CSCI 466: Networks

UDP and TCP

Reese Pearsall

Fall 2023

Announcements

PA 2 Posted. Due Wednesday October 18^{th}

Wireshark Lab 2 Posted. Due Friday October 13th

Sending objects through sockets example

PA2 Demo

(time.sleep)

OSI Model

Application Layer

Presentation Layer *

Session Layer *

Transport Layer

Network Layer

Data Link Layer

Physical Layer

Application Layer

Messages from Network Applications

Physical Layer

Bits being transmitted over some medium

**In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used*

Transport Layer Protocols:

1. Transmission Control Protocol (TCP)
2. User Datagram Protocol (UDP)

UDP

- "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

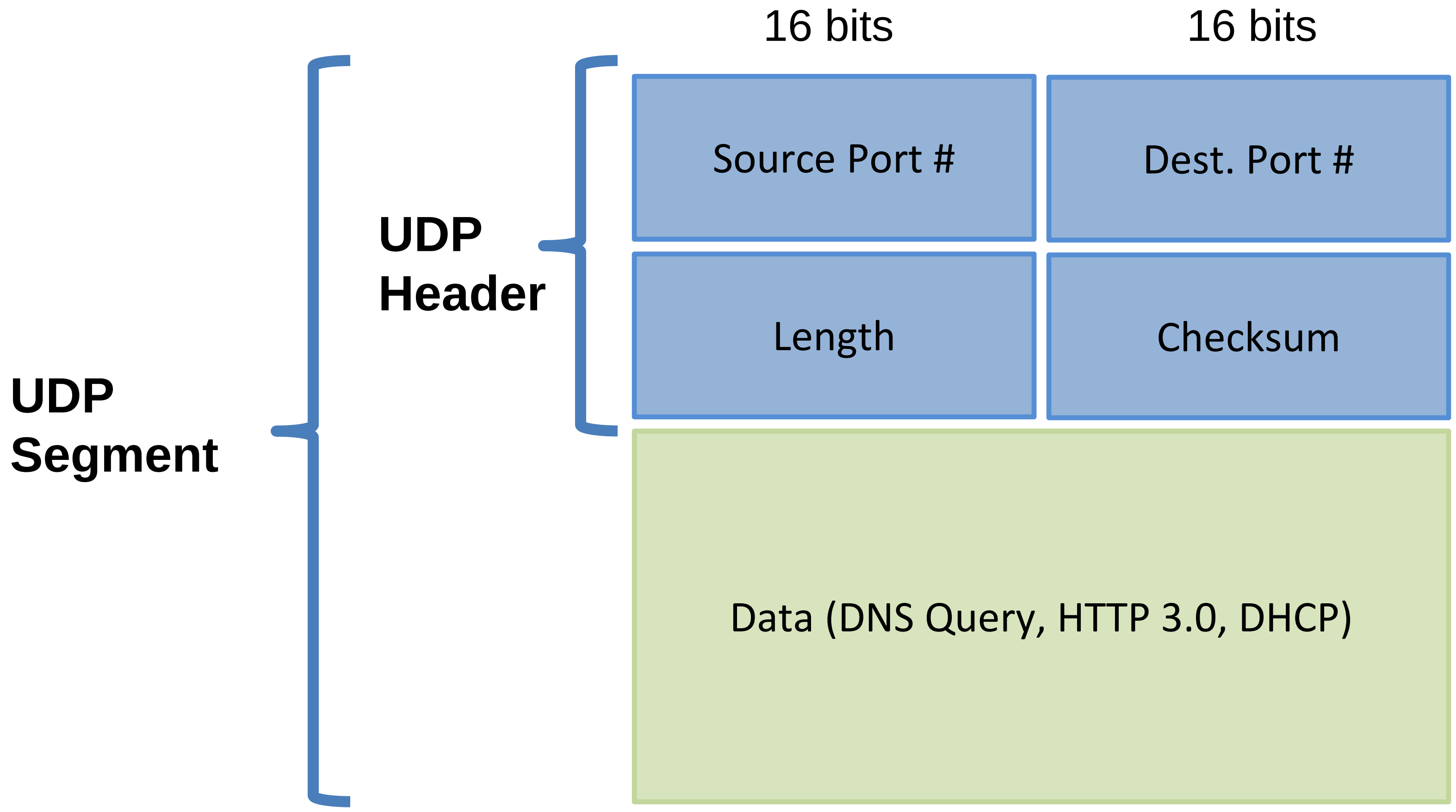
Why is there a UDP?

- no connection establishment (which can add RTT delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control
  - UDP can blast away as fast as desired!
  - can function in the face of congestion

Transport Layer

UDP

The UDP header is very small!! (8 bytes, 64 bits)

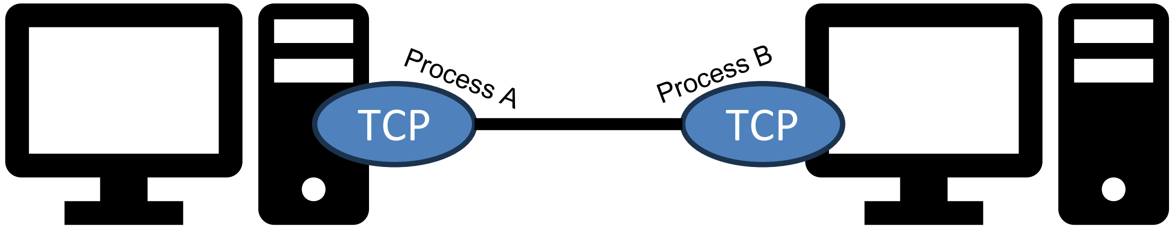


TCP

- Connection oriented, point-to-point (1 to 1)

→ **TCP Handshake** must occur before data is being transmitted

A *logical* connection



- Reliable, in order, data transfer

- Cumulative ACKs

- Pipelining

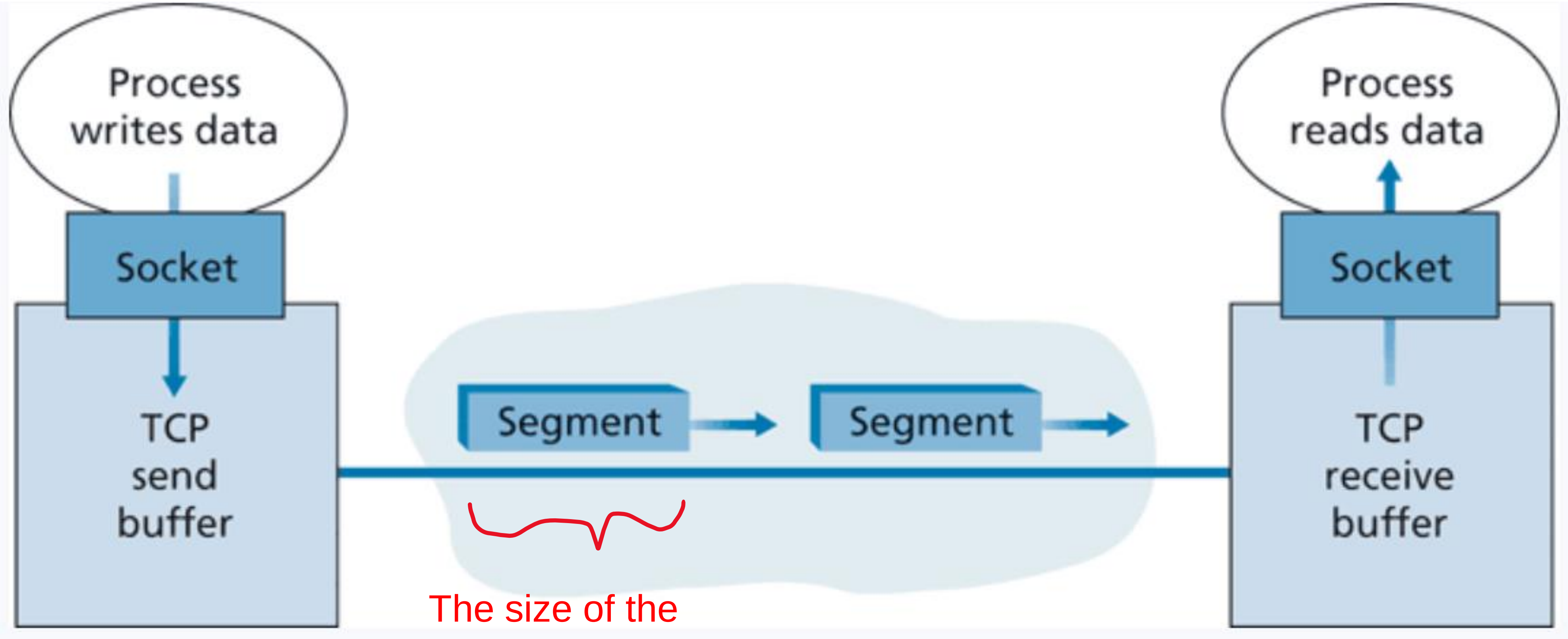
→ TCP Congestion and flow control set window size

