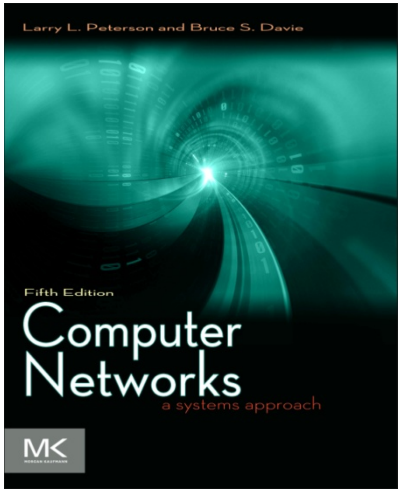


Chapter 5

End-to-End Protocols

5.1 & 5.2

Problem

- How to turn this host-to-host packet delivery service into a process-to-process communication channel

Chapter Outline

- Simple Demultiplexer (UDP)
- Reliable Byte Stream (TCP)

Chapter Goal

- Understanding the demultipexing service
- Discussing simple byte stream protocol

Problem: How to turn this host-to-host packet delivery service into a process-to-process communication channel

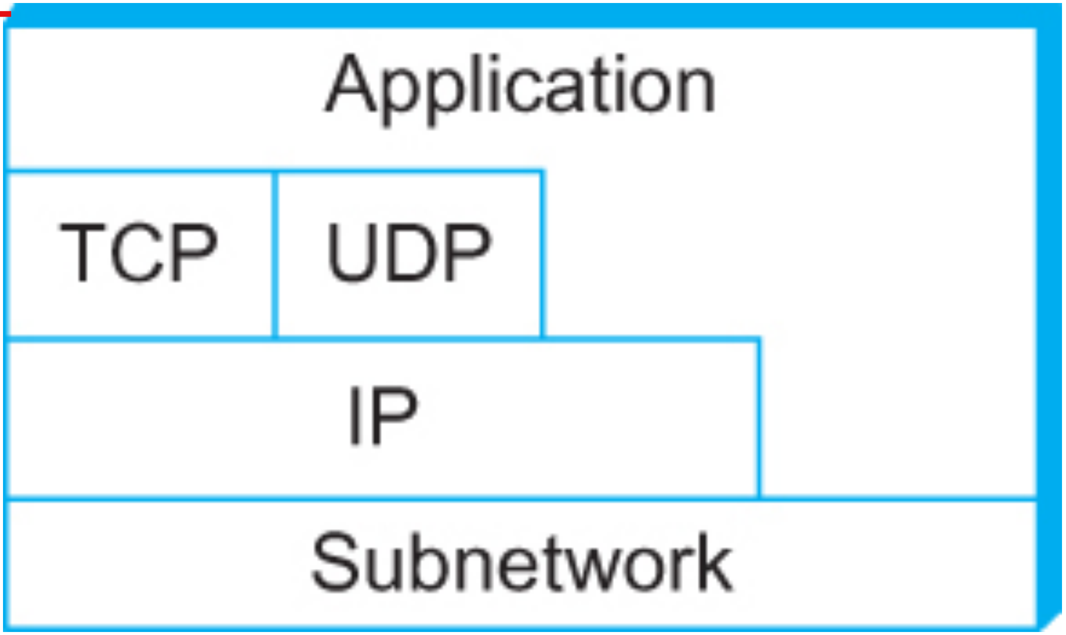
Chapt Outline

- Simple Demultiplexer (UDP)

- Reliable Byte Stream (TCP)

End-to-end Protocols

- Process-to-process communication is accomplished by the transport layer
- Communication between application programs running in end nodes
- This end-to-end protocol sits between application layer and the network layer



