Lecture 20 – Security of Transport Layer

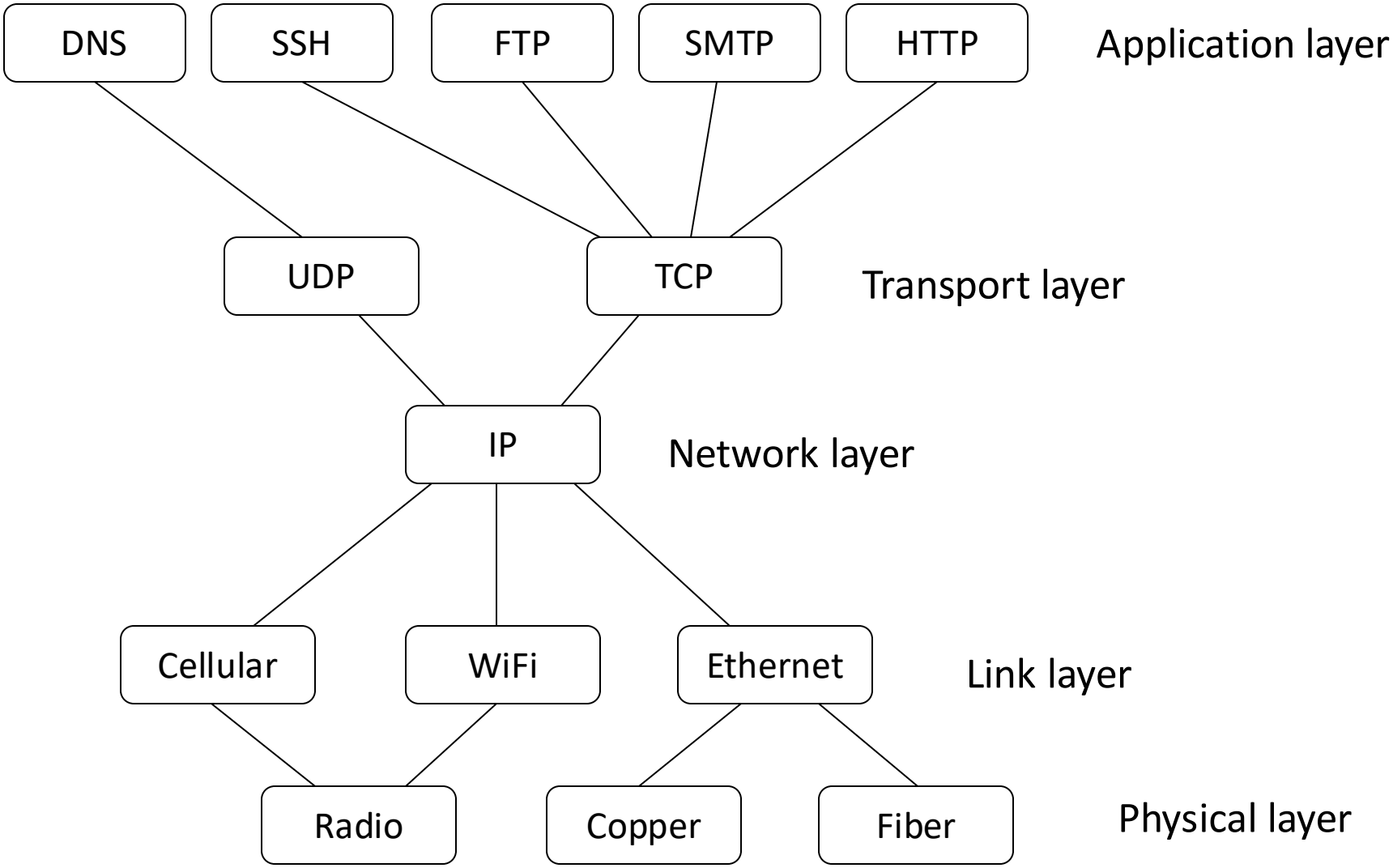
University of Illinois

ECE 422/CS 461

Learning Objectives

- Understand basic concepts of TCP
- Explore how TCP stacks up against the network security threat models
- Evaluate defenses against TCP hijacking attacks

Layering of Protocols



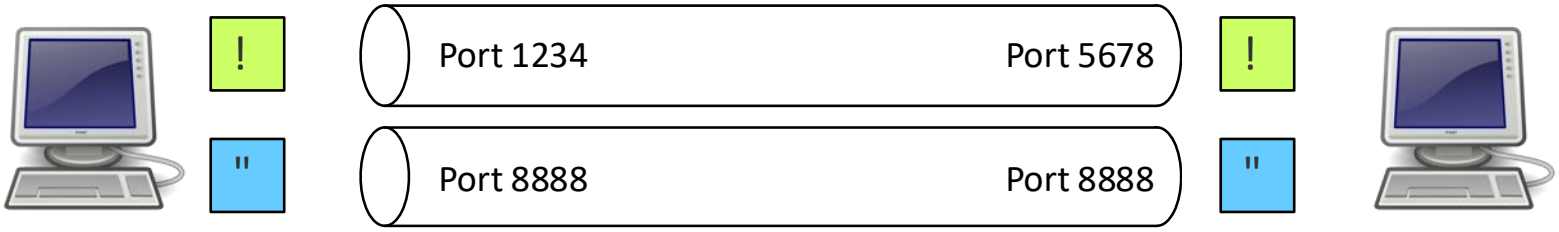
TCP vs. UDP vs. IP

- IP provides best-effort delivery between hosts
- TCP/UDP between processes (IP address, port)

- TCP (Transmission Control Protocol)
  - In-order delivery, reliable delivery, …
- UDP (User Datagram Protocol)
  - None of the above, security is similar to IP
- We will focus on TCP

Transmission Control Protocol

- Each process is identified by a *port number*
- TCP connection established between port *A* on host *X* to port *B* on host *Y*
  - Ports are 1–65535 (16 bits)



TCP Port Numbers

- Some destination port numbers used for specific applications by convention

Port	Application
80	HTTP (Web)
443	HTTPS (Web)
25	SMTP (mail)
22	SSH (secure shell)
514	RSH (remote shell)