- Flow controlled

→ Sender will not overwhelm receiver

- Full-duplex service

TCP

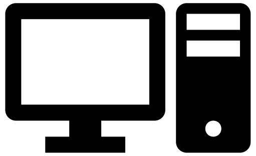


The size of the segment is determined by the maximum segment size (MSS)

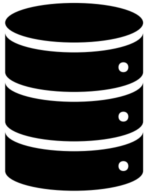
(Roughly 1500 bytes– size of a link layer frame)

TCP Sequence Numbers

A TCP connection is transmitting a **byte stream**



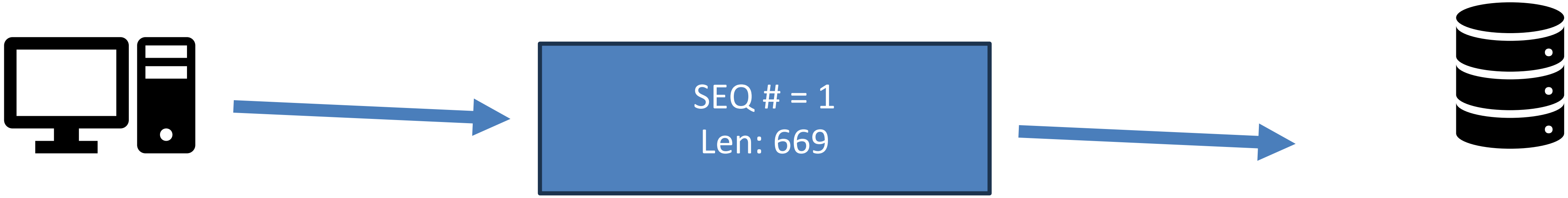
Sequence numbers are based on *how much data has been sent*
Acknowledgement numbers are based on *how much data has been successfully received*



