



University of Pittsburgh

# ECE 1150: Computer Networks

## The Transport Layer

**Mai Abdelhakim, PhD**

ECE Department

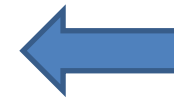
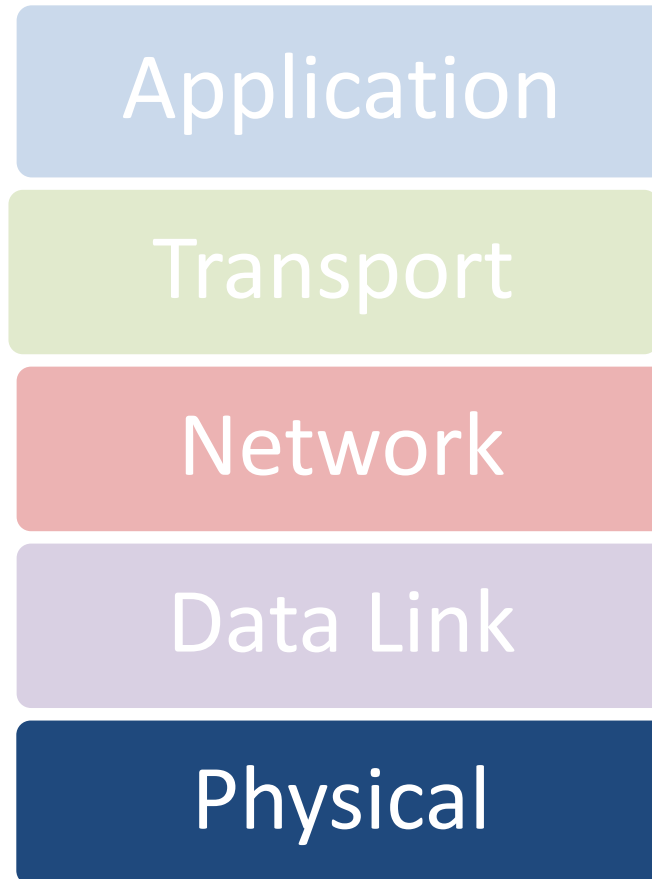
Swanson School of Engineering

University of Pittsburgh



# This Unit

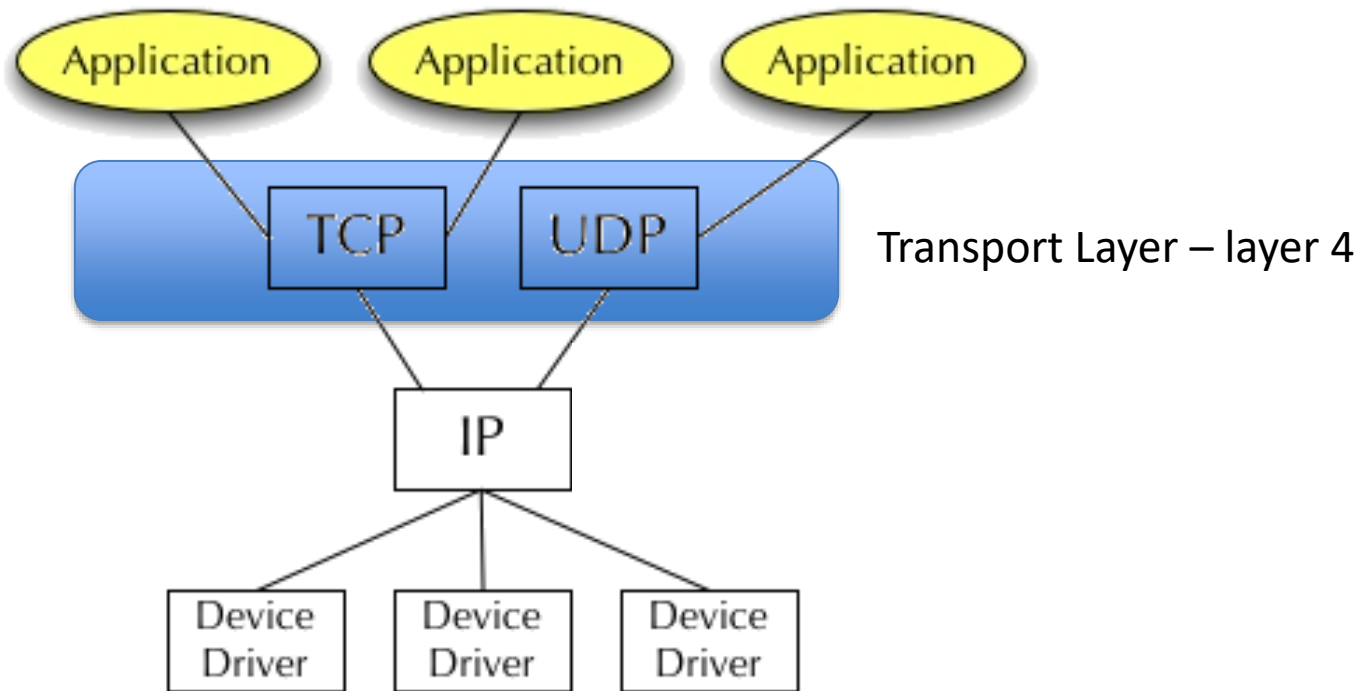
- Transport layer



We are here  
(Layer 4)