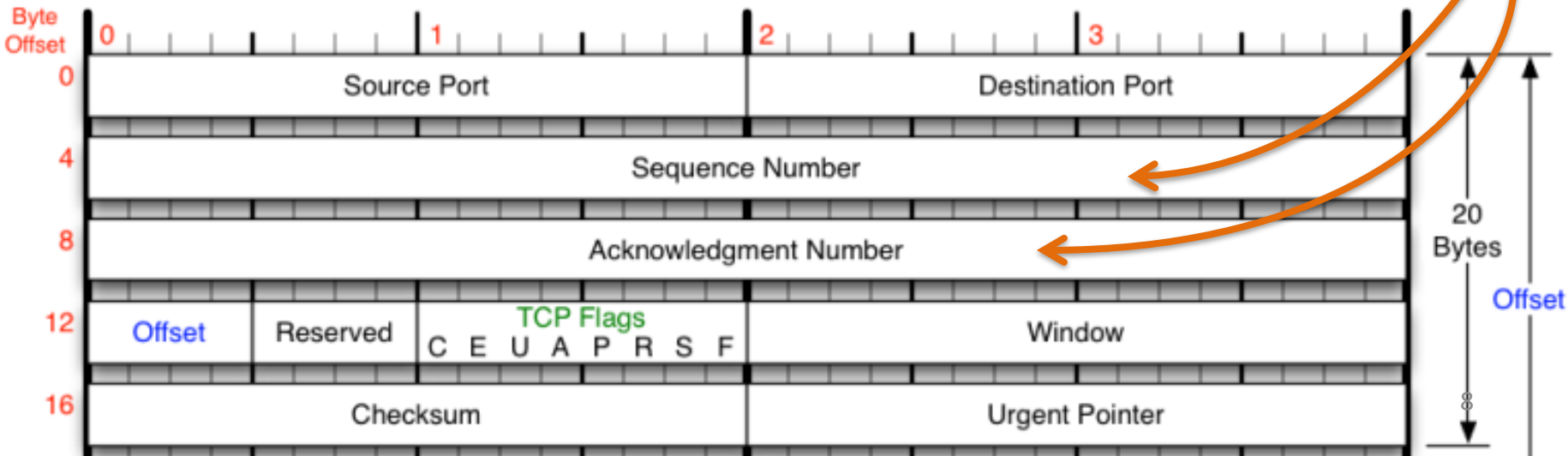
Transmission Control Protocol

- Have: a network where packets may be dropped, re-ordered, duplicated
- Want to provide: abstraction of a stream of bytes delivered **reliably** and **in-order**
  - Two logical data streams, one in each direction

- Main idea:
  - Sequence number for in-order delivery
  - Ack + retransmission for reliable delivery

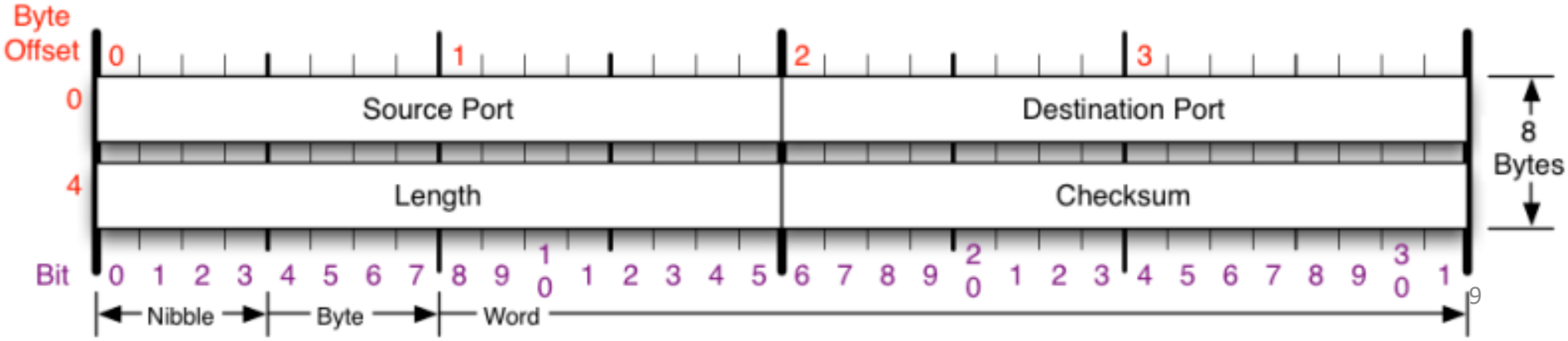
Transmission Control Protocol

- 16-bit port numbers
- Sequence number: index of the first byte of payload in the outgoing data stream
- Acknowledgement number: index of next expected byte in opposite data stream



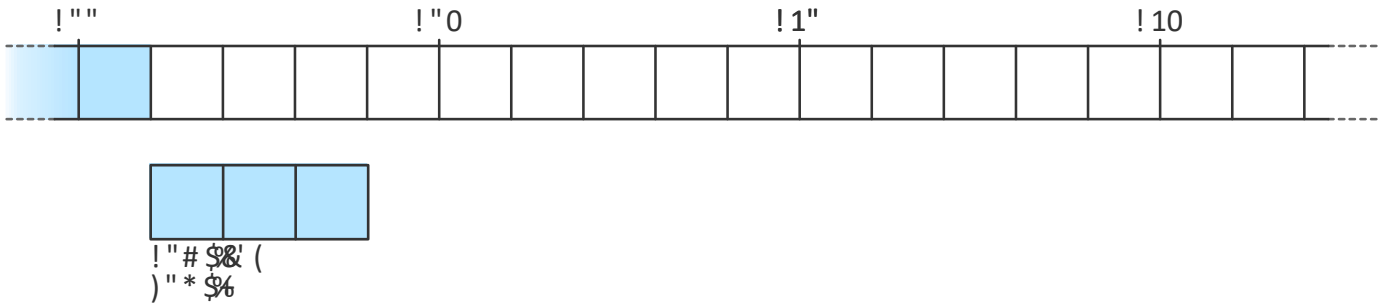
User Datagram Protocol

- Comparison with UDP header
  - Also has ports
  - No sequence number and ack number