Transport layer =>

Network layer =>

End-to-end Protocols

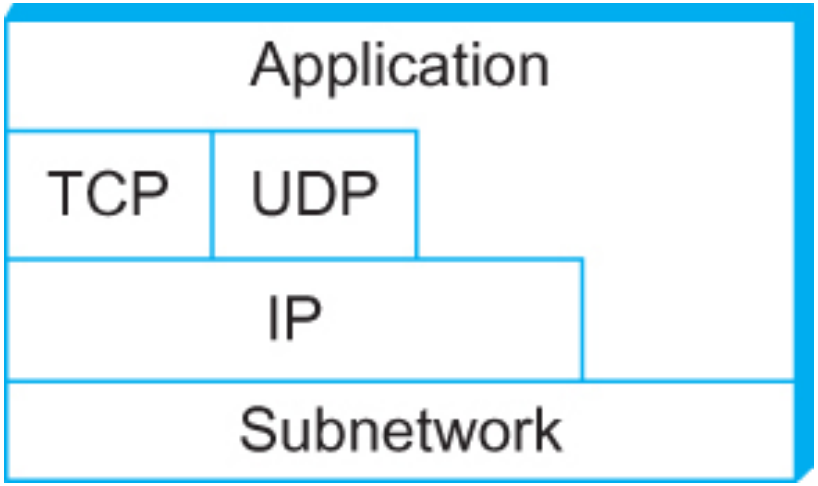
- Common properties that a transport protocol can be expected to provide:
  - Guarantees message delivery
  - Delivers messages in the same order they were sent
  - Delivers at most one copy of each message
  - Supports arbitrarily large messages
  - Supports synchronization between the sender and the receiver
  - Allows the receiver to apply flow control to the sender
  - Supports multiple application processes on each host

End-to-end Protocols

- Typical limitations of the network on which transport protocol will operate
  - Drop messages
  - Reorder messages
  - Deliver duplicate copies of a given message
  - Limit messages to some finite size
  - Deliver messages after an arbitrarily long delay

Transport layer =>

Network layer =>



End-to-end Protocols

- Challenge for Transport Protocols
  - Develop algorithms that turn the less-than-desirable properties of the underlying network into the high level of service required by application programs
- This chapter looks at different transport protocols employing algorithms in the context of four representative services

1. A simple asynchronous demux service (UDP)
2. A reliable byte-stream service (TCP)
3. ~~A request/reply service (RPC)~~ **not going to cover**
4. ~~A service for real-time applications (RTP)~~

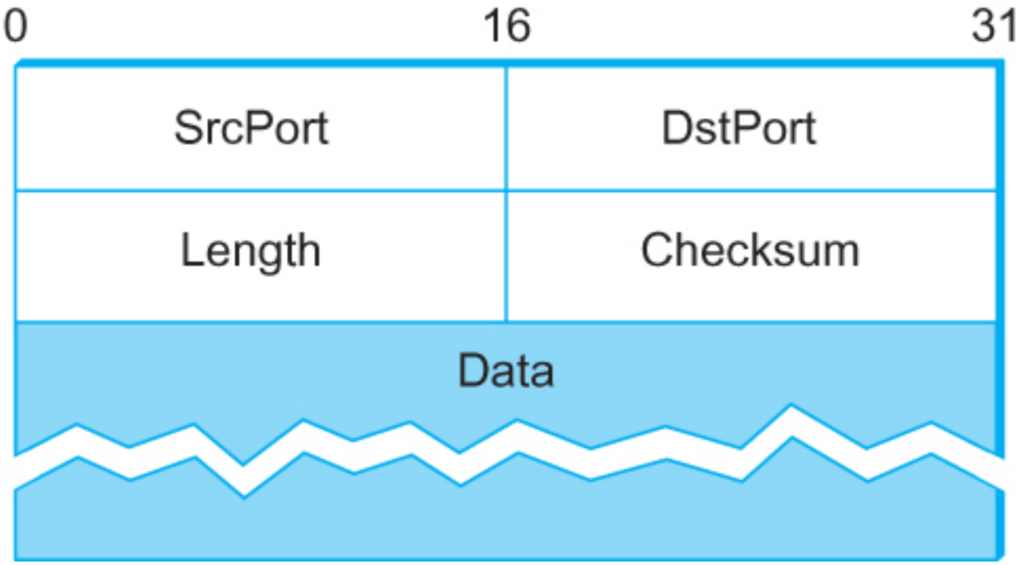
Simple Demultiplexer (UDP)

Look at how it act like demultiplexer tool

- Extends host-to-host delivery service into a process-to-process communication service
- Allows multiple application processes on each host to share the network (multiplexing/ demultiplexing)
- Same processes on end nodes can identify each other through ports
  - Source process to send a message to a port and the destination process to receive the message from a port

Simple Demultiplexer (UDP)