# Transport Layer

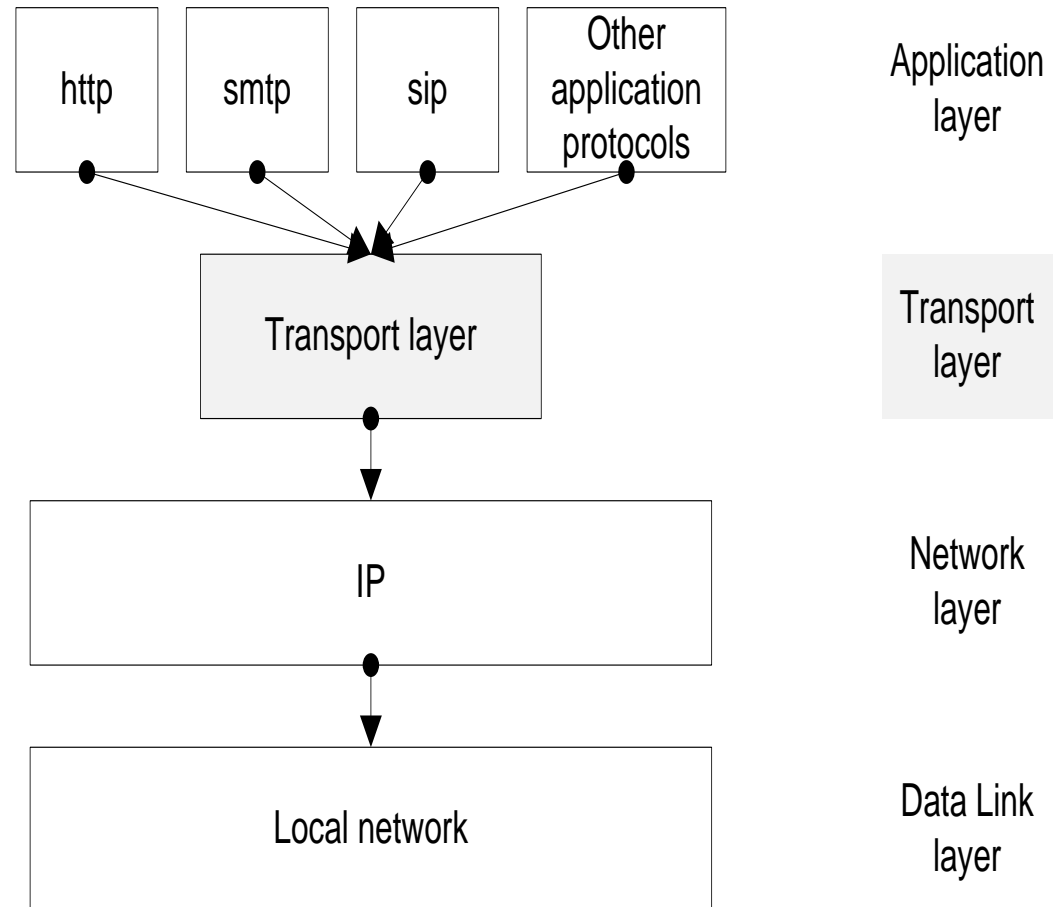
- Two types:
  - TCP: Transport control protocol (reliable)
  - UDP: User Datagram protocol (easy)



Schematic of TCP/IP Operation

# TCP Functions

- Therefore TCP has to perform many tasks
  1. Segmentation
  2. Reliability
  3. Flow-control
  4. Multiplexing for applications
  5. Connection establishment



# TCP Functions - Segmentation

- TCP allows transfer arbitrarily large data blocks
- However, there is a **maximum packet size** in network layer
  - 65,536 bytes
- What happens if the application wants to send a file of size 5 Mega bytes?

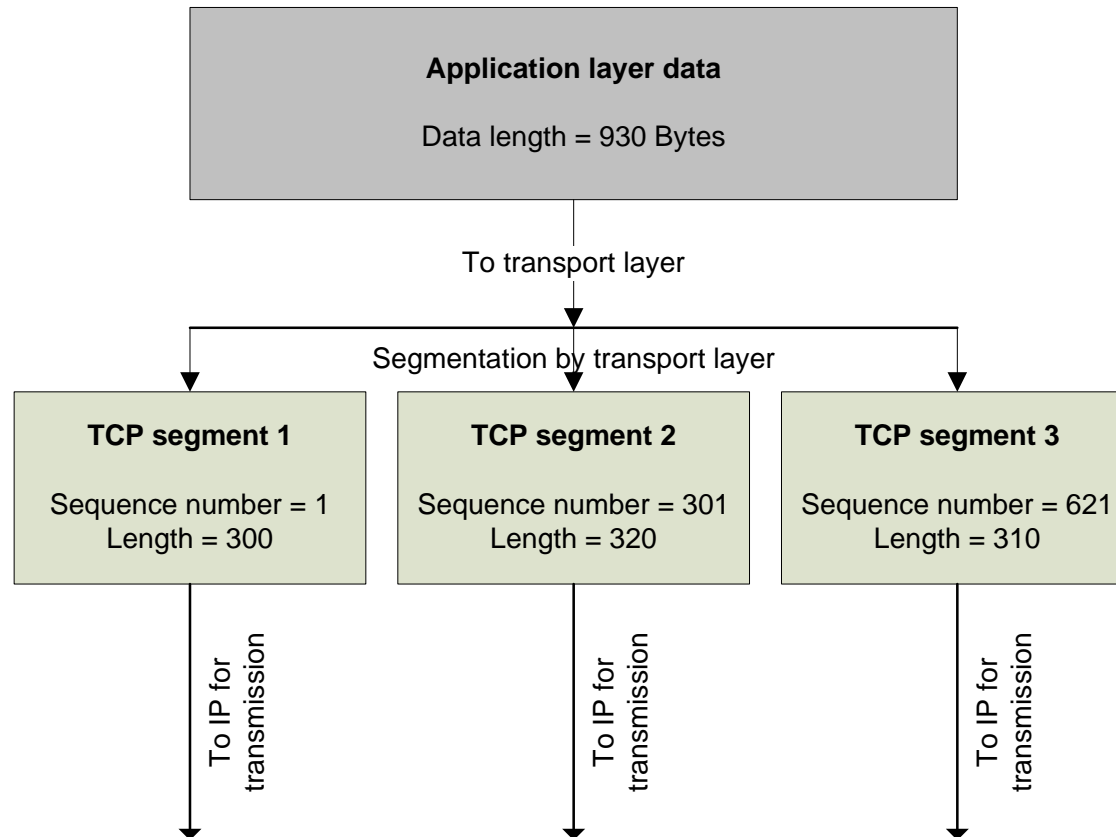
# TCP Functions - Segmentation

TCP does **segmentation**

- At the sender, **break files into smaller blocks** (called segments or datagrams)
- At the receiving end, re-assemble these blocks into the file
- A **sequence number** is assigned to each datagram
  - Sequence numbers help **receiver order datagrams** even when received out of order
  - Datagrams may also get **duplicated**
    - Sequence numbers help **identify these duplicates**

# Sequence Numbers: Number of First Byte in Sequence

Assume we start with sequence number 1



Sequence number of a TCP segment

=

sequence number of the previous segment + length of previous segment

# TCP functions - Segmentation

- Advantages of segmentation
  - **Errors are less** likely in smaller segments
  - **Easier for routers** to hold segments in memory
    - Reduce complexities in packet handling
- Disadvantages
  - **More computations** in sending and receiving