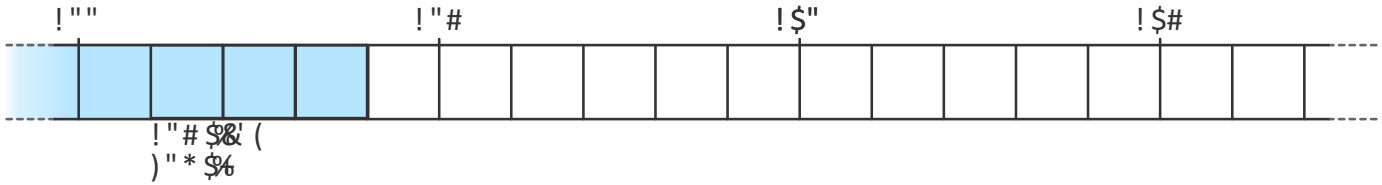
TCP Sequence Number Example

- Sender sends 3-byte segment
- Sequence number indicates where data belongs in data stream (at byte 401)



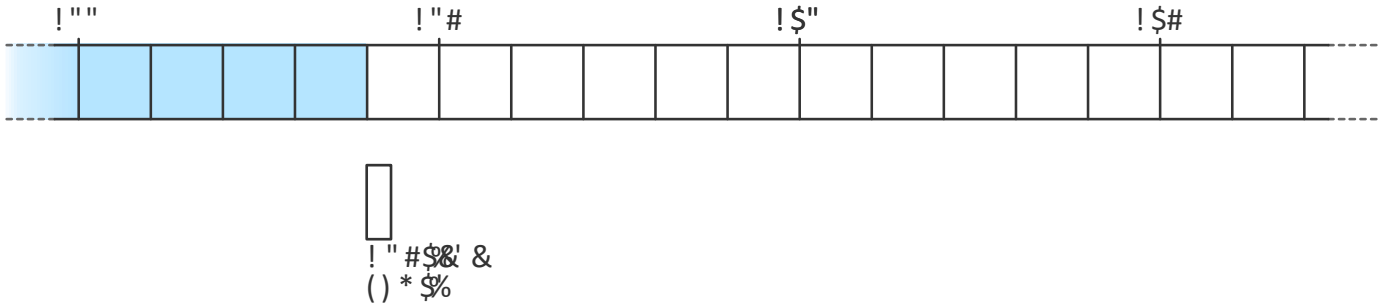
TCP Sequence Number Example

- Receiver adds segment data to receive buffer at position corresponding to byte seq. no. 401



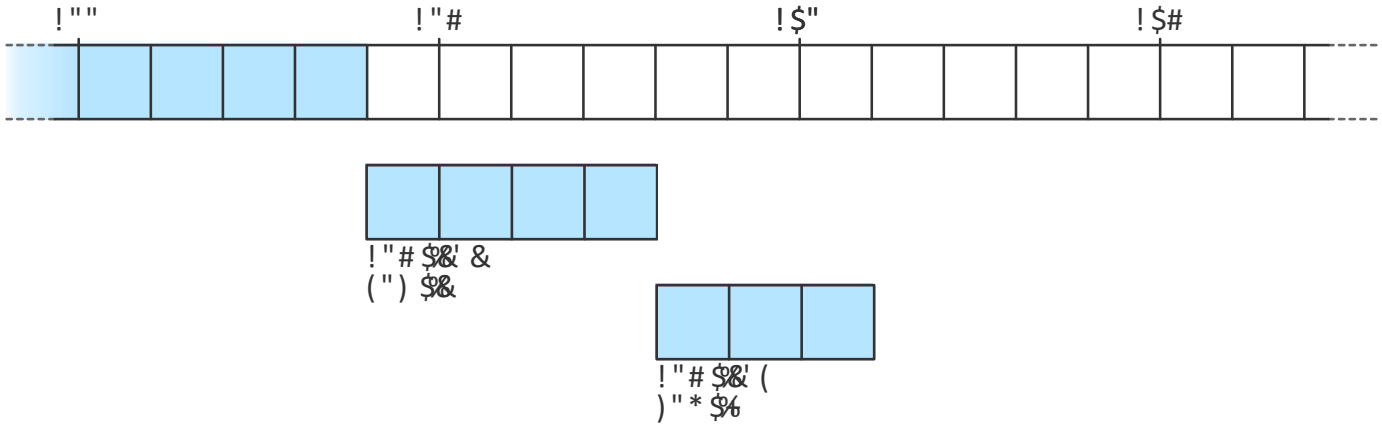
TCP Sequence Number Example

- Receiver acknowledges received data
  - Sets acknowledgement number to indicate next expected byte in sequence



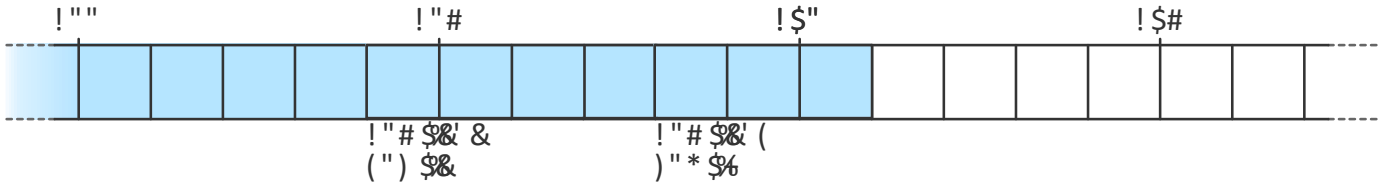
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement



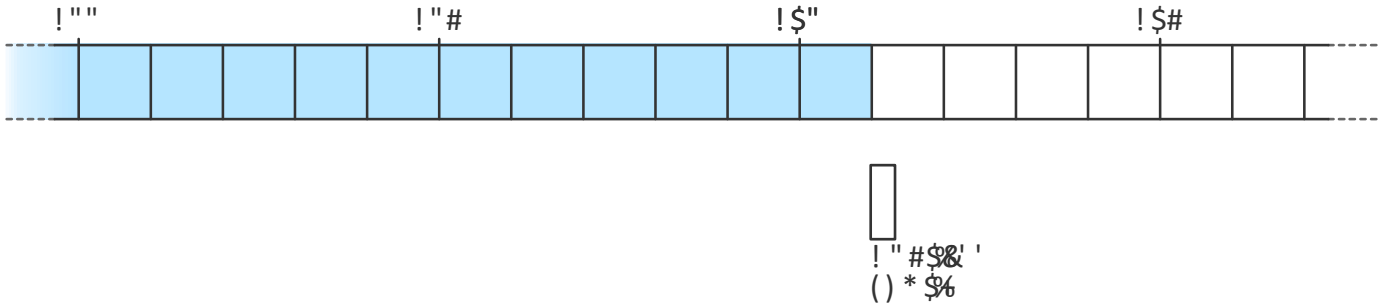
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement



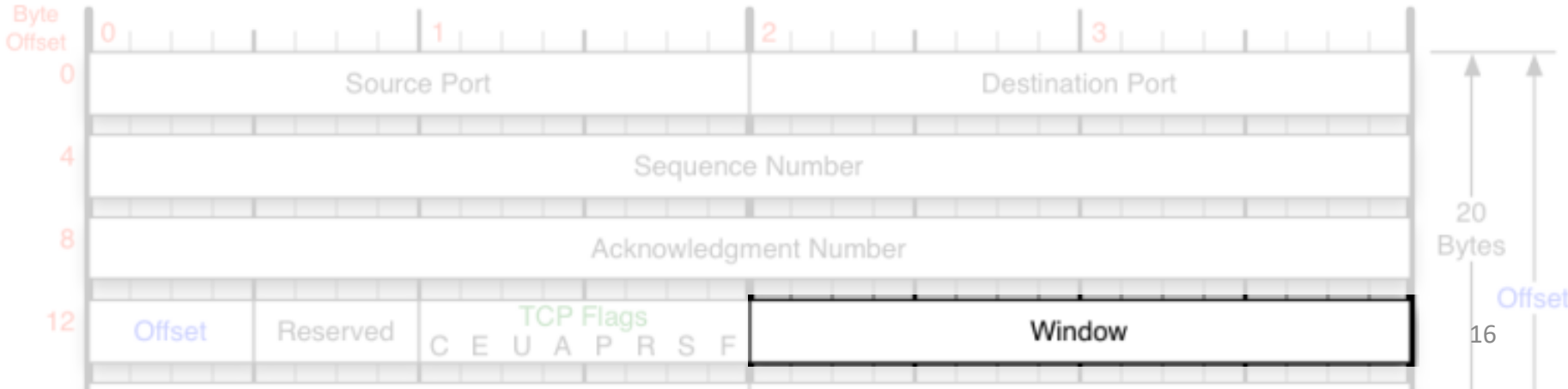
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement
- Receiver always acknowledges with sequence number of next expected byte



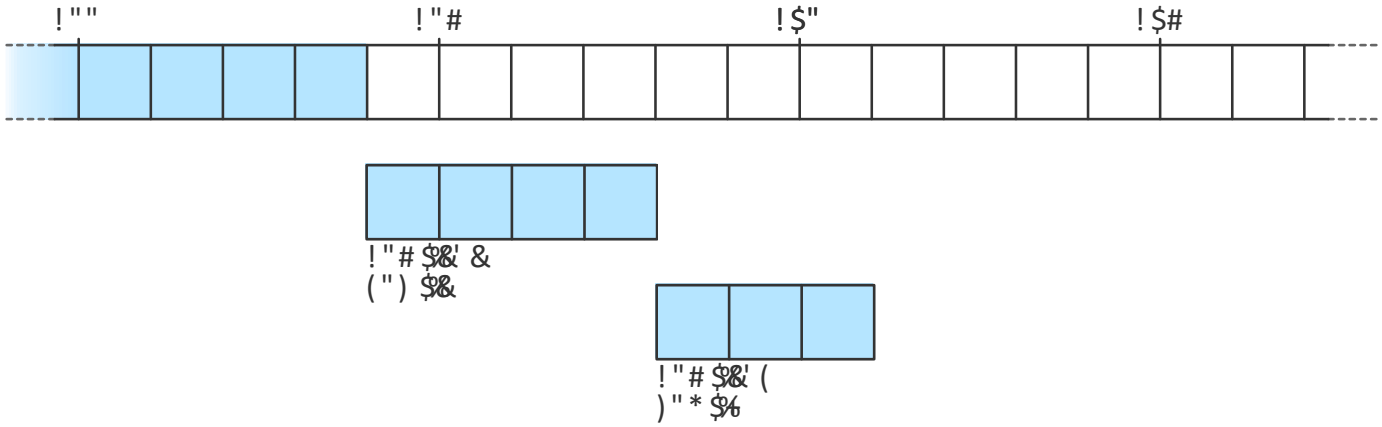
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement
- Maximum number of unacknowledged bytes determined by TCP *window* specified by receiver



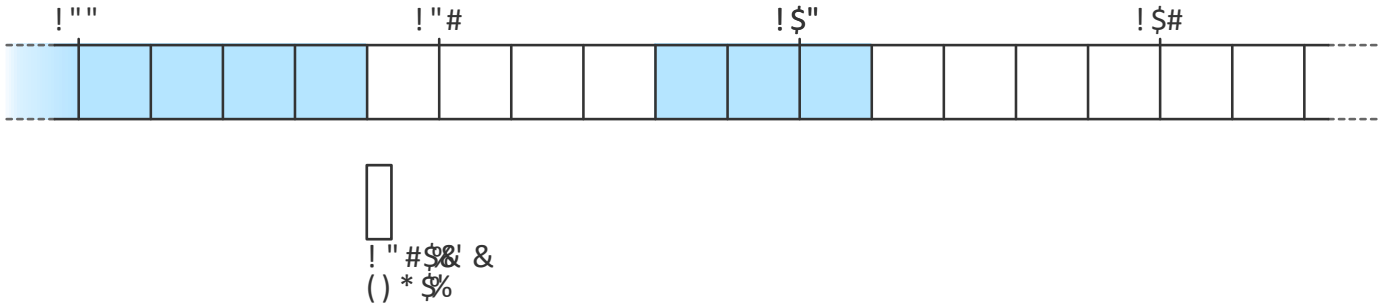
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement
- What if the first packet is dropped in network?



