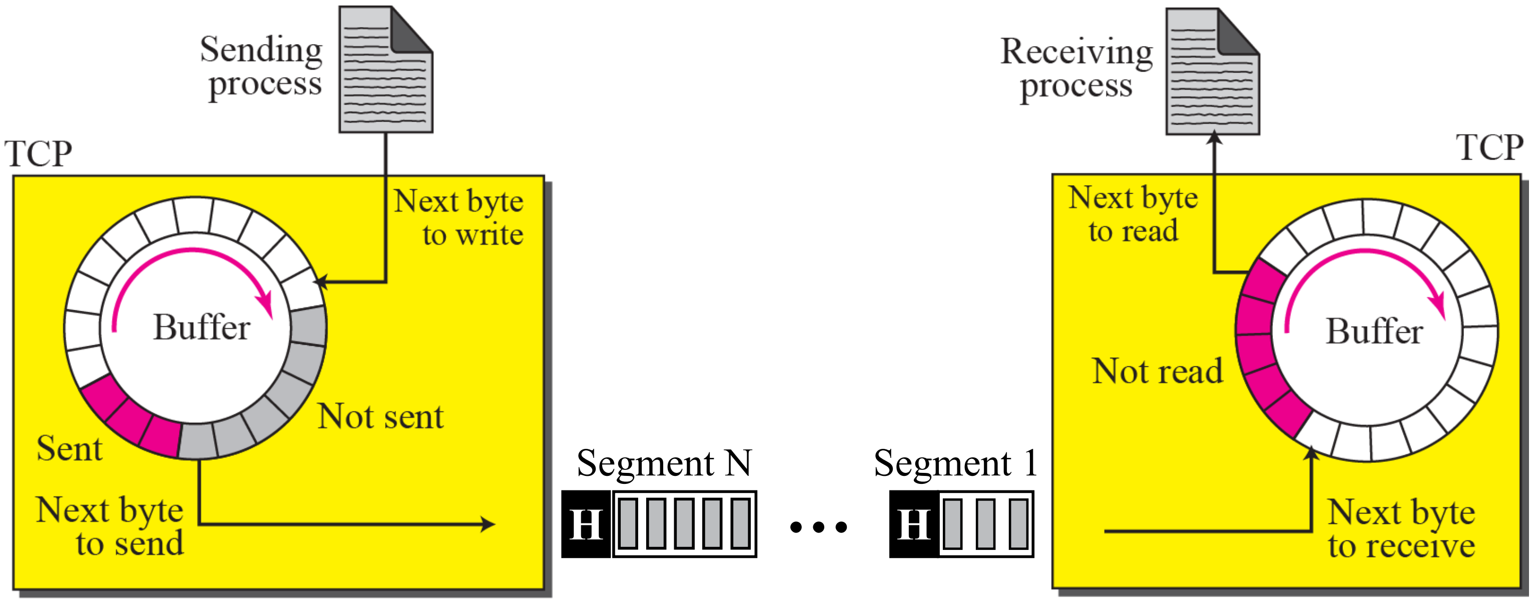
- Congestion and Flow Control
  - Segmentation, buffers and sliding window (future lectures)
  - TCP is a byte-oriented protocol