# TCP Functions - Reliability

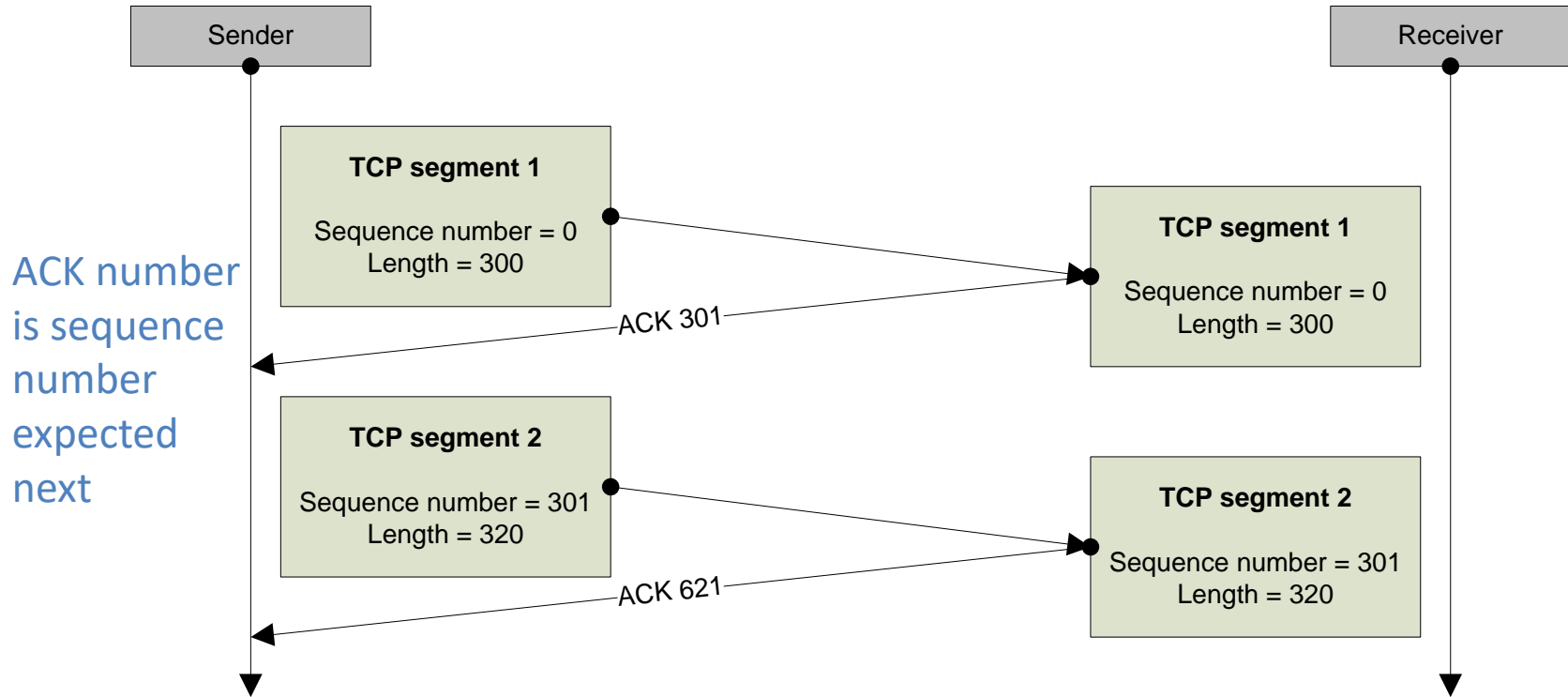
- Datagrams may get damaged during transmission
- **TCP adds a checksum** to detect errors. This checksum is not as robust as CRC.

# TCP Functions - Reliability

- **Reliable end-to-end** communication service
- TCP **recovers from network damage** to data
- Basic mechanisms **similar to error control at data link layer**, but now **applied end-to-end**
  - Receiver sends a positive **acknowledgment (ACK)** if all goes well
  - If the **ACK is not received** within a **timeout** interval, the sender retransmits the data
- Two techniques: **Stop-and-wait** & **sliding window**
- Also used for **flow control**