Transport Layer

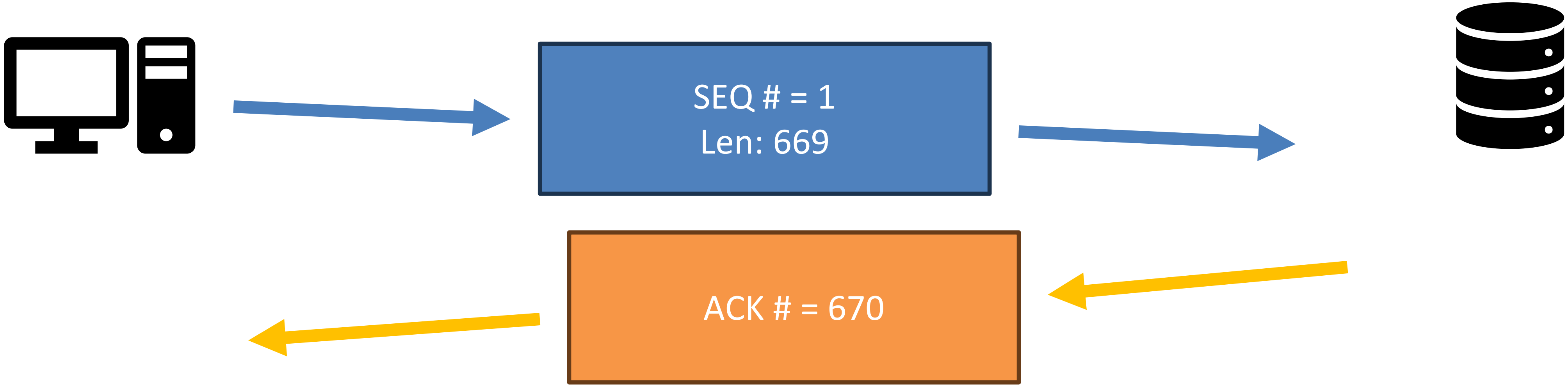
TCP Sequence Numbers

A TCP connection is transmitting a **byte stream**



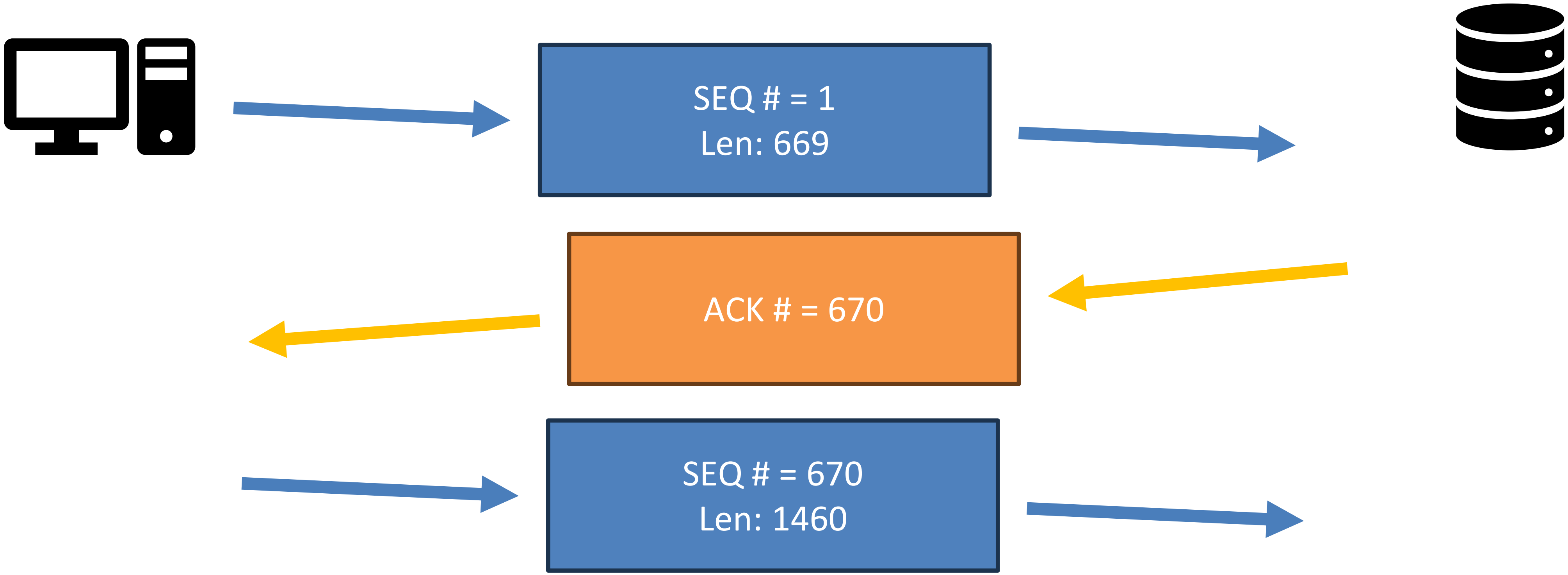
TCP Sequence Numbers

A TCP connection is transmitting a **byte stream**



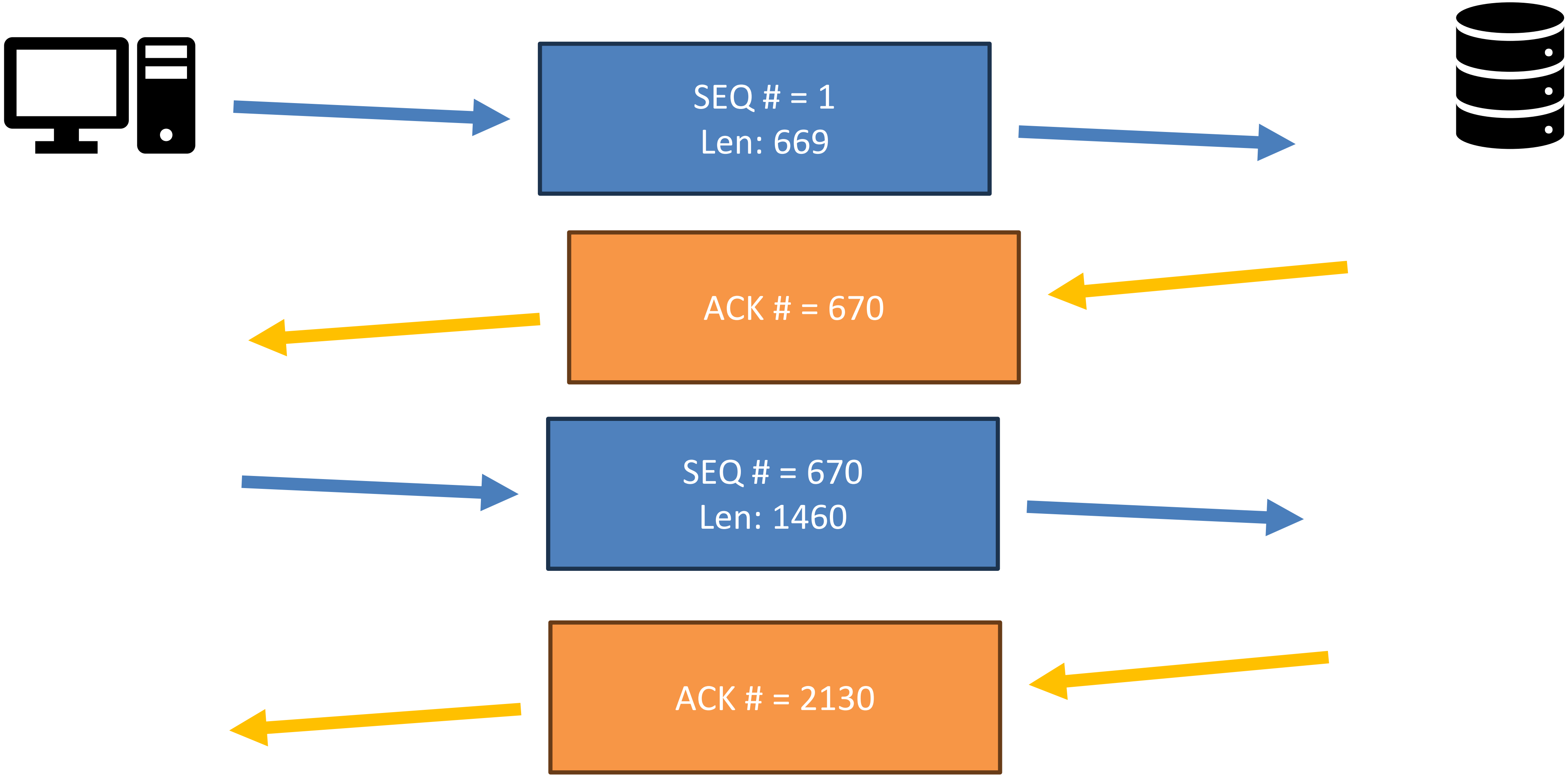
TCP Sequence Numbers

A TCP connection is transmitting a **byte stream**



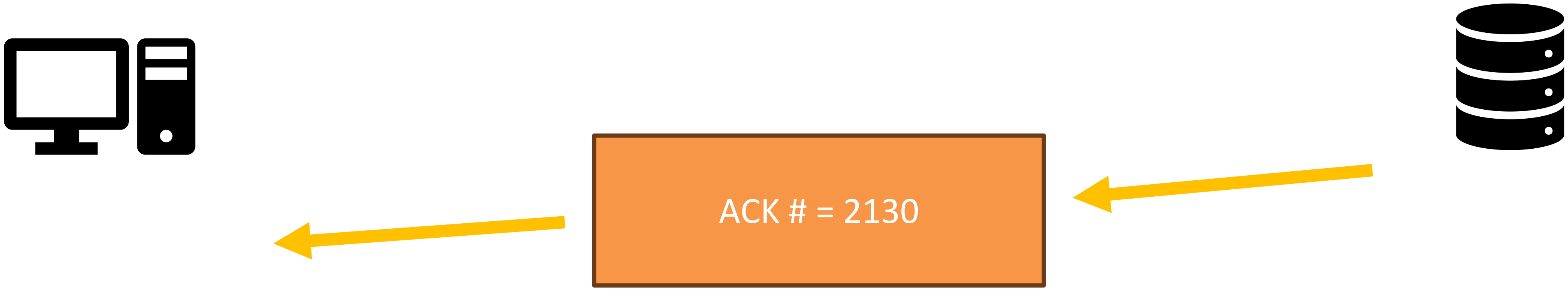
TCP Sequence Numbers

A TCP connection is transmitting a **byte stream**



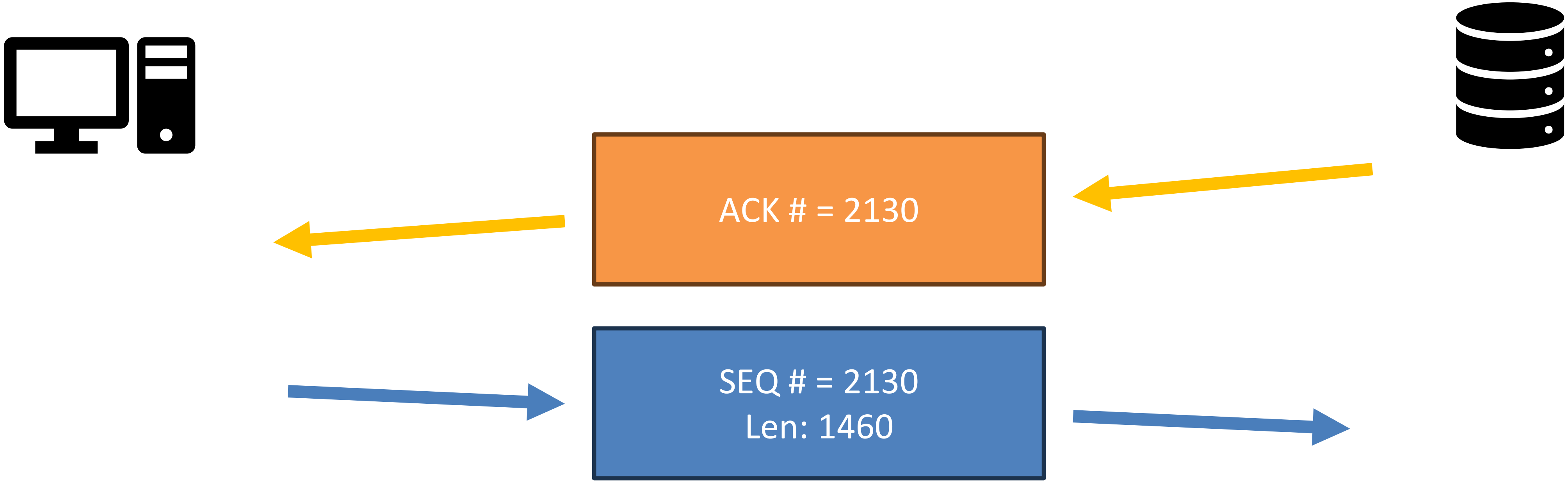
TCP Sequence Numbers

A TCP connection is transmitting a **byte stream**



TCP Sequence Numbers

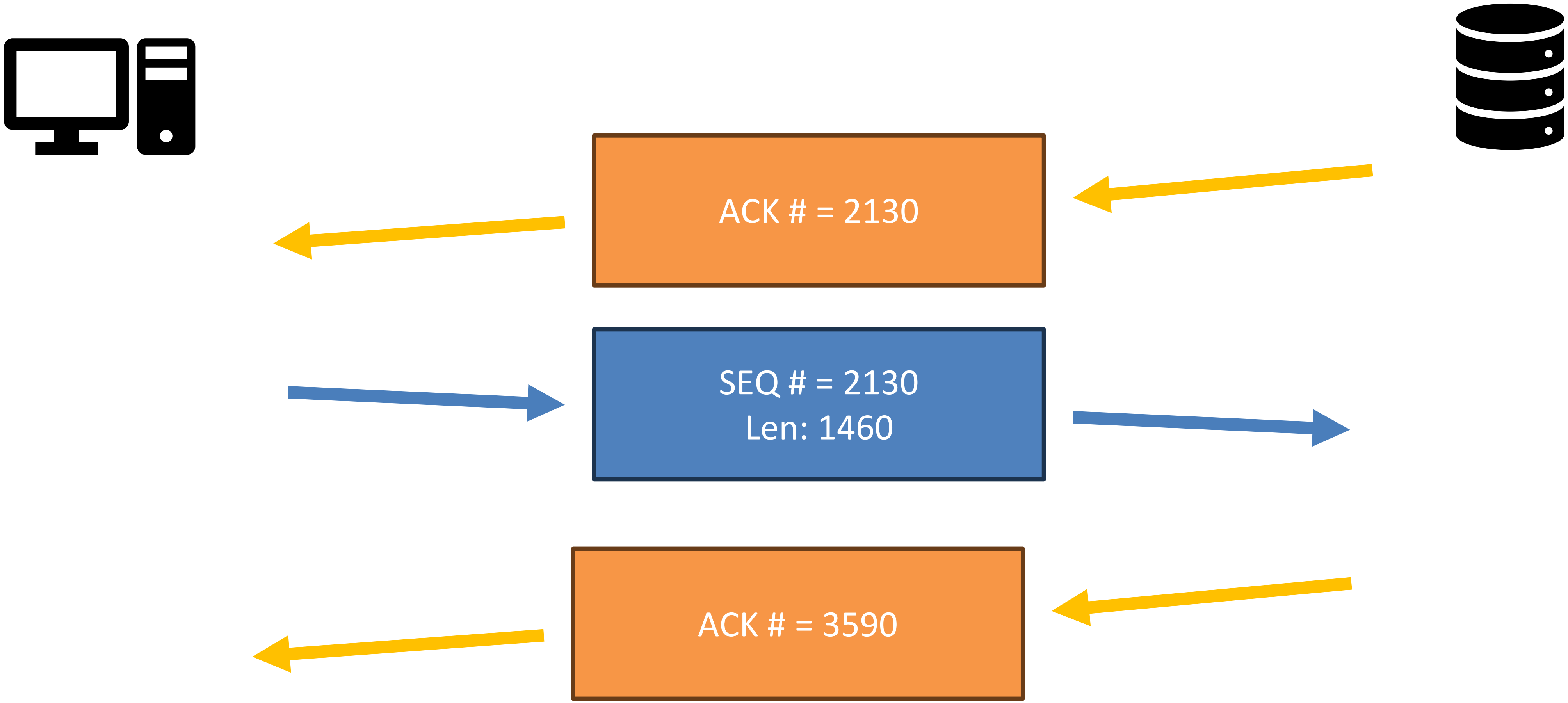
A TCP connection is transmitting a **byte stream**