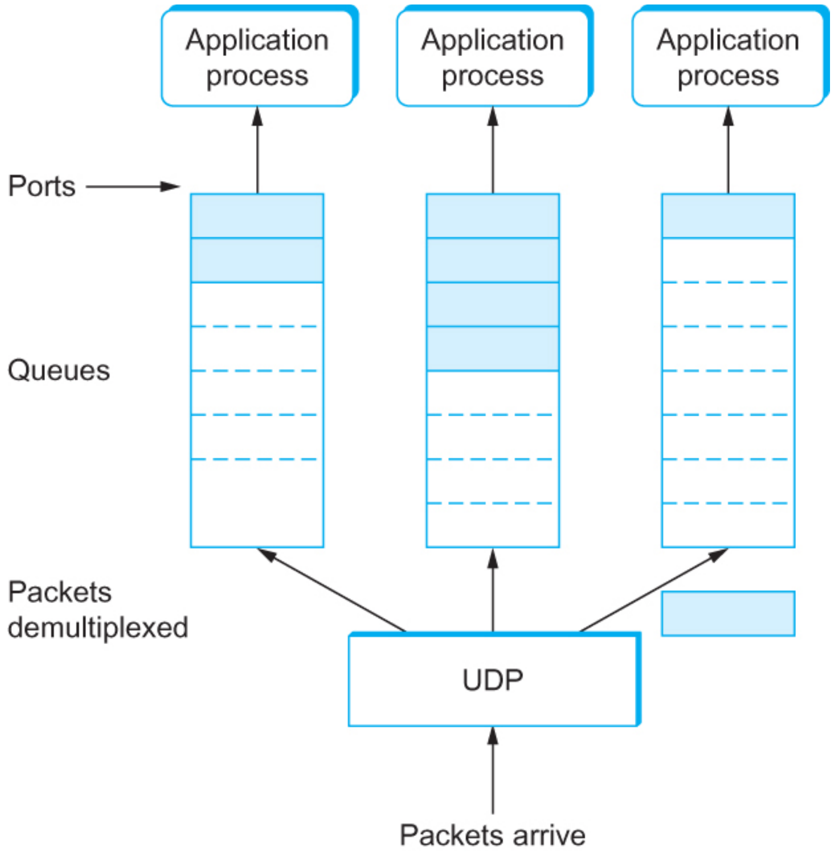
If you needed to implement a live streaming service and did not have plenty of bandwidth, it would be best to use UDP for the transport protocol.



Format for UDP header

List of UDP port number/description chart such as:
456 for DHCP Client, 156 SQL server, …etc

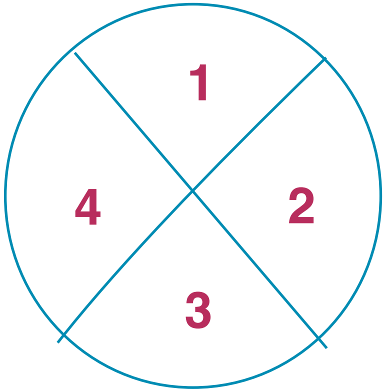
Simple Demultiplexer (UDP)

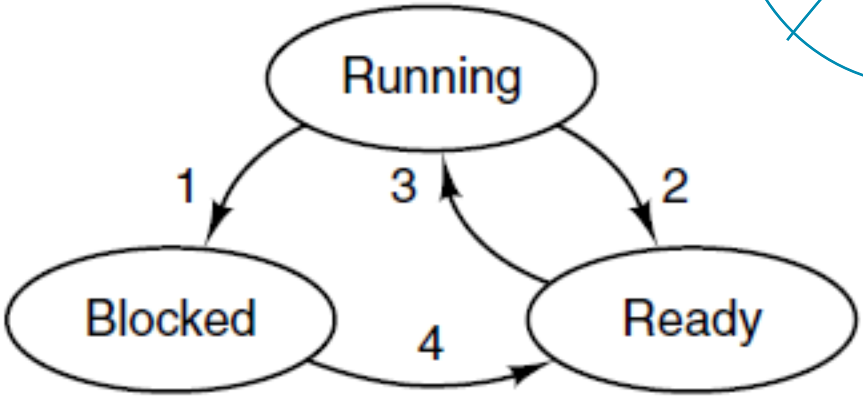


UDP Message Queue

Simple Demultiplexer (UDP)