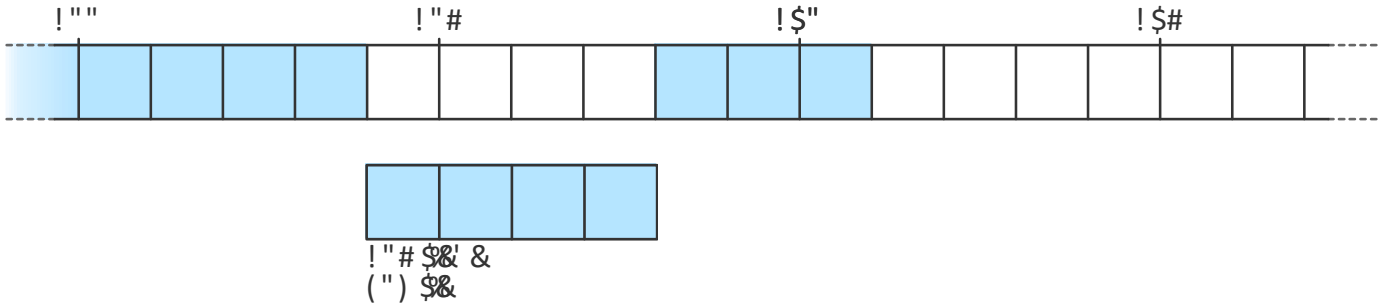
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement
- What if the first packet is dropped in network?
- Receiver always acknowledges with sequence number of next expected byte



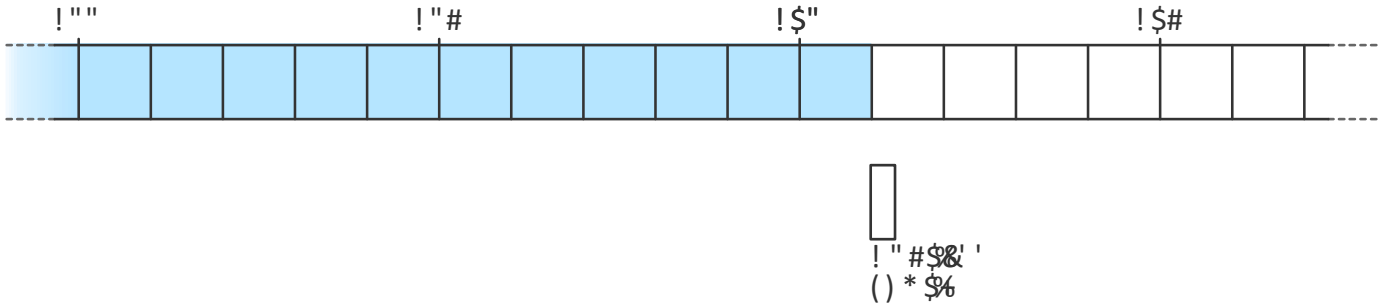
TCP Sequence Number Example

- Sender retransmits lost packet



TCP Sequence Number Example

- Sender retransmits lost packet
- Receiver acknowledges



TCP Sequence and Ack Numbers

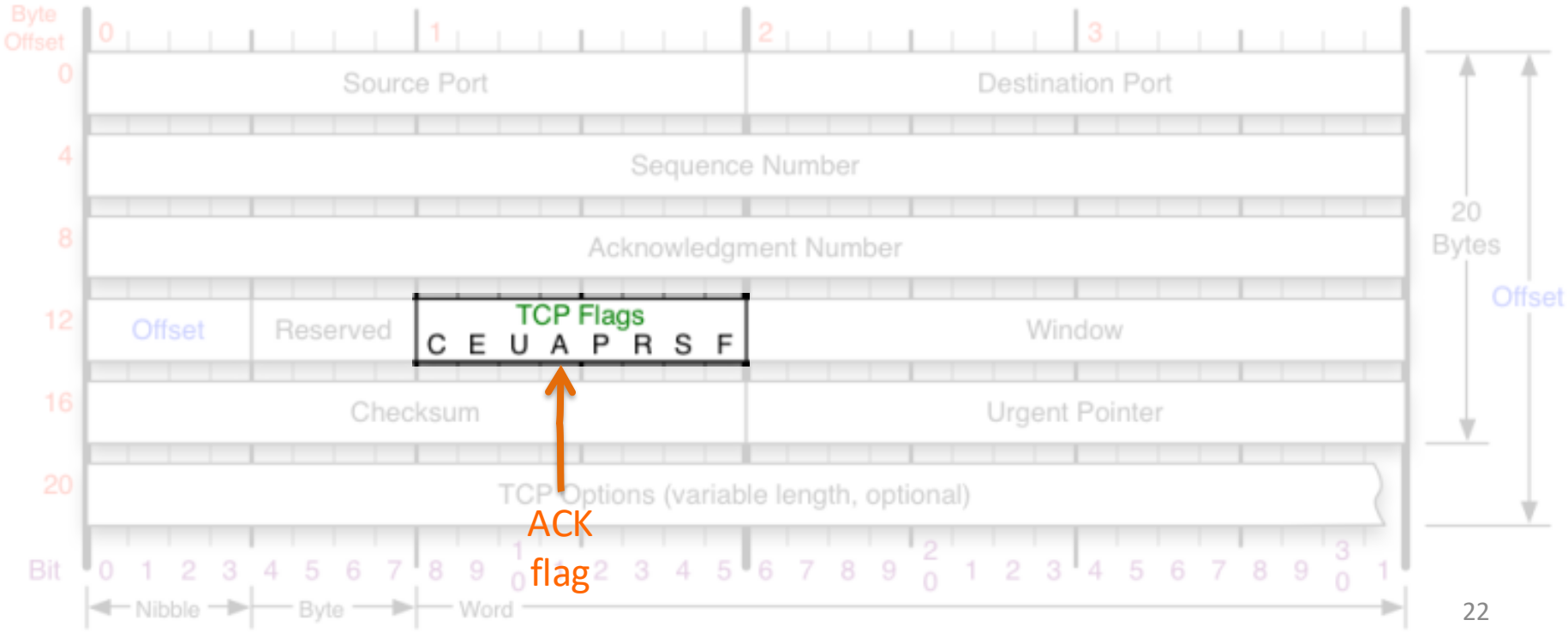
- Initial sequence number is random
  - *Note:* Wireshark shows *relative* sequence numbers
- Sequence numbers wrap around:

  …, 2^{32}-2, 2^{32}-1, 0, 1, 2, …

- ACKs may be piggybacked on data flowing in opposite direction or sent without data
- All packets after initial connection setup will carry an acknowledgement

TCP Flags

- 8 one-bit flags in TCP header
- We'll see how some of others are used later

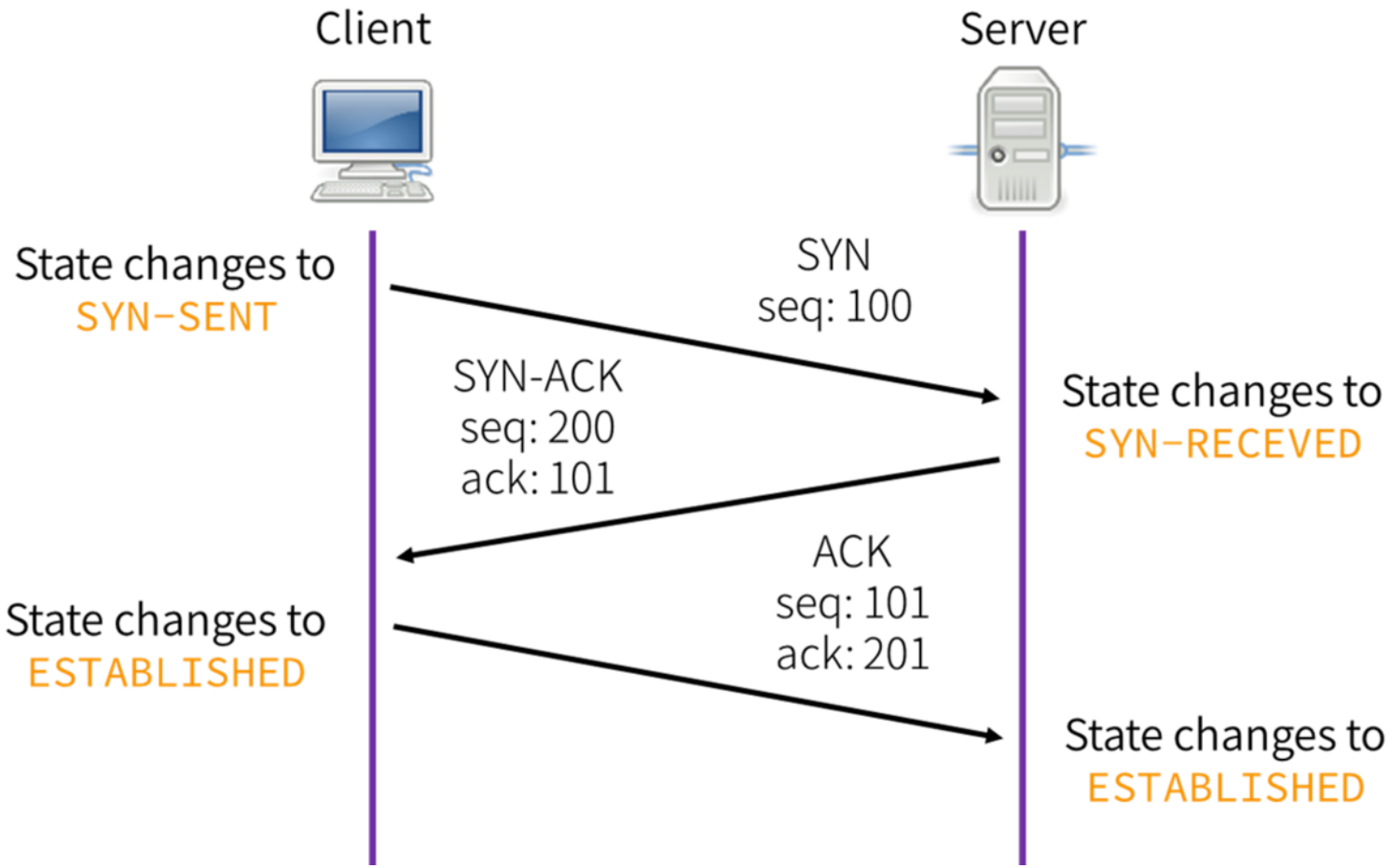


TCP Connection Establishment

- Connection initiator sends a TCP packet with SYN flag set and an initial sequence number
  - Usually the *client* in an application interaction

- Receiver sends a TCP packet with both SYN and ACK flags set and its own initial sequence number (for data in the opposite direction)
  - Usually the *server* in an application interaction

TCP Three-Way Handshake



TCP Normal Connection End

- Eventually one side is ready to end the connection: sends packet with FIN flag set
- The other side acknowledges receipt of FIN packet with ACK
- The other side may keep sending data
- Eventually the other side also sends FIN packet: this terminates the TCP connection