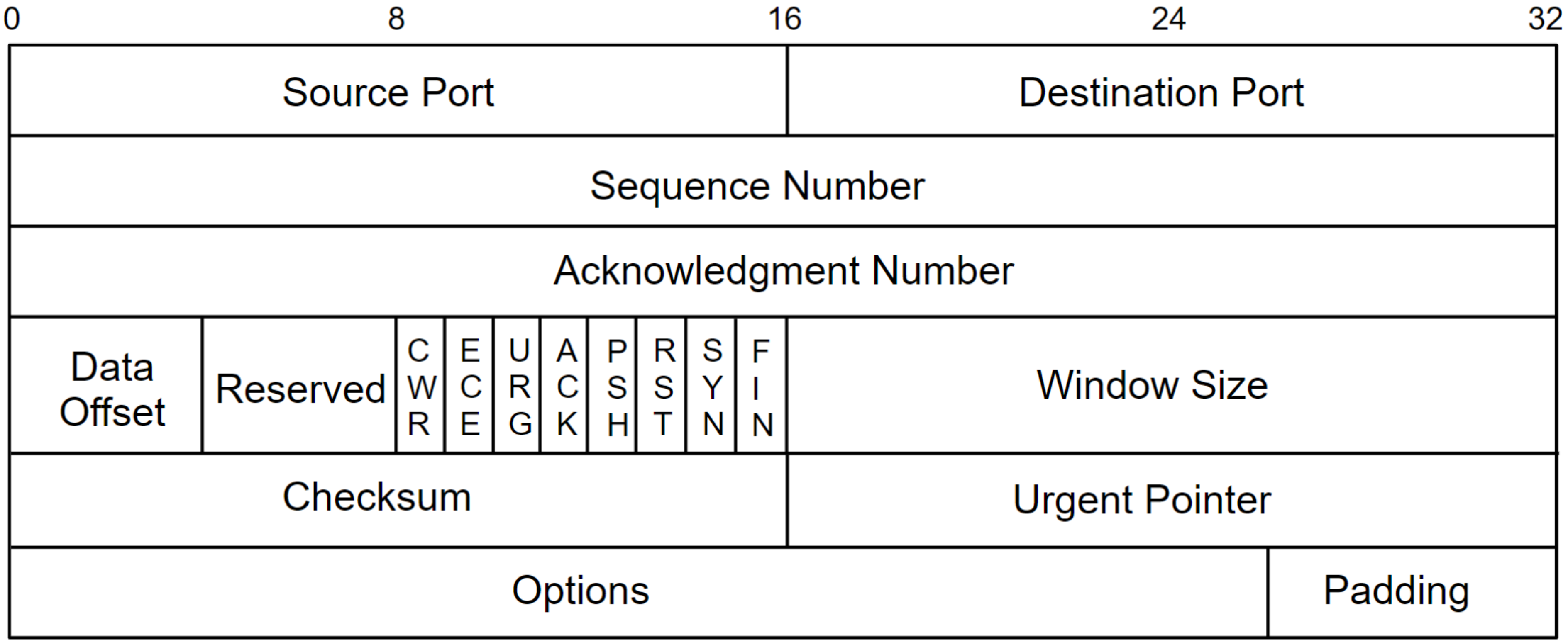
Transport Layer

TCP Sequence Numbers

A TCP connection is transmitting a **byte stream**

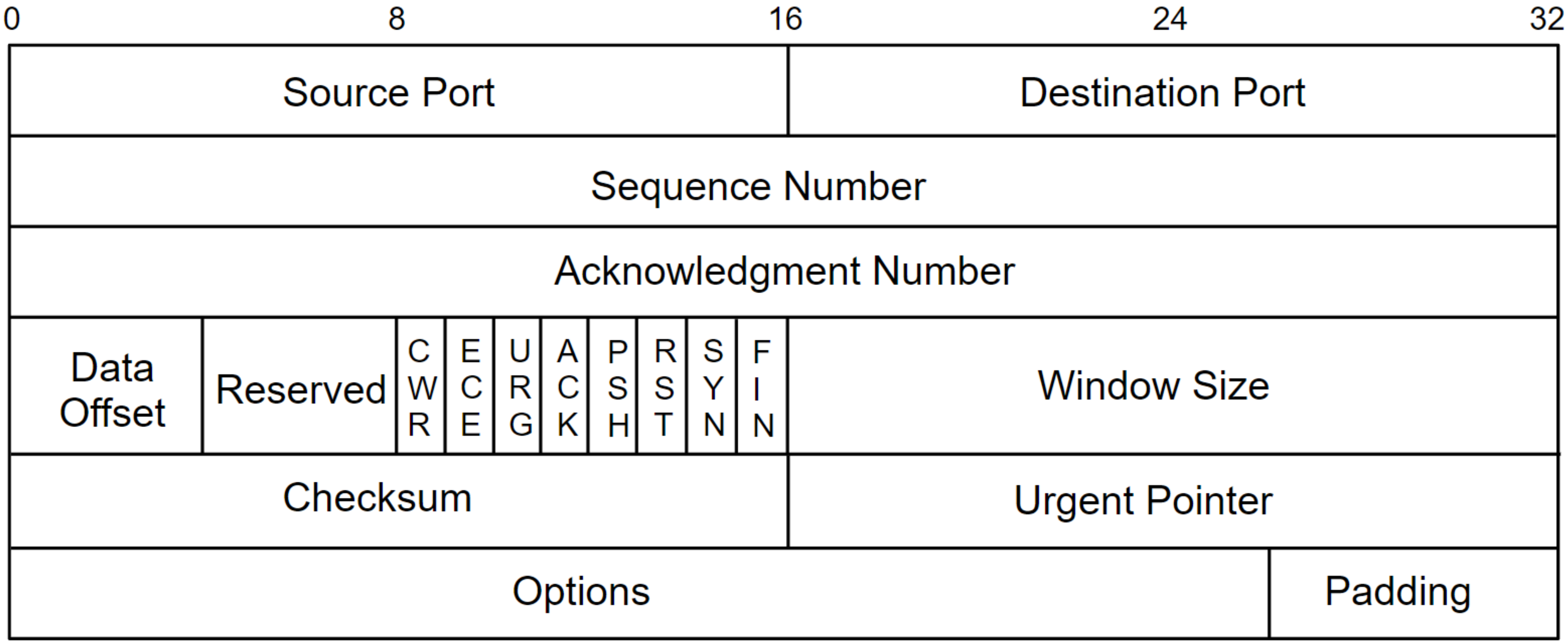


TCP Header


0
8
16
24
32
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Reserved
CWR
ECE
URG
ACK
PSH
RST
SYN
FIN
Window Size
Checksum
Urgent Pointer
Options
Padding

TCP Header

(20-60 bytes of data)



Transport Layer

TCP Header

(20-60 bytes of data)