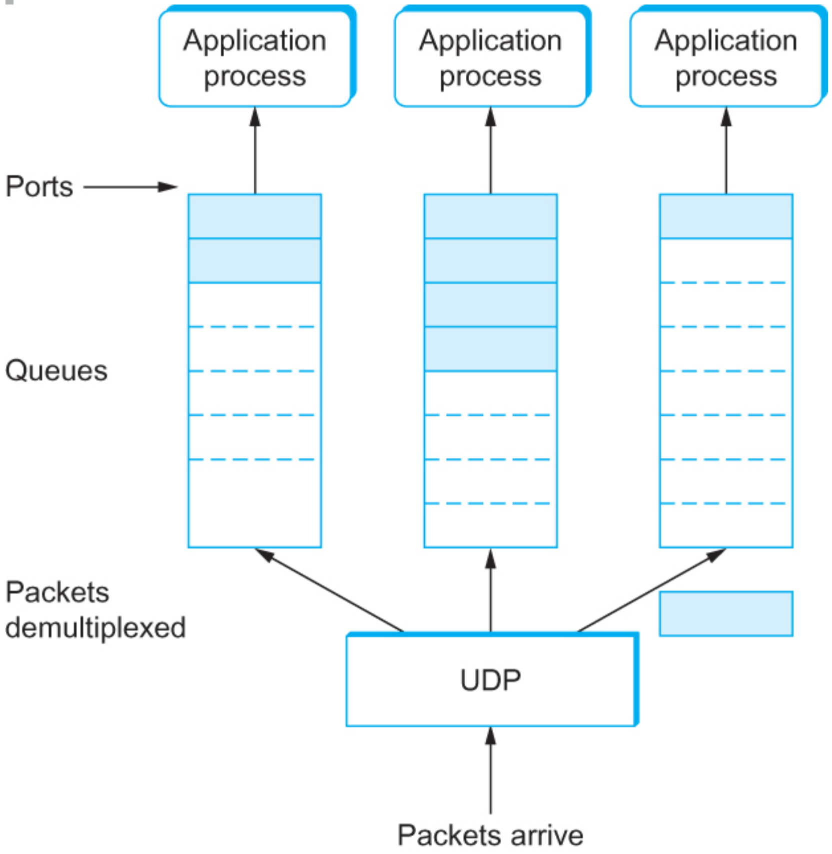
Process States



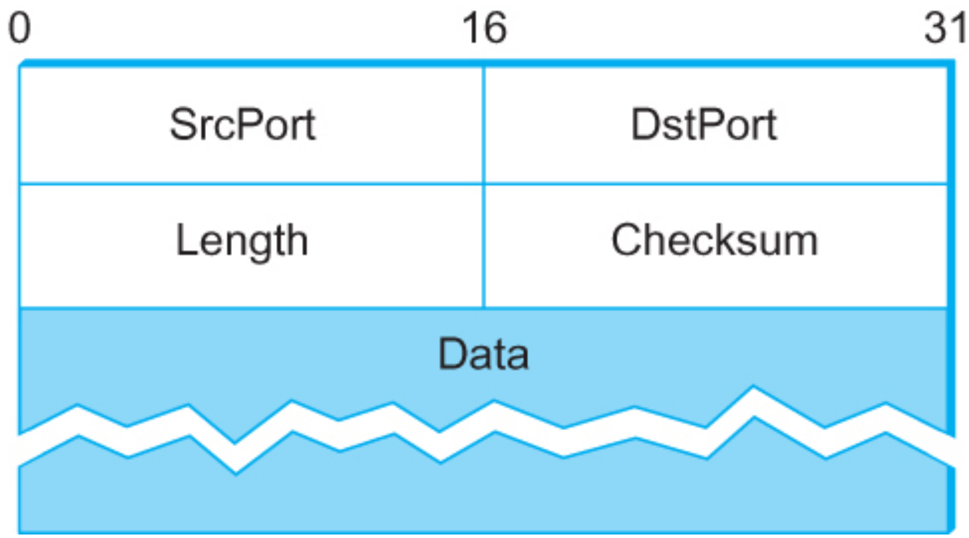


1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Simple Demultiplexer (UDP)