# Simple Flow Control Mechanism

## Stop - and - Wait

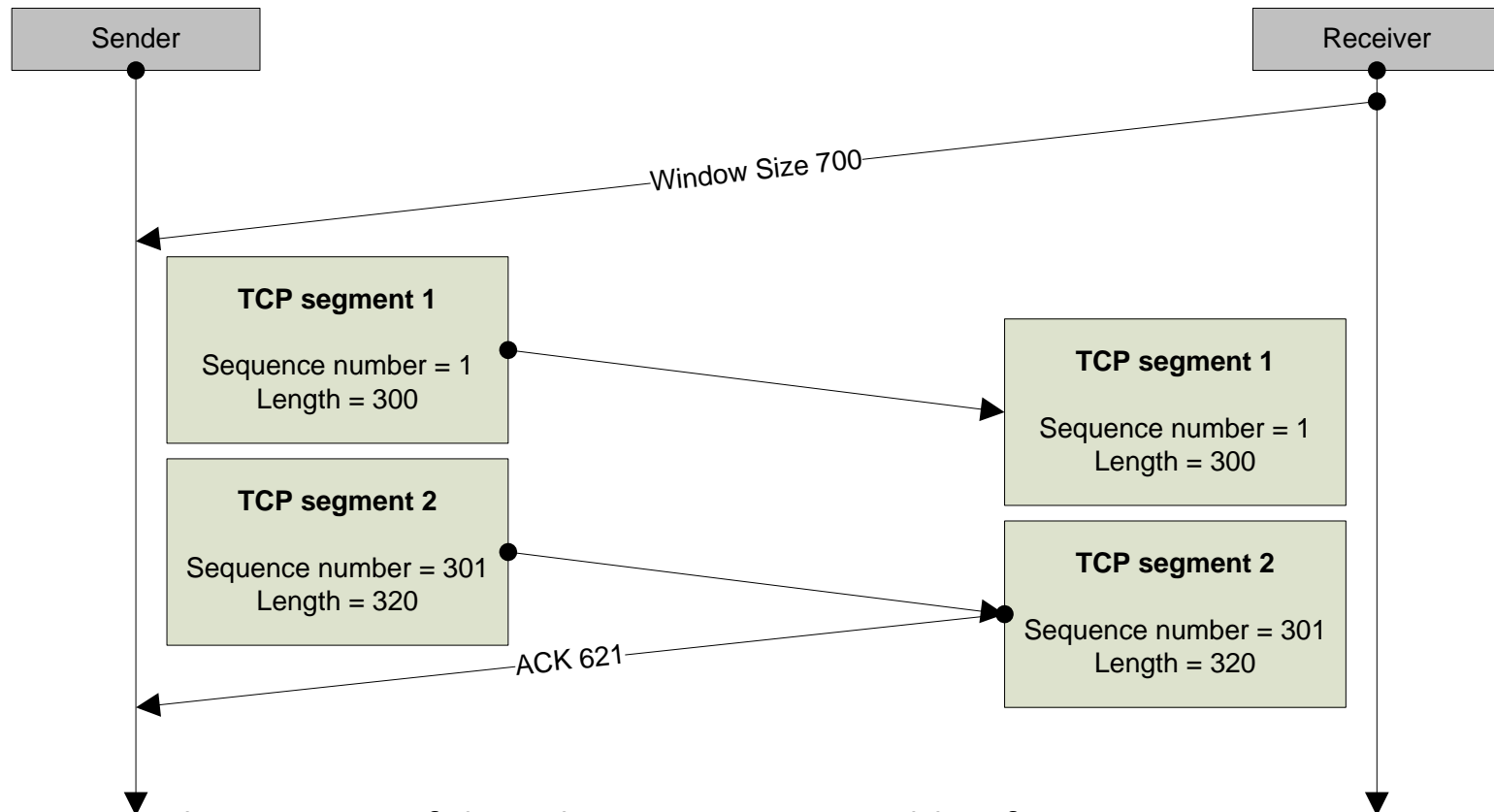


- Flow control mechanism shown here is called stop-and-wait
  - Sender waits for ACK before sending next datagram
- Very slow

# Flow Control - Sliding window

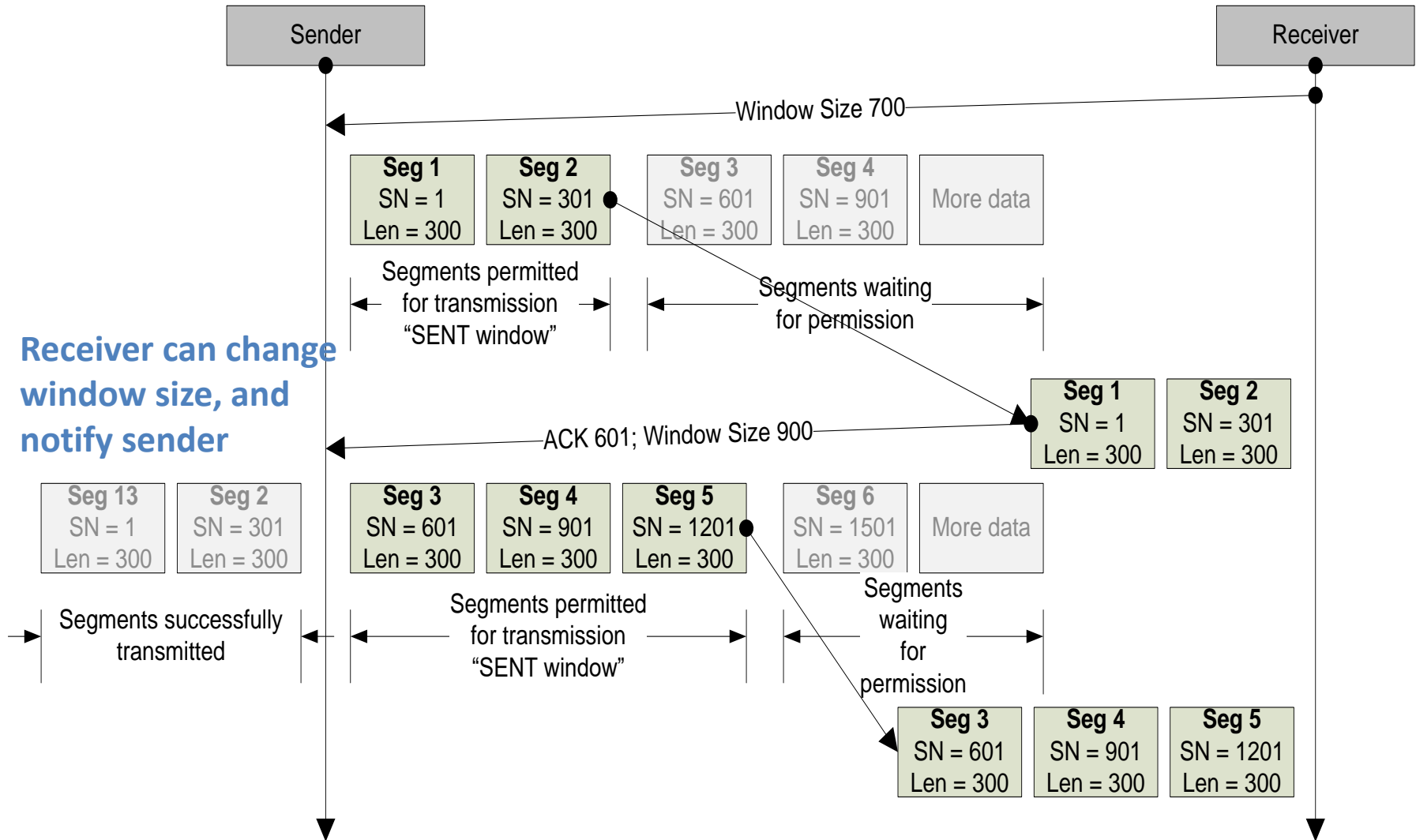
- Accomplished by specifying a “**window**” with every **ACK**
  - “**Window**” indicates **how many bytes** of data sender may transmit **before receiving any more ACK**
- Window slides as **receiver acknowledges** packets or modifies window size

# TCP Flow Control with Window Size



- Window size is the amount of data the receiver is capable of processing
- Receiver begins by announcing the window size
- ACK 621: means that all data up to byte 620 is received correctly