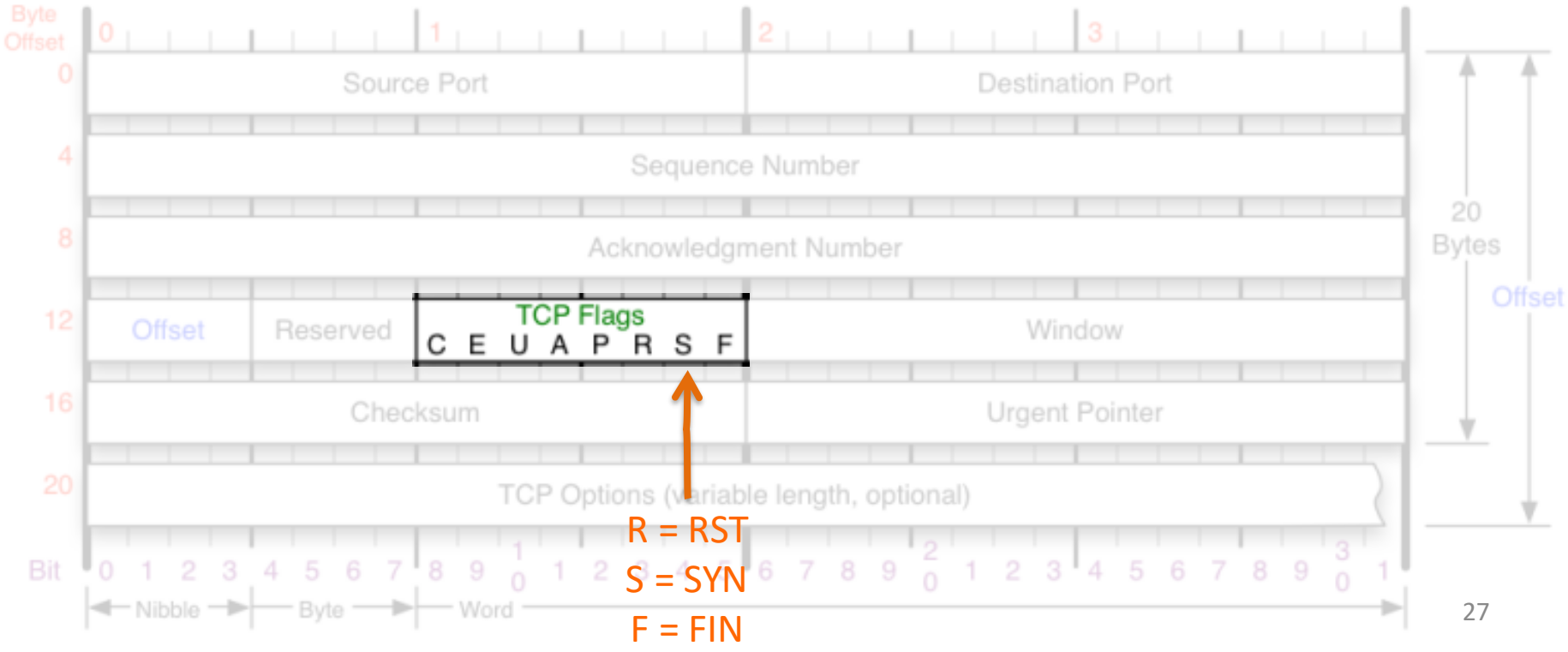
TCP Connection Reset

- Invalid packets will trigger a reset (RST) packet, telling the sender to stop
- If a host receives a TCP packet with RST flag set, it tears down the connection

- Common causes for RST
  - Sequence number outside allowed window
  - Acknowledgement number way out of range
  - Receiving packets when no connection exists

TCP Flags

- 8 one-bit flags in TCP header
  - ACK, RST, SYN, FIN



TCP Security Properties

	Passive	Off-Path	MitM
Availability	—		
Confidentiality		—	
Integrity	—	—	
Authenticity	—		

- Recall passive = passive and on-path
  off-path = active and off-path
  MitM = active and on-path

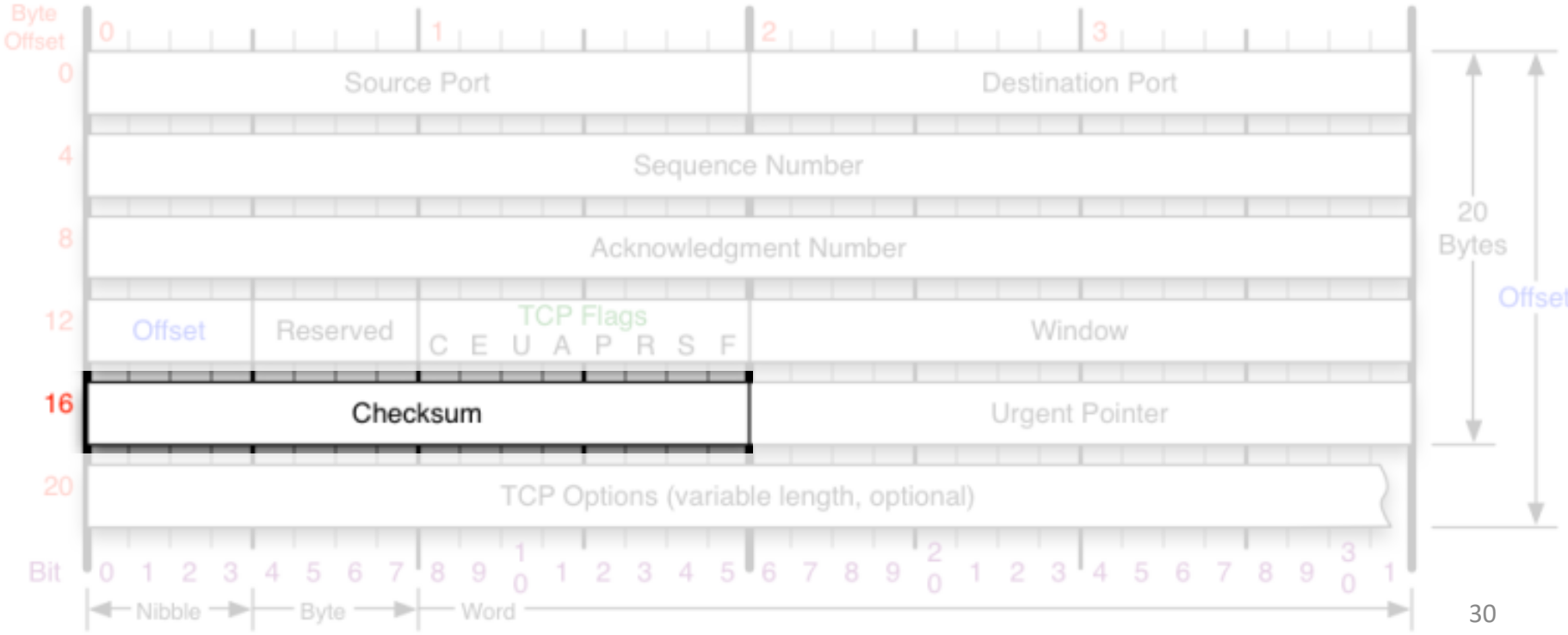
TCP Security Properties

	Passive	Off-Path	MitM
Availability	–		X
Confidentiality	X	–	X
Integrity	–	–	X
Authenticity	–		X

- No confidentiality against an on-path attacker
- No availability against MitM attacker
- No integrity and authenticity against MitM

What about Checksum?