UDP Message Queue



Format for UDP header

https://www.youtube.com/watch?v=v3WeRfpvmG8

Reliable Byte Stream (TCP)

- In contrast to UDP, Transmission Control Protocol (TCP) offers the following services
  - Reliable
  - Connection oriented
  - Byte-stream service

Reliable Byte Stream (TCP)

- Guarantees reliable, in-order delivery of a stream of bytes
- It is full-duplex; supports a pair of byte streams, one flowing in each direction. **allowing both parties to send and receive data within the context of the single TCP connection**
- It includes flow-control mechanism, allowing the receiver to limit how much data the sender can transmit at a given time
- **mux/demux**
  It supports a demultiplexing mechanism, allowing multiple applications programs on any given host to simultaneously carry on a conversation with their peers.
- It also implements a highly tuned congestion-control mechanism
  - to keep the sender from overloading the network

Flow control VS Congestion control

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

The congestion window (cwnd) is controlled by sender side.
Flow control is controlled by receiver side.

CW is based on the network capacity and conditions. It is usually referred to in multiples of maximum segment size (MSS). The cwnd is initially increased by TCP Slow Start.

End-to-end Issues

- The following issues need to be addressed by TCP in a connectionless approach:
  - Supporting logical connections between processes running on two different computers in the Internet
  - Connections are likely to have widely different RTT times
  - Packets may get reordered in the Internet
  - Each side of a connection must be able to learn what resources the other side is able to apply to the connection
  - The sending side must be able to learn what the capacity of the network is (congestion control).