# Sliding Window

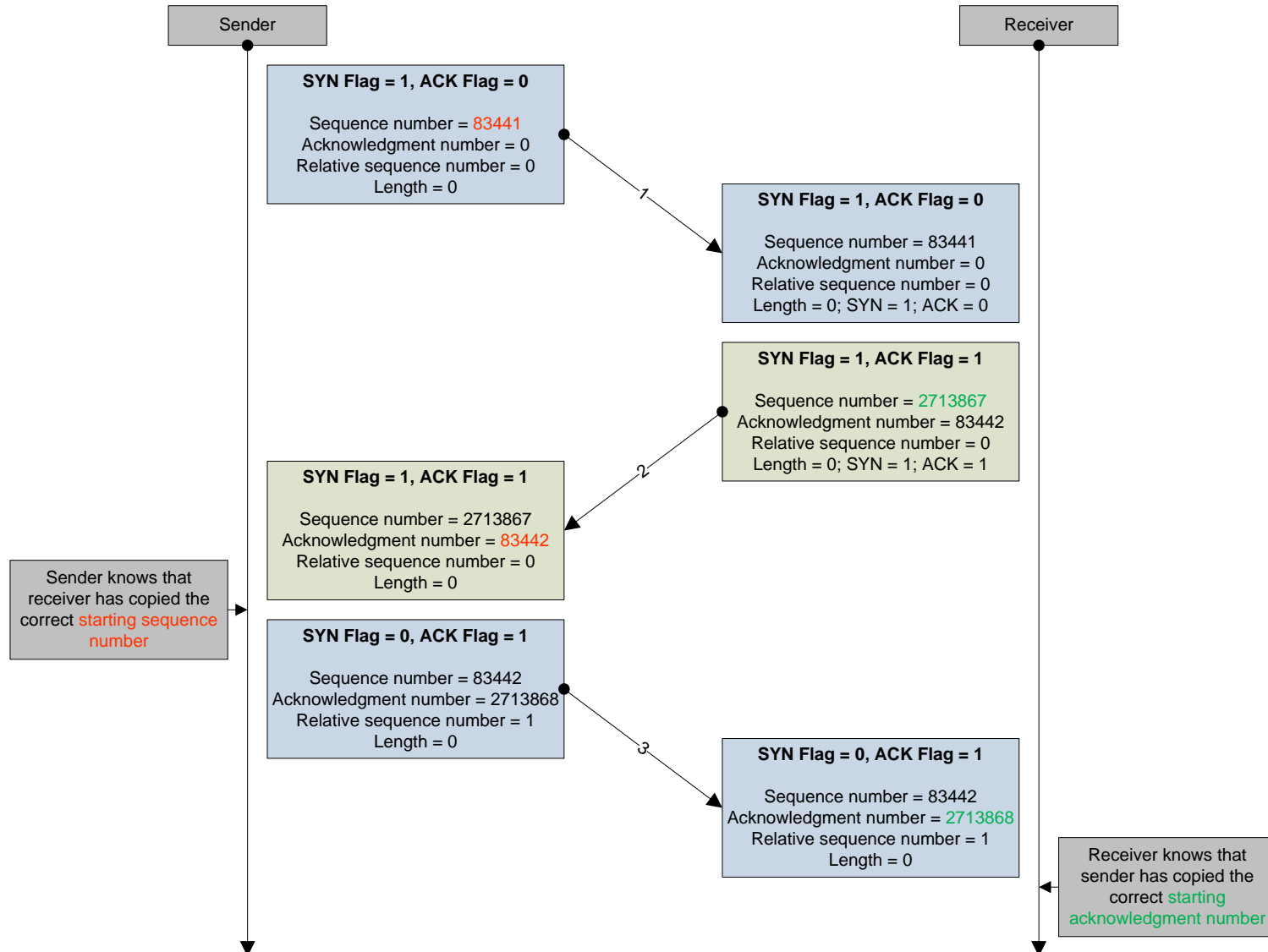


# TCP Functions – Connection Establishment

- Sequence numbers are core part of TCP
- It is not a good idea to reuse the same sequence numbers in succession
  - Creates problems in detecting duplicates
- Hence, before communication starts, sender and receiver negotiate the initial sequence numbers to use in TCP connection
  - Called 3-Way Handshake, which is made to establish a connection

# 3-Way Handshake

1. Sender generates Initial Sequence Number (ISN).  
E.g. ISN = 83441.
2. Receiver ACKs this reception by putting the (**ACK** = sender ISN + 1), and **inserts its own Rx ISN**.
3. When the sender receives the ACK, it also knows the ISN of the Rx, it then sends an ACK