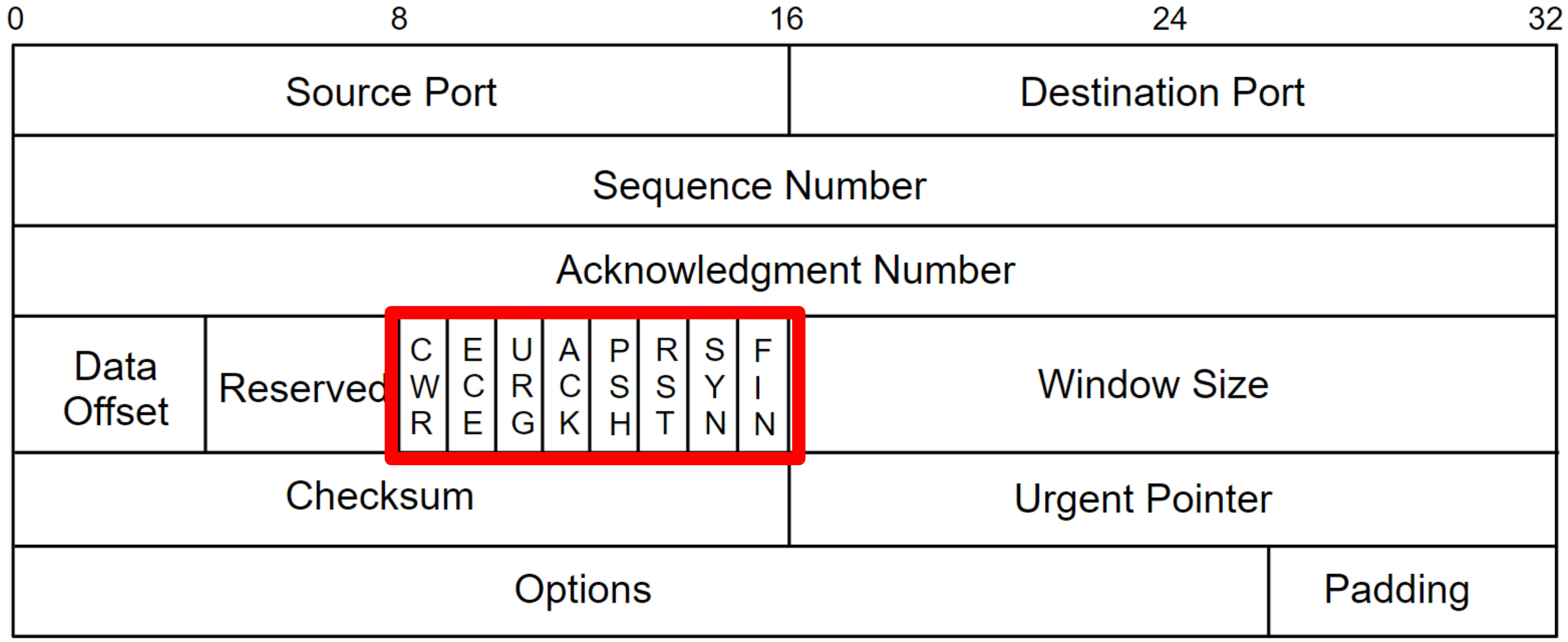
<table>
<tr><th colspan="10">0 8 16</th><th colspan="2">24 32</th></tr>
<tr><td colspan="10">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="12">Sequence Number</td></tr>
<tr><td colspan="12">Acknowledgment Number</td></tr>
<tr><td>Data Offset</td><td>Reserved</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="2">Window Size</td></tr>
<tr><td colspan="10">Checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="11">Options</td><td>Padding</td></tr>
</table>

Count by bytes, not segment

Transport Layer

TCP Header

(20-60 bytes of data)

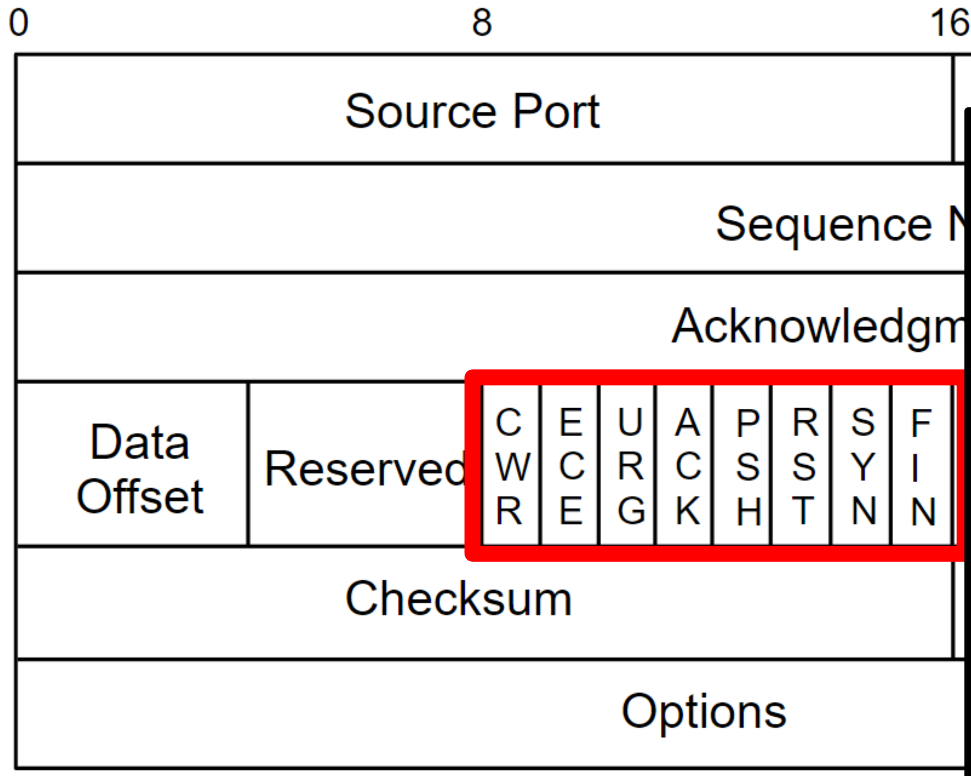


Count by bytes, not segment

Transport Layer

TCP Header

(20-60 bytes of data)



Acronym	Name	Meaning
SYN	Synchronization	Used to create a TCP connection
ACK	Acknowledgment	Used to acknowledge the reception of data or synchronization packets
PSH	Push	Instruct the network stacks to bypass buffering
URG	Urgent	Indicates out-of-band data that must be processed by the network stacks before normal data
FIN	Finish	Gracefully terminate the TCP connection
RST	Reset	Immediately terminate the connection and drop any in-transit data

CWR, ECE – Used for congestion control

TCP Header

(20-60 bytes of data)

0 8 16 24 32

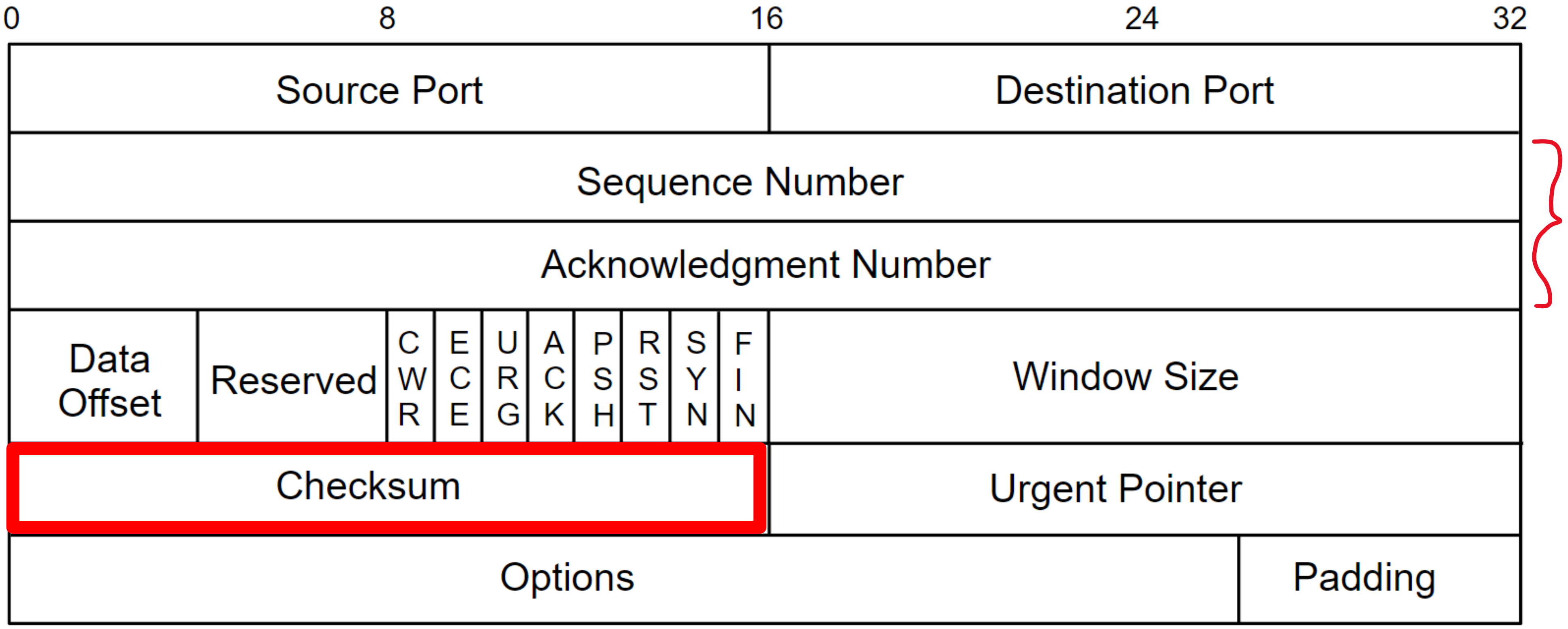
<table>
<tr><td colspan="10">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="12">Sequence Number</td></tr>
<tr><td colspan="12">Acknowledgment Number</td></tr>
<tr><td>Data Offset</td><td>Reserved</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="2">Window Size</td></tr>
<tr><td colspan="10">Checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="11">Options</td><td>Padding</td></tr>
</table>