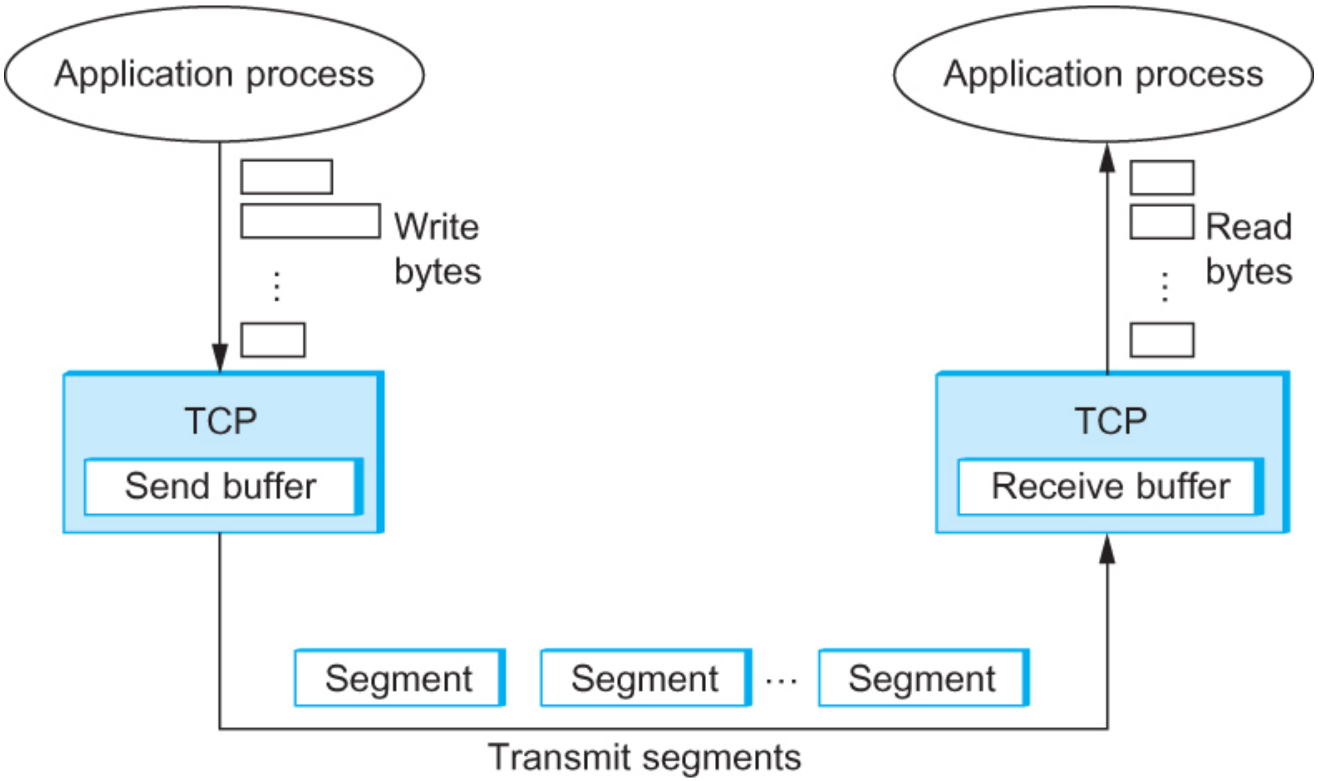
TCP Segment

- TCP is a byte-oriented protocol

TCP will put enough incoming bytes together into segment, then travel to other point, put send buffer to receive buffer, and deliver to application of the other host

- TCP does not, itself, transmit individual bytes over the Internet.
- So what is Protocol Data Unit (PDU) of the TCP Protocol?

TCP Segment



How TCP manages a byte stream.

Triggering Transmission

- TCP supports a byte stream abstraction
- Application programs write bytes into streams
- It is up to TCP to decide that it has enough bytes to send a segment

Triggering Transmission

- Other factors governing this decision
  - Ignore flow control: window is wide open, as would be the case when the connection starts
  - TCP has three mechanism to trigger the transmission of a segment

1) Congestion control 2) MSS 3) push operation

max segment size

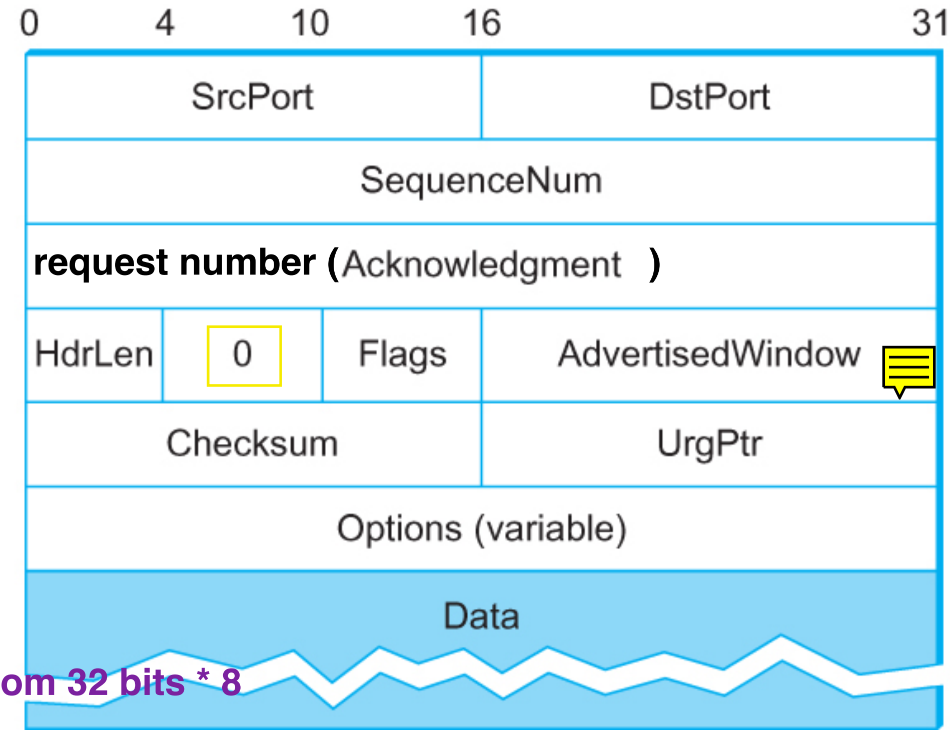
1) TCP maintains a variable MSS and sends a segment as soon as it has collected MSS bytes from the sending process

- MSS is usually set to the size of the largest segment TCP can send without causing local IP to fragment.
- MSS: MTU of directly connected network – (TCP header + and IP header)