Note: Acknowledgment number is increased by 1 if SYN or ACK flags are set



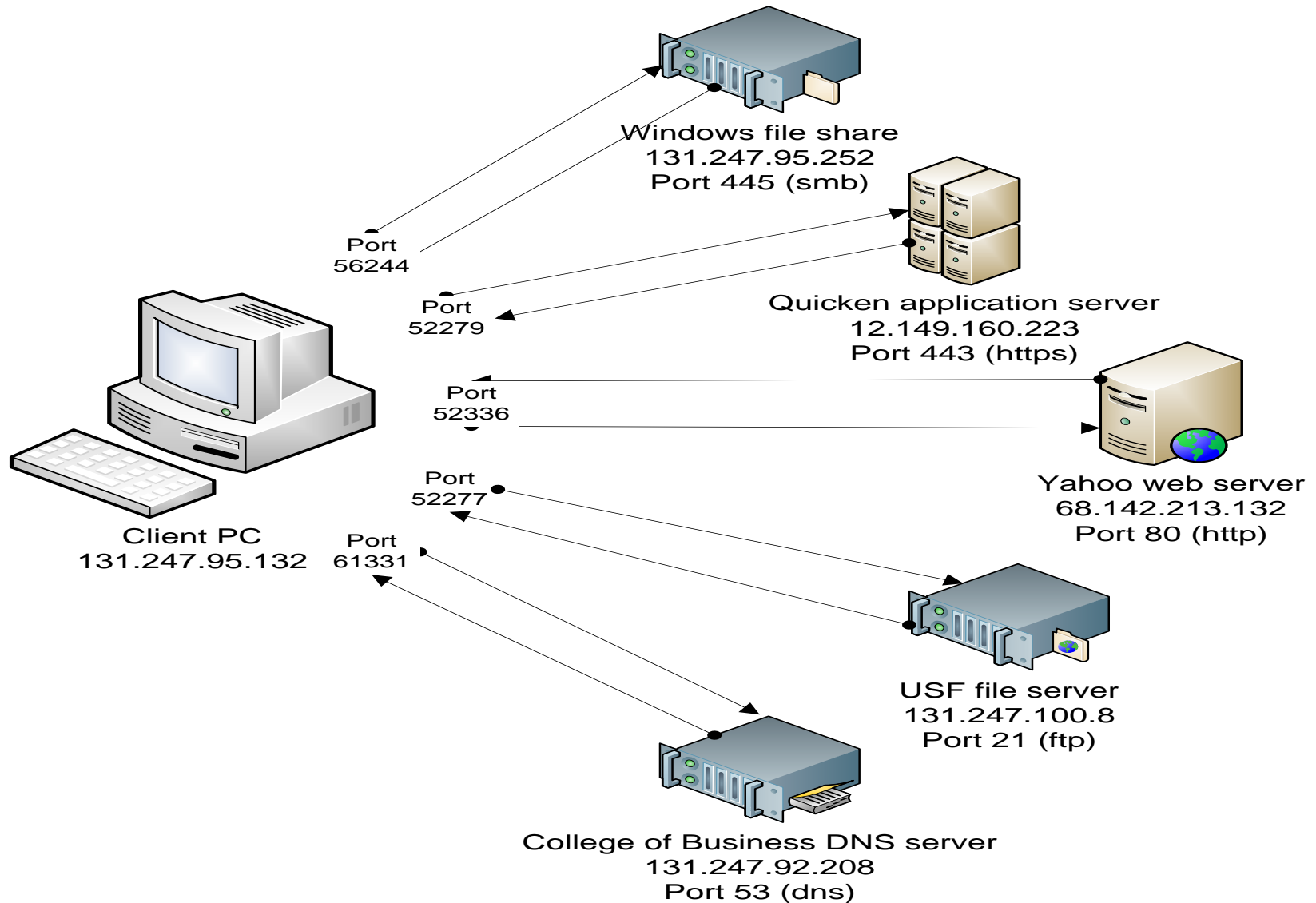
# Connection Establishment with 3-Way Handshake

- Three way handshake
  1. Sender generate a random initial sequence number (ISN) and sends it to destination
  2. Destination generates a random initial sequence number and sends it along with the acknowledgment to sender
  3. Sender acknowledge

# TCP Functions - Multiplexing

- Transport layer performs **application multiplexing**
  - Allowing multiple applications to use the same network interface card.
- This is accomplished by **assigning different port numbers** to different applications within a host
  - Each **application** that needs a network connection (each browser tab) is assigned a **port number**
  - Port number is 16 bits
- An IP address and port address together is called a **socket**

# Port addresses and multiplexing



# TCP Port Assignment

- On the **sender** side, the **operating system** assigns one of the free ports to an application that requires network connectivity
- How do you know what port to connect to on the receiver side?
  - On what port is the web server listening?
  - Standard ports
    - Assigned by IANA: Internet Assigned Numbers Authority
  - Common ports: <https://www.utilizewindows.com/list-of-common-network-port-numbers/>
    - 80 : http
    - 443: SSL (https)
    - 53: DNS
    - 23: Telnet

# Viewing Ports Usage With Netstat Command

- Netstat: Utility provided by OS to view port usage

```
C:\>netstat -n
```

Active Connections

Foreign address: IP addresses the local device is connected to

Proto	Local Address	Foreign Address	State
TCP	127.0.0.1:5354	127.0.0.1:46133	ESTABLISHED
TCP	127.0.0.1:46133	127.0.0.1:5354	ESTABLISHED
TCP	192.168.1.5:46122	192.168.1.153:445	ESTABLISHED
TCP	192.168.1.5:46149	74.125.67.138:443	CLOSE_WAIT
TCP	192.168.1.5:46151	74.125.45.104:80	CLOSE_WAIT
TCP	192.168.1.5:51117	131.247.80.160:143	ESTABLISHED
TCP	192.168.1.5:51124	131.247.80.160:143	ESTABLISHED
TCP	192.168.1.5:51292	131.247.16.141:22	ESTABLISHED
TCP	192.168.1.5:51369	131.247.80.29:445	SYN_SENT

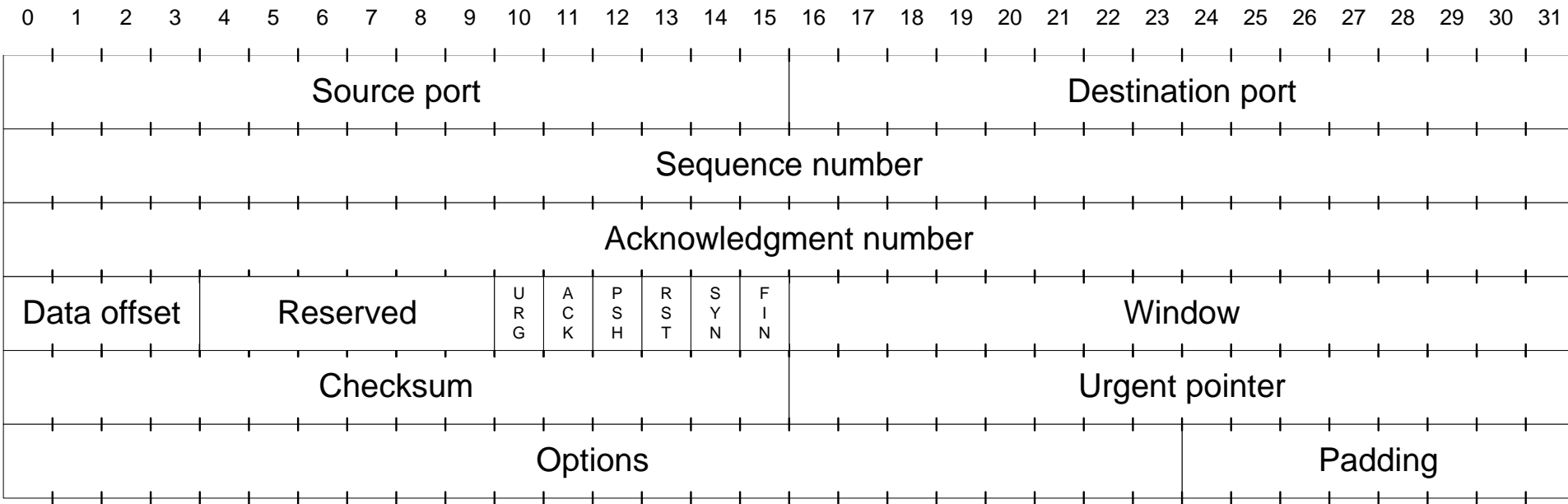
CLOSE\_WAIT: Wait to request of terminating connection