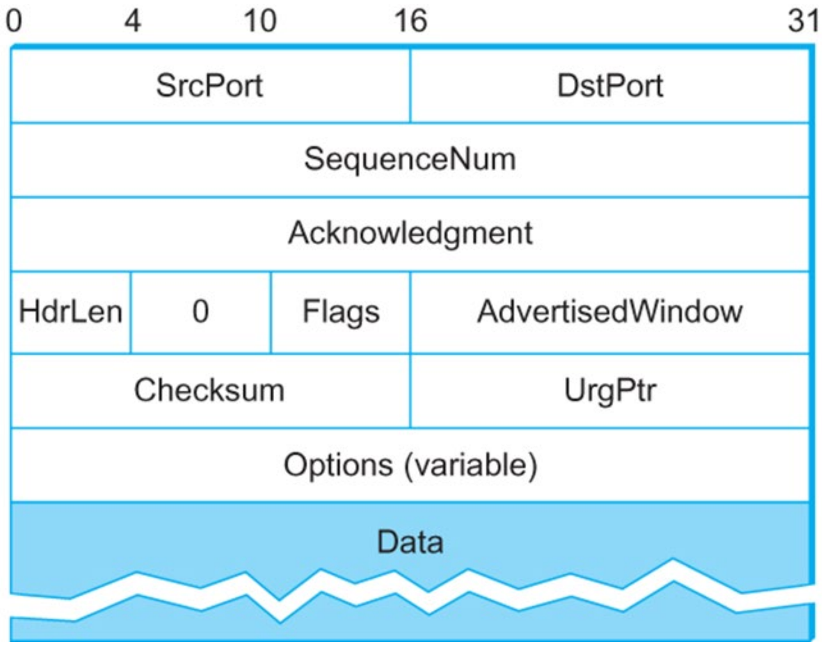
How TCP manages a byte stream.

TCP Design – Overview

- Congestion and Flow Control
  - Segmentation, buffers and sliding window (future lectures)
  - TCP is a byte-oriented protocol



TCP Header



0 4 10 16 31

SrcPort | DstPort

SequenceNum

Acknowledgment

HdrLen | 0 | Flags | AdvertisedWindow

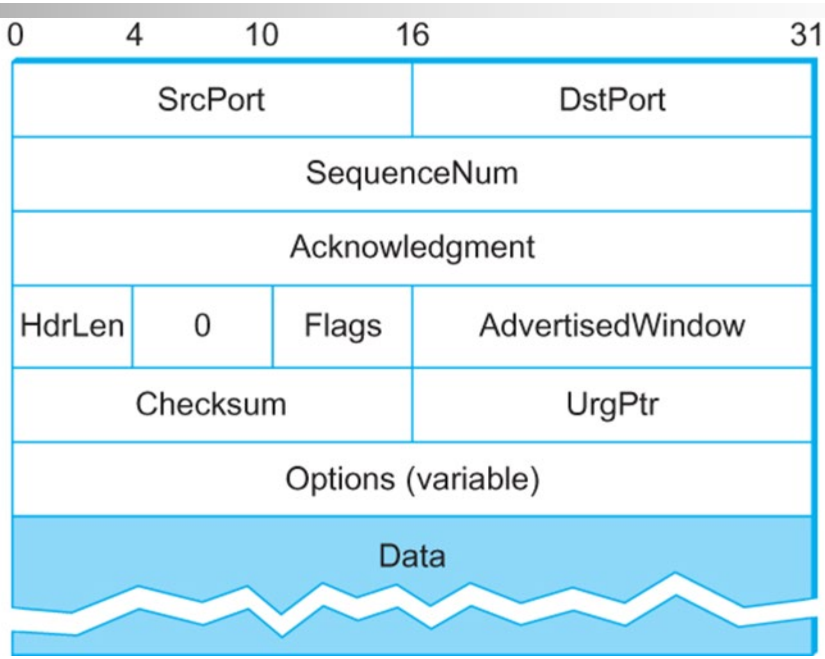
Checksum | UrgPtr

Options (variable)

Data

- The SrcPort and DstPort fields identify the source and destination ports, respectively.
- The Acknowledgment, SequenceNum, and AdvertisedWindow fields are all involved in TCP’s sliding window algorithm.

TCP Header



- Because TCP is a byte-oriented protocol, each byte of data has a sequence number;
- the SequenceNum field contains the sequence number for the first byte of data carried in that segment.
- The Acknowledgment and AdvertisedWindow fields carry information about the flow of data going in the other direction.

TCP Header

- The Checksum field is used in exactly the same way as for UDP—it is computed over
  - the TCP header,
  - the TCP data, and
  - the pseudo header, which is made up of the source address, destination address, and length fields from the IP header.