2) Sending process has explicitly asked TCP to send it

- TCP supports push operation

TCP Header



SrcPort			DstPort
SequenceNum			
request number (Acknowledgment)			
HdrLen	0	Flags	AdvertisedWindow
Checksum			UrgPtr
Options (variable)			
Data			

each ward
4 bytes x 5
= 20 bytes
before we
get to
opions

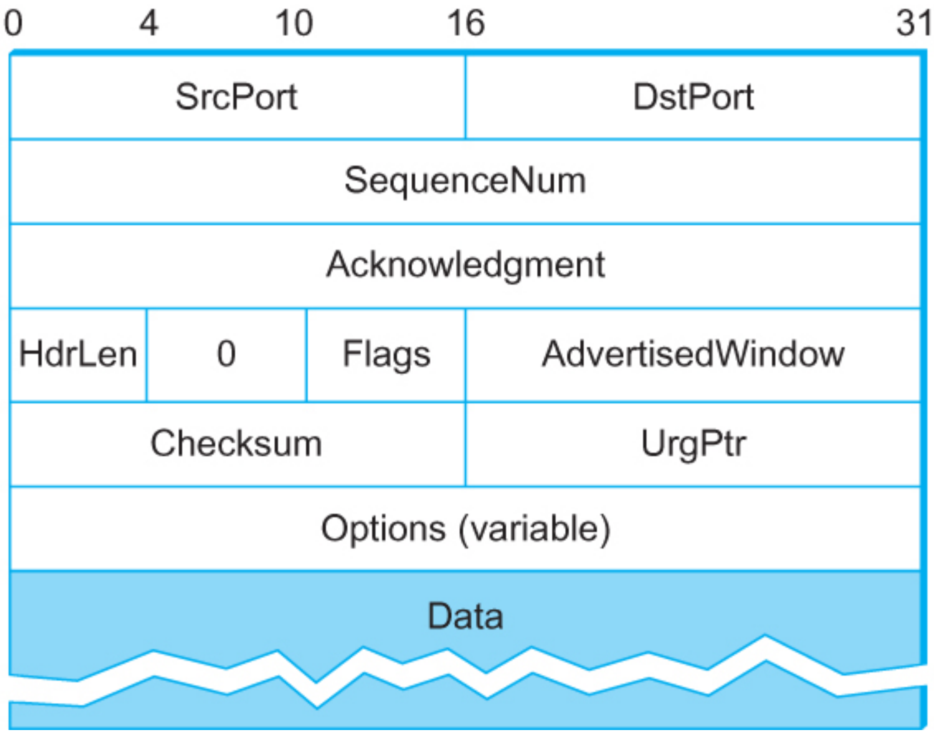
4 bytes come from 32 bits * 8

Every byte has sequence no#

TCP Header Format

https://www.youtube.com/watch?v=zneZTNI8WP8

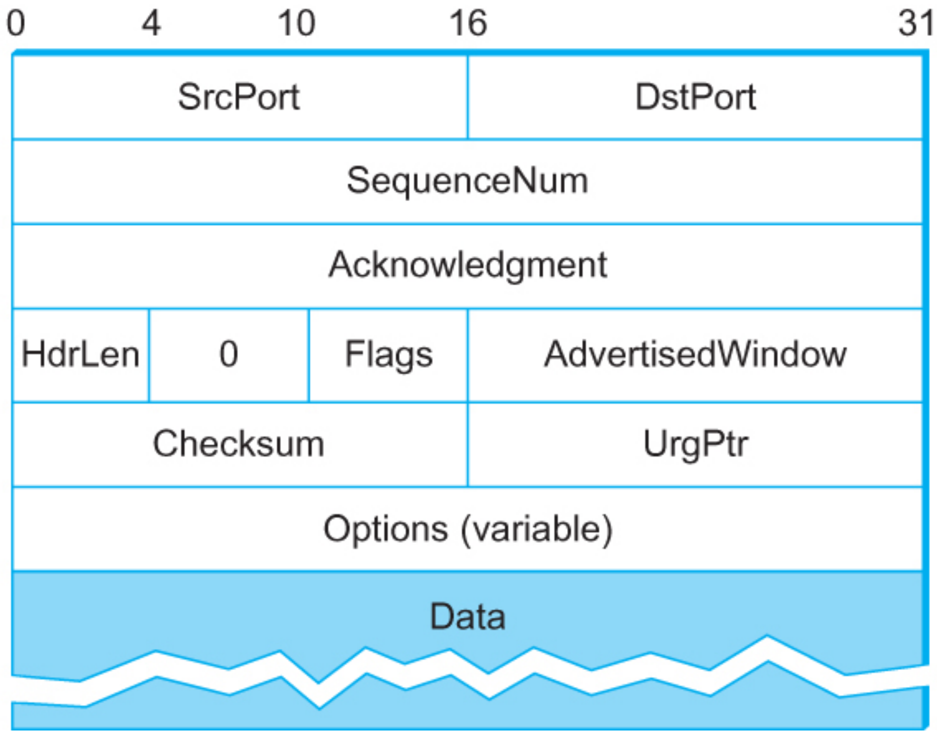
TCP Header



max of 5 and 15 ward because header is variable
so we tell where data starts

TCP Header Format

TCP Header



0 4 10 16 31

SrcPort DstPort

SequenceNum

Acknowledgment

HdrLen 0 Flags AdvertisedWindow

Checksum UrgPtr

Options (variable)

Data

TCP Header Format

TCP Header

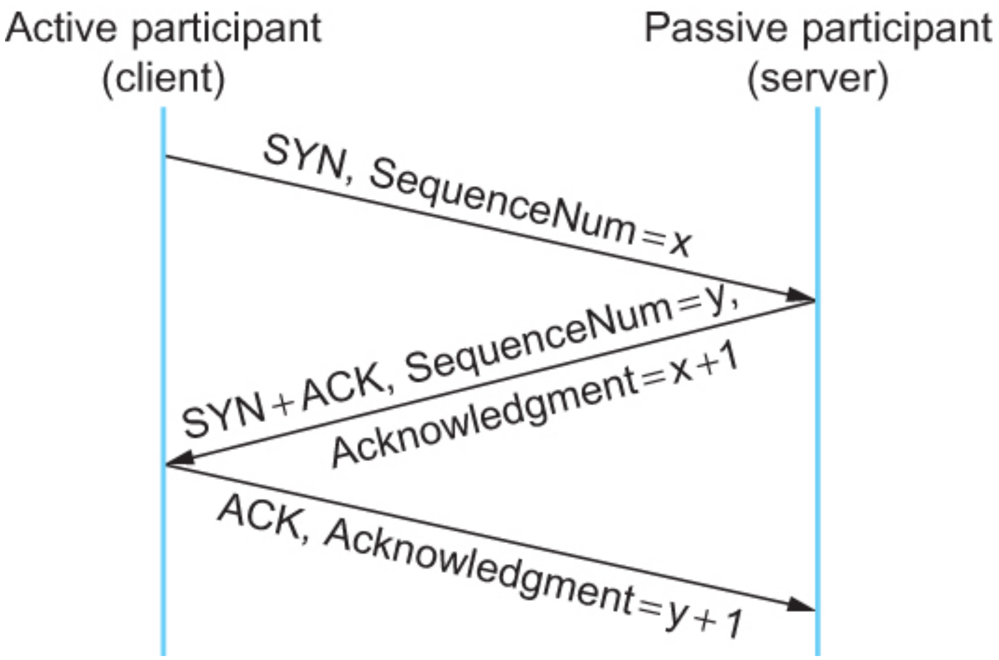
- The 6-bit Flags field is used to relay control information between TCP peers.
- The possible flags include SYN, FIN, RESET, PUSH, URG, and ACK.
- The SYN and FIN flags are used when establishing and terminating a TCP connection, respectively.
- The ACK flag is set any time the Acknowledgment field is valid, implying that the receiver should pay attention to it.
  - In other words, this is an Acknowledgement message

TCP Header

- The URG flag signifies that this segment contains urgent data. When this flag is set, the UrgPtr field indicates where the nonurgent data contained in this segment begins.
- The urgent data is contained at the front of the segment body, up to and including a value of UrgPtr bytes into the segment.
- The PUSH flag signifies that the sender invoked the push operation, which indicates to the receiving side of TCP that it should notify the receiving process of this fact.
- Finally, the RESET flag signifies that the receiver has become confused. **Invalid Ack or SeqNum values**

Connection Establishment/Termination in TCP

Syn frm Sen’, Syn-ACK frm Rec’, Ack frm Sen’ => established both way



Timeline for three-way handshake algorithm

Summary

- We have discussed how to convert host-to-host packet delivery service to process-to-process communication channel.
- We have discussed UDP
- We have discussed TCP