Ports on local device

Ports on remote servers

[https://en.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](https://en.wikipedia.org/wiki/Transmission_Control_Protocol)

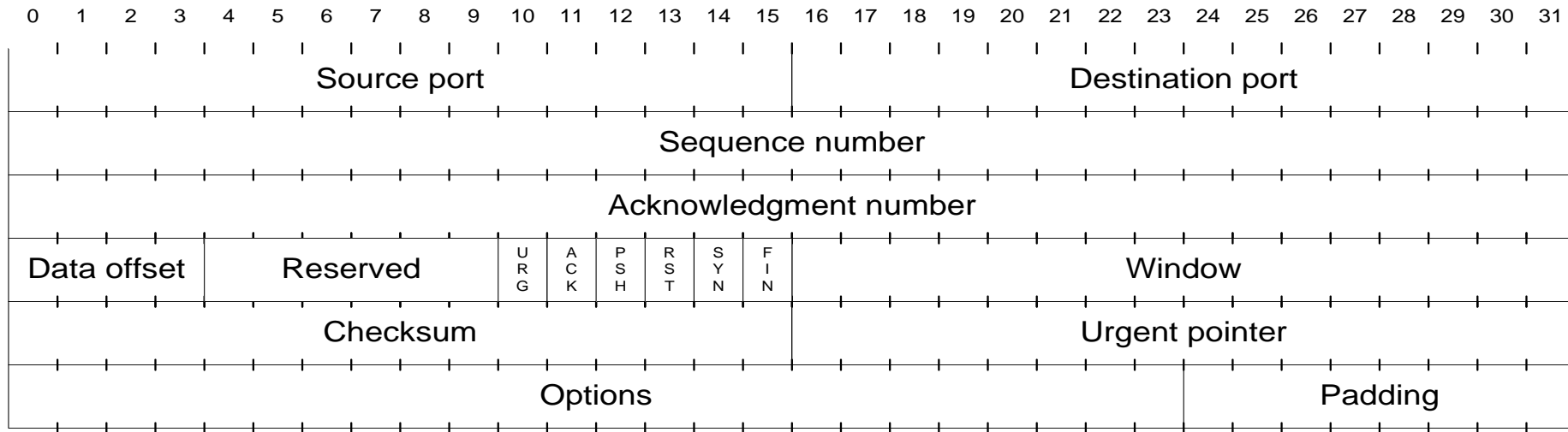
# TCP Header



Note: Each tick mark represents a bit position

# TCP Header Fields

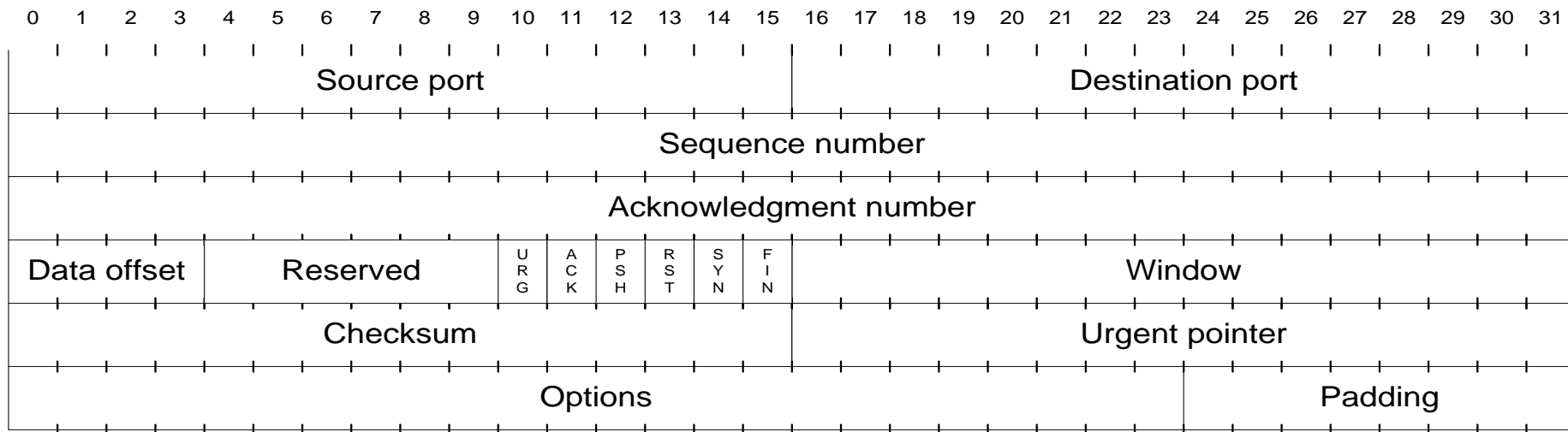
- Port addresses have 16 bits
  - $2^{16} = 65,536$  ports possible per host
  - If computing resources are available, a single computer running TCP can support 65,536 simultaneous network connections



Note: Each tick mark represents a bit position

# TCP Header Fields

- Sequence and acknowledgment numbers have 32 bits
  - Sequence number** is the sequence number of the **first data byte** in the datagram
  - Acknowledgments are cumulative
    - ACK 1079** implies that all data till byte number **1078** have been received correctly