Count by bytes, not segment

How many bytes the receiver is willing to accept

TCP Header

(20-60 bytes of data)



Count by bytes, not segment

Used to detect bit errors

TCP Header

TCP Segment Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset / Res / Flags		Window Size	
128	Header and Data Checksum		Urgent Pointer	
160...	Options			

UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

TCP Handshake

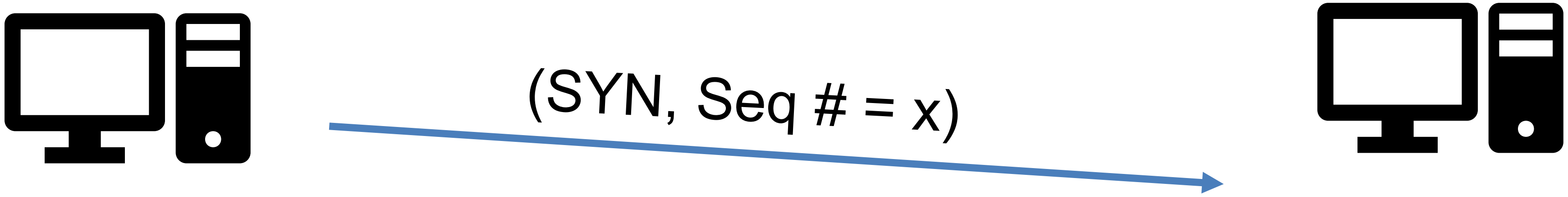
When a process wants to establish a TCP connection with another host, a **TCP handshake** must occur

TCP Handshake

When a process wants to establish a TCP connection with another host, a **TCP handshake** must occur

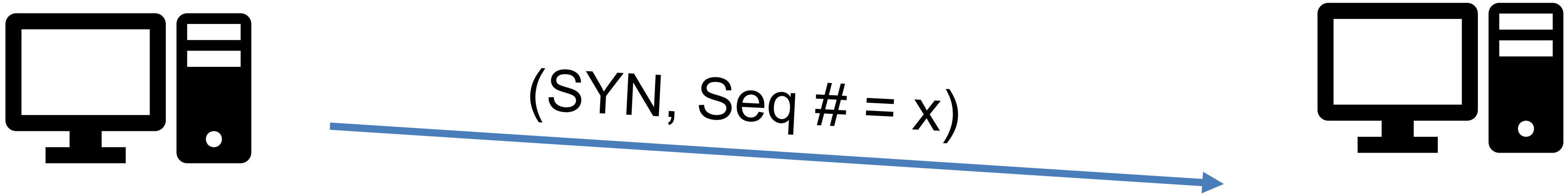
TCP Handshake

When a process wants to establish a TCP connection with another host, a **TCP handshake** must occur



TCP Handshake

When a process wants to establish a TCP connection with another host, a **TCP handshake** must occur



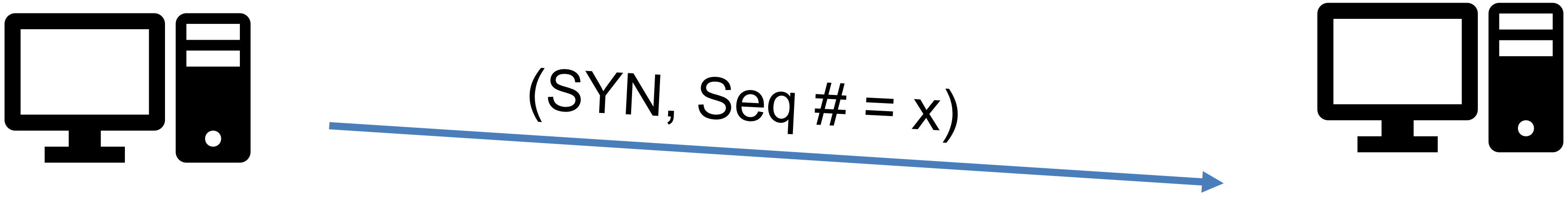
0 8 16 24 32

Source Port										Destination Port
Sequence Number										
Acknowledgment Number										
Data Offset	Reserved	CWR	ECE	URG	ACK	PSH	RST	SYN 1	FIN	Window Size
Checksum										Urgent Pointer
Options										Padding

Transport Layer

TCP Handshake

When a process wants to establish a TCP connection with another host, a **TCP handshake** must occur

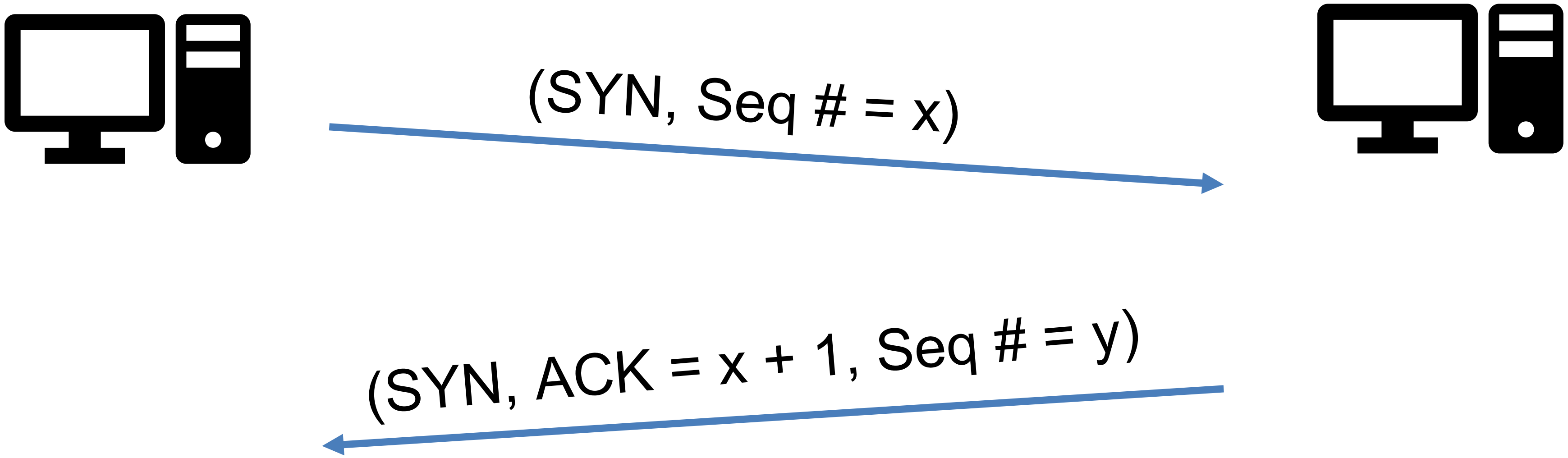


When establishing the connection, enable the **SYN** flag (set to 1)