- Can be easily computed for modified/forged packets



TCP Security Properties

	Passive	Off-Path	MitM
Availability	—	?	X
Confidentiality	X	—	X
Integrity	—	—	X
Authenticity	—	?	X

- How about an off-path attacker?

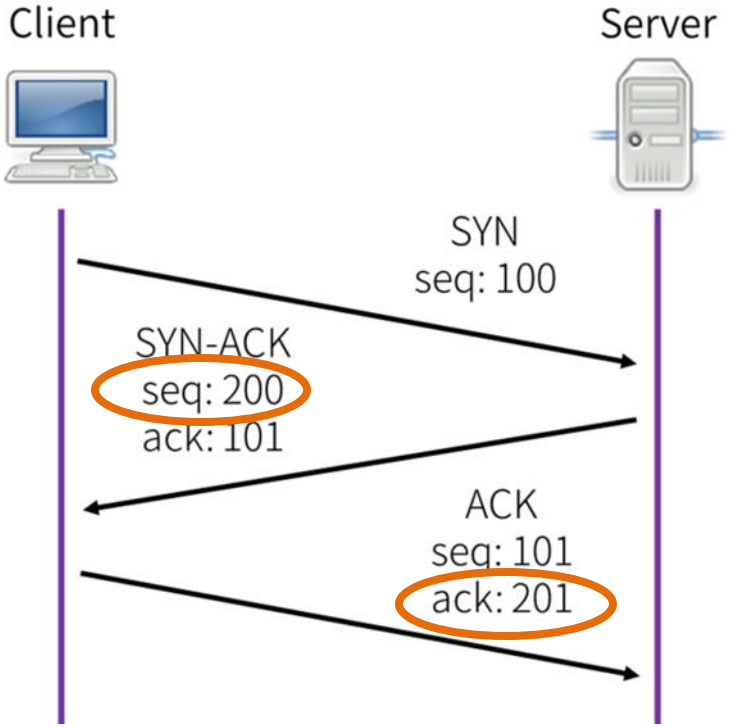
TCP Security Properties

	Passive	Off-Path	MitM
Availability	–	✗	✗
Confidentiality	✗	–	✗
Integrity	–	–	✗
Authenticity	–	?	✗

- How about an off-path attacker?
  - Off-path attacker can violate availability with denial of service attacks
  - What about authenticity?

TCP Connection Spoofing

- Can an off-path attacker impersonate another host when *initiating* a TCP connection?
- No, attacker will not receive SYN-ACK

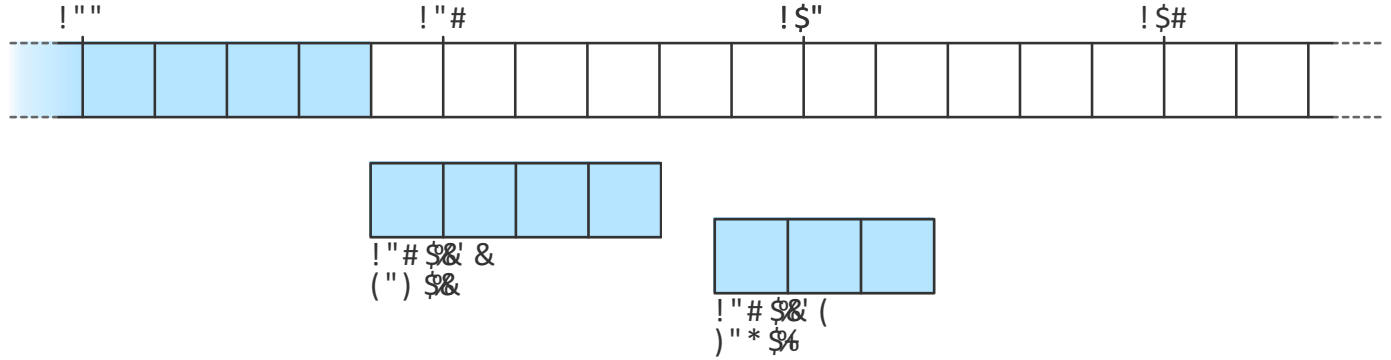


TCP Connection Spoofing

- Can an off-path attacker impersonate another host when *initiating* a TCP connection?
- No, attacker will not receive SYN-ACK
  - Need to guess the random sequence number the server picked, which happens with 2^{-32} probability

TCP Connection Hijacking

- Can an off-path attacker impersonate another host in an *existing* TCP connection?
- No, need to guess:
  - The initiator's port number (16-bit)
    - The respondent's port number is usually well-known
  - The sequence number and ack number (each 32-bit)
    - But can be off within the window size (denoted W)



TCP Connection Hijacking

- Can an off-path attacker impersonate another host in an *existing* TCP connection?
- No, need to guess:
  - The initiator's port number (16-bit)
    - The respondent's port number is usually well-known
  - The sequence number and ack number (each 32-bit)
    - But can be off within the window size (denoted W)
  - Success probability at most $W^2 * 2^{-(16+32+32)}$

TCP Reset Attack

- Can an off-path attacker reset an *existing* TCP connection?
- No, need to guess:
  - The initiator's port number (16-bit)
    - The respondent's port number is usually well-known
  - The sequence number (32-bit)
    - For RST, the sequence number must be exact, not just within window
    - RST packet does not have to carry an ack number
  - Success probability at most $2^{-(16+32)}$

TCP Security Properties

	Passive	Off-Path	MitM
Availability	—	✗	✗
Confidentiality	✗	—	✗
Integrity	—	—	✗
Authenticity	—	?	✗

- How about an off-path attacker?

TCP Security Properties

	Passive	Off-Path	MitM
Availability	–	✗	✗
Confidentiality	✗	–	✗
Integrity	–	–	✗
Authenticity	–	✓	✗

- How about an off-path attacker?
  - TCP (with random initial sequence number) has reasonable authenticity against off-path attackers!

TCP Security Properties

	Passive	Off-Path	MitM
Availability	–	✗	✗
Confidentiality	✗	–	✗
Integrity	–	–	✗
Authenticity	–	✓	✗

- Next lecture: we can use cryptography to achieve confidentiality, integrity, and authenticity even against MitM attackers

Layering of Protocols