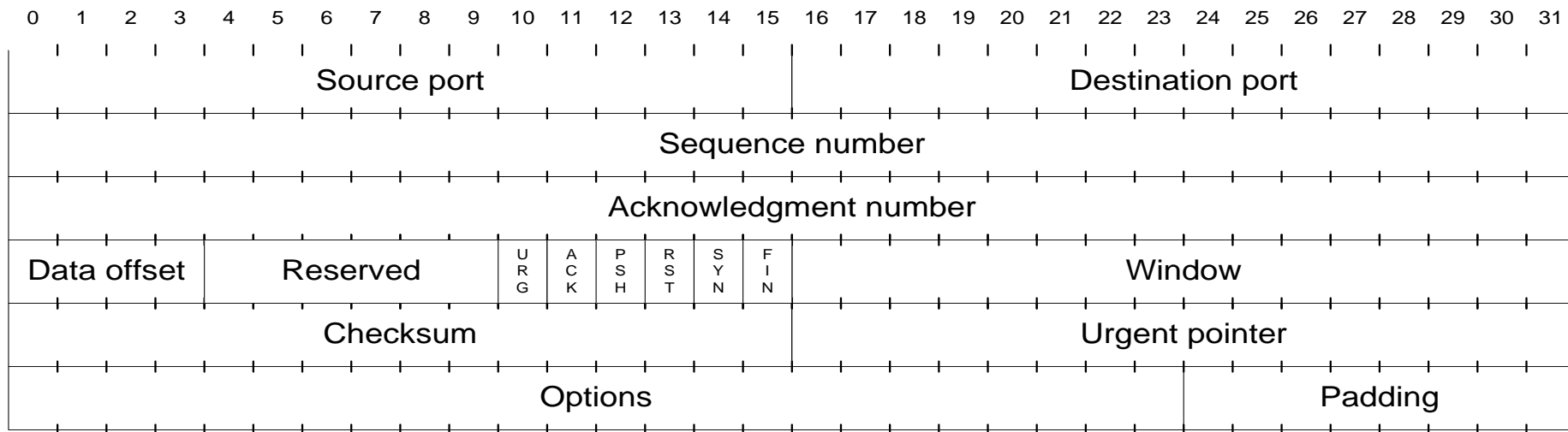


Note: Each tick mark represents a bit position



# TCP Header Fields

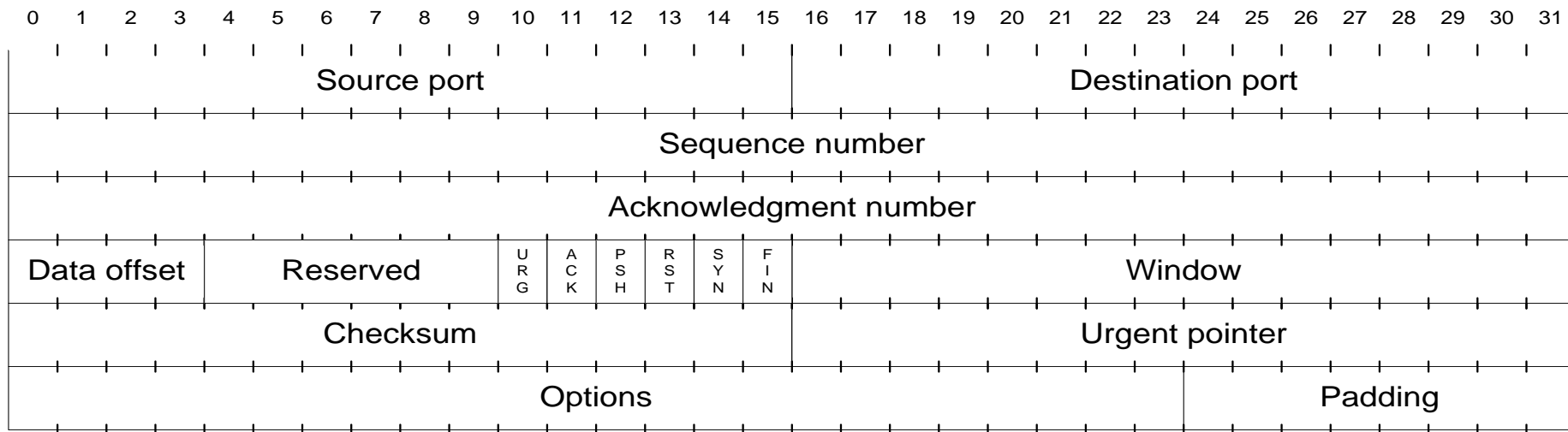
- Control fields
  - **ACK: 1** implies that the value of the acknowledgment field is meaningful
  - **SYN: 1** implies that the segment is trying to synchronize sequence numbers
  - **RST:** Reset the connection



Note: Each tick mark represents a bit position

# TCP Header Fields

- Window size: Number of data octets the sender of this information is willing to accept beyond the ACK number field
- Checksum: For error detection
- Urgent pointer: Indicates that data must be processed immediately (earlier applications).

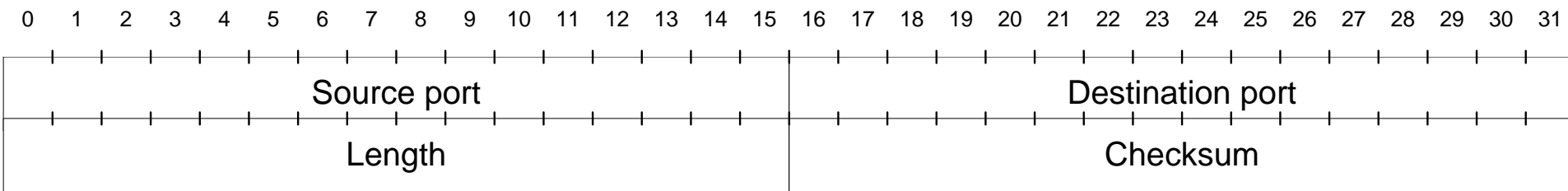


Note: Each tick mark represents a bit position

# User Datagram Protocol (UDP)

- Defined in [RFC 768](#) (1980)
- Many applications do not need TCP, such as
  - Applications sending [very small amounts](#) of data
  - Prefer [speed](#) to reliability (voice)
- In these cases, if we can avoid TCP, we [eliminate the overhead](#) of keeping track of sequence numbers, window sizes etc.
- With UDP: [No retransmissions, no sequence number validation, no window size defined.](#)

# UDP Header



Note: Each tick mark represents a bit position

Checksum: similar to TCP, but often not used in computer-intensive applications such as video streaming

Length: of the datagram

# Transport layer protocols

## TCP service:

- Connection-oriented: setup required between client and server processes (3-way handshake)
- Reliable transport between sending and receiving process
- Flow control: sender won't overwhelm receiver

## UDP service:

- **Unreliable data transfer** between sending and receiving process
- Faster transmission
- **Does not** provide: connection setup, reliability, flow control

# Summary

- Segmentation
  - Need sequence numbers
- Flow control achieved via stop-and-wait or sliding window
- Port numbers for application multiplexing
- TCP connection establishment through three-way handshake to agree on the initial sequence numbers
- UDP vs TCP