Set an initial sequence number

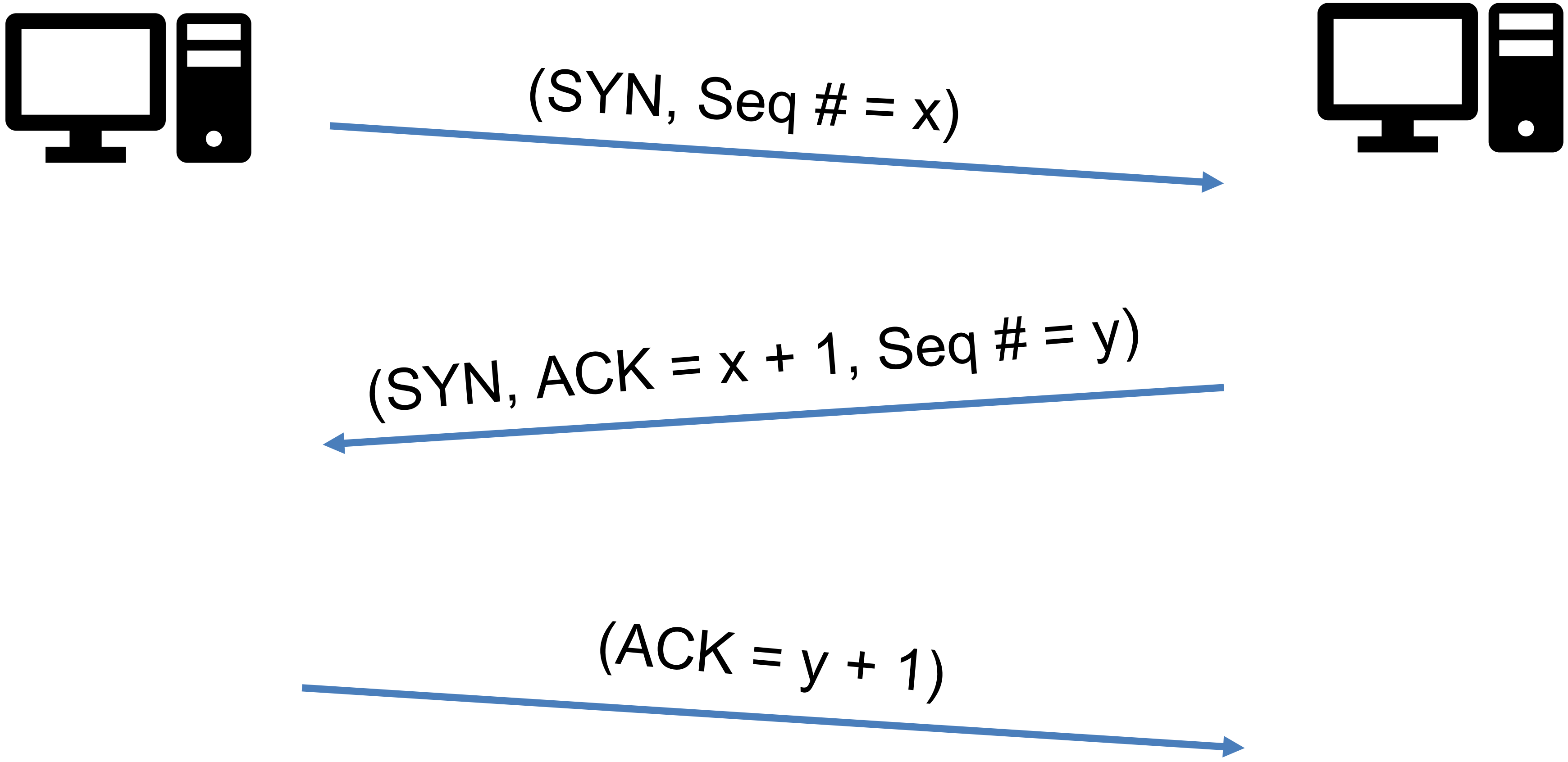
TCP Handshake

When a process wants to establish a TCP connection with another host, a **TCP handshake** must occur



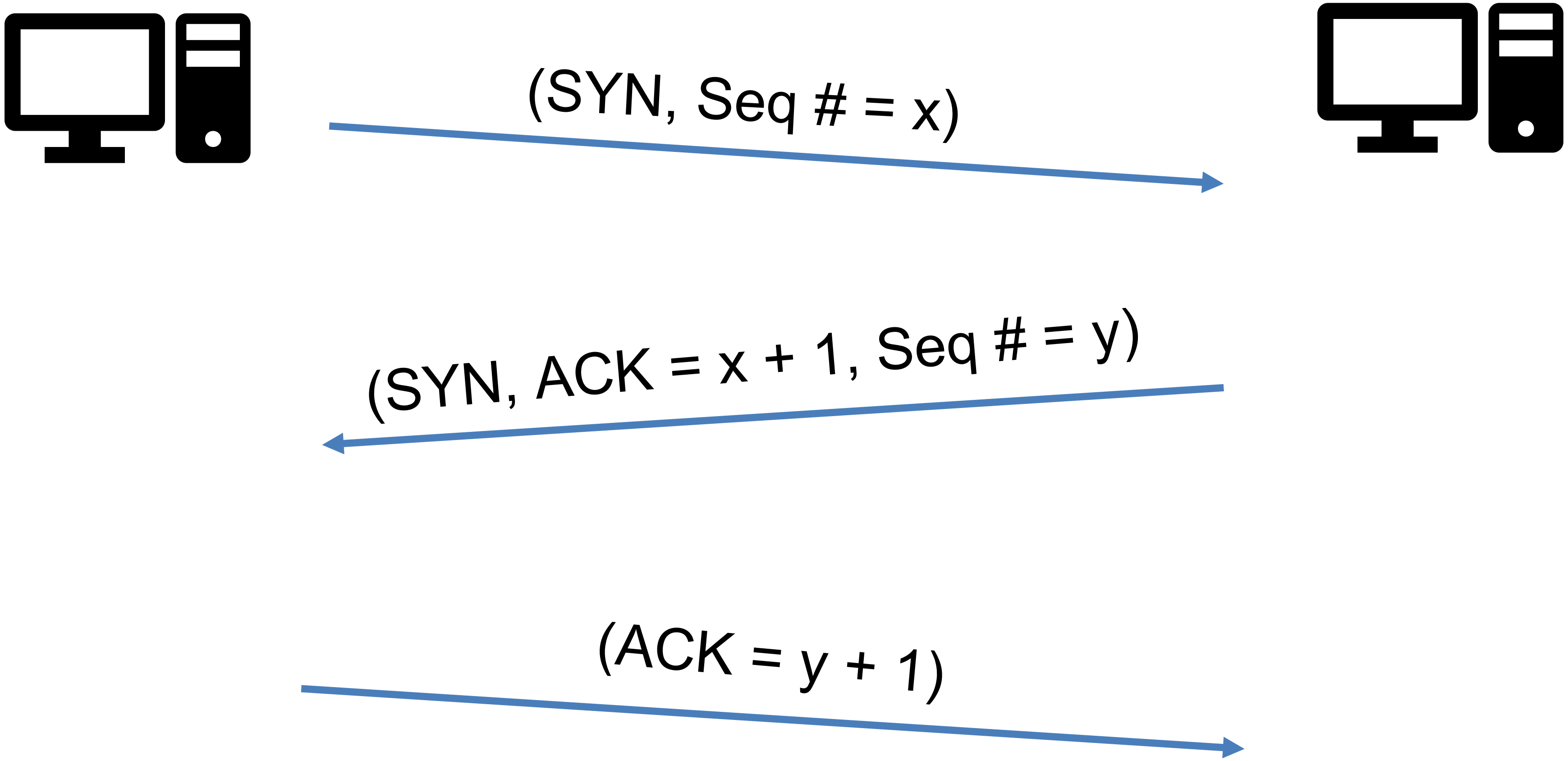
TCP Handshake

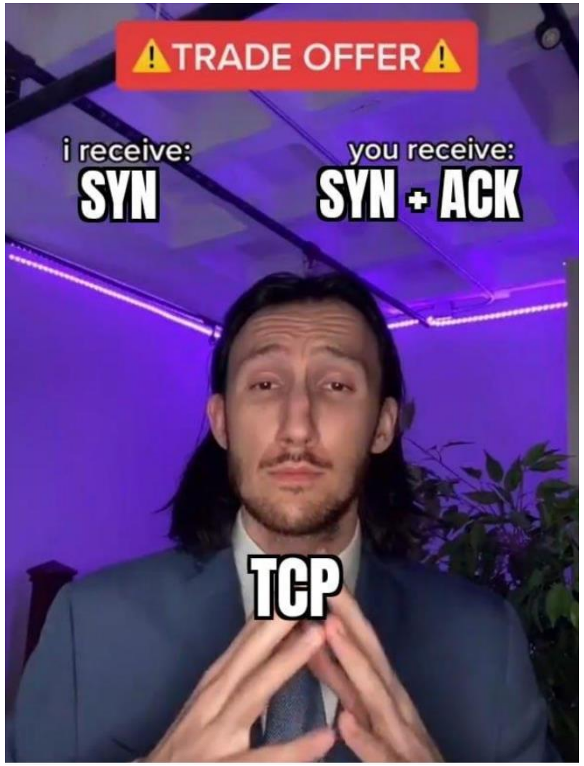
When a process wants to establish a TCP connection with another host, a **TCP handshake** must occur



TCP Handshake

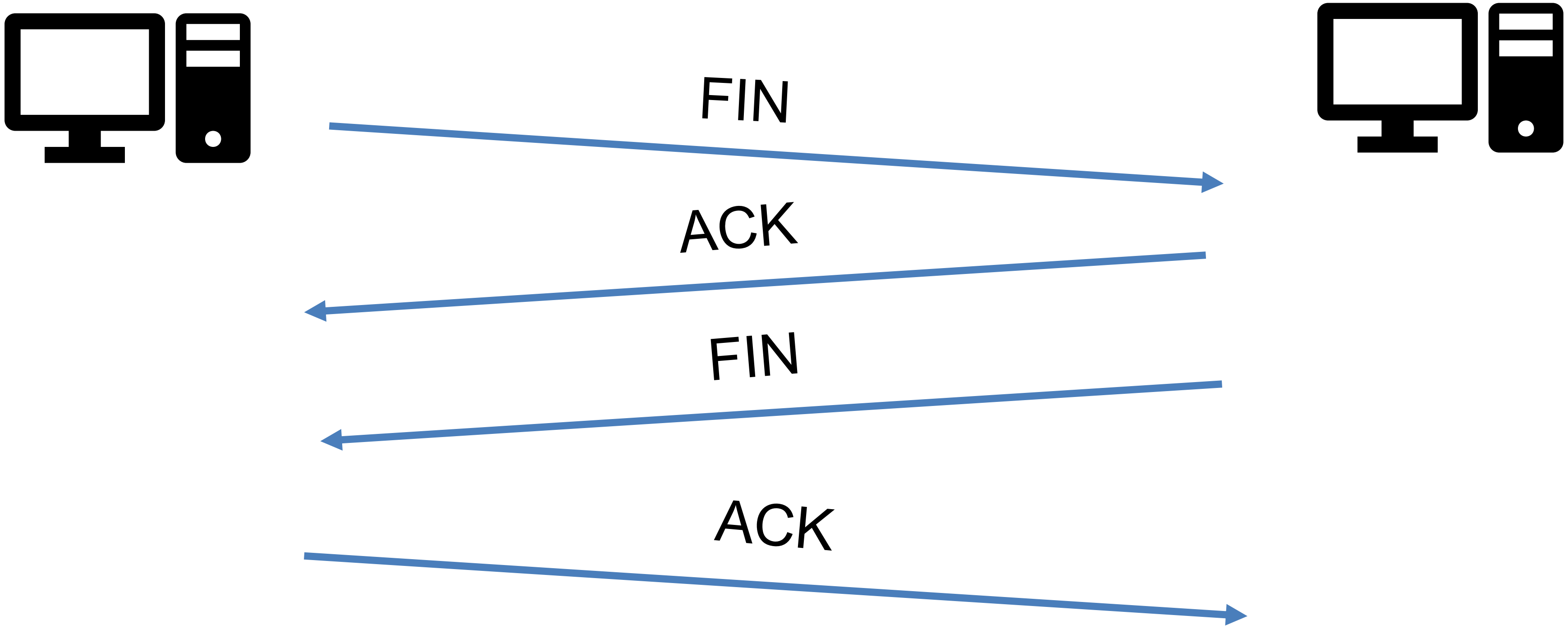
When a process wants to establish a TCP connection with another host, a **TCP handshake** must occur



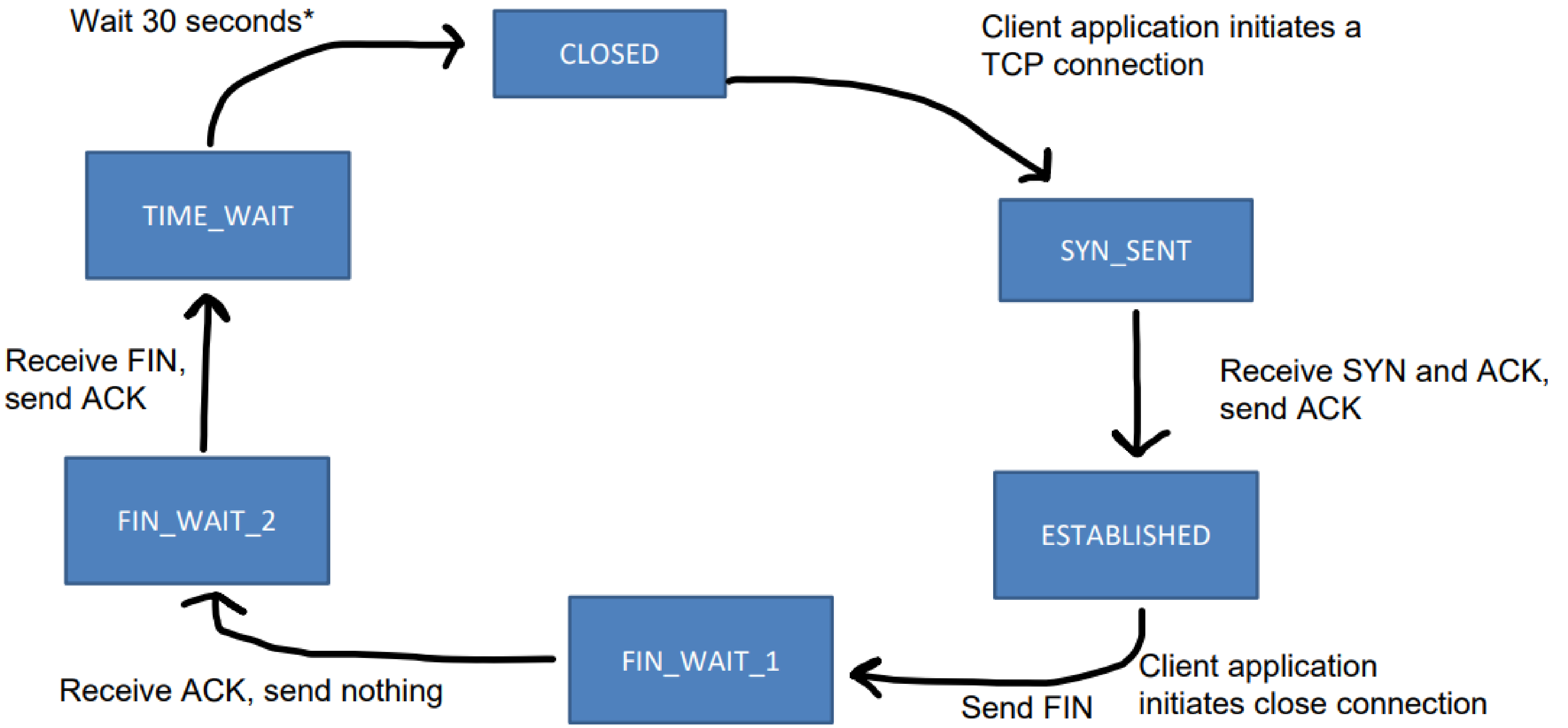


TCP Goodbye

When a process wants to terminate a TCP connection with another host, it sends a **FIN** packet



TCP States


Wait 30 seconds*
CLOSED
Client application initiates a TCP connection
TIME_WAIT
SYN_SENT
Receive FIN, send ACK
Receive SYN and ACK, send ACK
FIN_WAIT_2
ESTABLISHED
FIN_WAIT_1
Client application initiates close connection
Receive ACK, send nothing
Send FIN

TCP States

What if we receive a packet that has an invalid port number?

TCP Packet → send a TCP segment back with the **RST** flag on
UDP Packet → Send an **ICMP** datagram (network layer thing)

TCP / UDP in Wireshark

RFCs

RFCs (Request for Comments) documents and describes the details and standards of how internet protocols (such as HTTP, TCP, UDP) should work

TCP- **RFC 793**

TRANSMISSION CONTROL PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

September 1981

UDP- **RFC 768**

User Datagram Protocol

Introduction

This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

Format

```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|
|          data octets ...
+---------------- ...
```

User Datagram Header Format

DNS- **RFC 1035**

DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION

1. STATUS OF THIS MEMO

This RFC describes the details of the domain system and protocol, and assumes that the reader is familiar with the concepts discussed in a companion RFC, "Domain Names - Concepts and Facilities" [RFC-1034].

The domain system is a mixture of functions and data types which are an official protocol and functions and data types which are still experimental. Since the domain system is intentionally extensible, new data types and experimental behavior should always be expected in parts of the system beyond the official protocol. The official protocol parts include standard queries, responses and the Internet class RR data formats (e.g., host addresses). Since the previous RFC set, several definitions have changed, so some previous definitions are obsolete.

RFCs

A Standard for the Transmission of IP Datagrams on Avian Carriers

Status of this Memo

This memo describes an experimental method for the encapsulation of IP datagrams in avian carriers. This specification is primarily useful in Metropolitan Area Networks. This is an experimental, not recommended standard. Distribution of this memo is unlimited.

Overview and Rational

Avian carriers can provide high delay, low throughput, and low altitude service. The connection topology is limited to a single point-to-point path for each carrier, used with standard carriers, but many carriers can be used without significant interference with each other, outside of early spring. This is because of the 3D ether space available to the carriers, in contrast to the 1D ether used by IEEE802.3. The carriers have an intrinsic collision avoidance system, which increases availability. Unlike some network technologies, such as packet radio, communication is not limited to line-of-sight distance. Connection oriented service is available in some cities, usually based upon a central hub topology.