Lecture 25: The Internet (Abridged): Network and Transport Layers

Professor Adam Bates

CS 461 / ECE 422

Fall 2019

Goals for Today

- Learning Objectives:
  - Understand the fundamental building blocks of the Internet, specifically the Network and Transport Layers
  - Consider the security issues at each layer
- Announcements, etc:
  - MP2 Checkpoint 2 grades released
  - MP4 is out! Checkpoint #1 due Oct 30 6pm

Reminder: Please put away devices at the start of class

Internet Protocol

- **Internet Protocol (IP)** defines *structure* of packets and *how they are handled* by routers
  - IP packets are also called *datagrams*
- IP packets have an IP header that tells routers what to do with the packet
- Rest of packet (payload) is ignored by router
  - Not true anymore: *middleboxes* may examine and modify payload (e.g. to detect malware)

(Lay) Security Properties

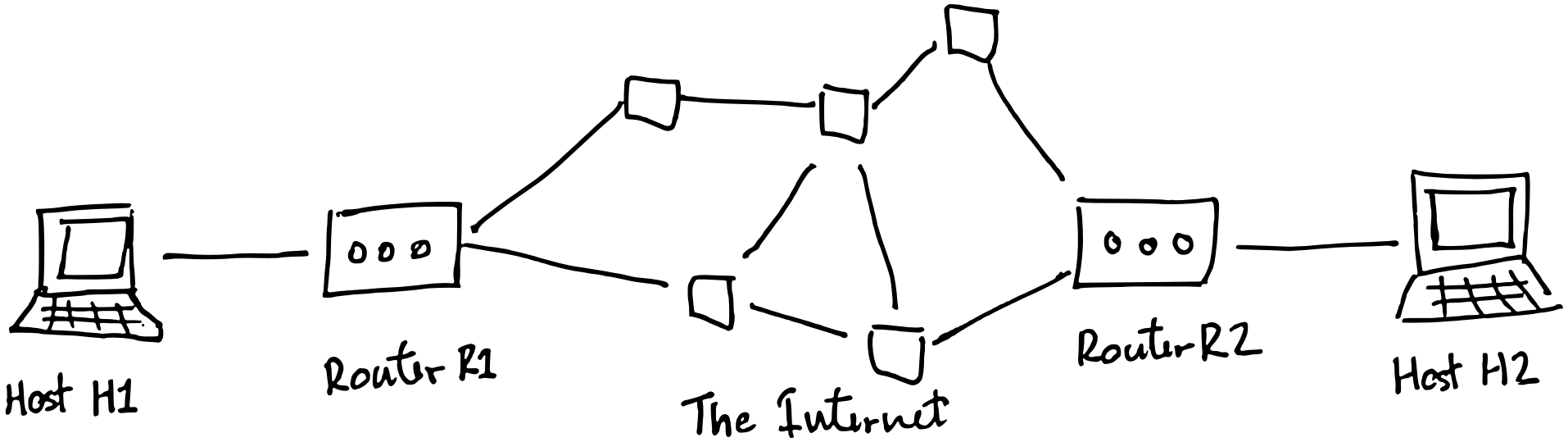
- **Availability:**
  no one can deny me access to services
- **Confidentiality:**
  no one can "see" my private information
- **Integrity:**
  no one can "mess with" my data
- **Authenticity:**
  no can pretend to be someone else

(Technical) Security Properties

- **Availability:**
  attacker can't prevent communication
- **Confidentiality:**
  attacker can't learn protected information
- **Integrity:**
  attacker can't modify communications
- ***Authenticity:***
  attacker can't forge communications

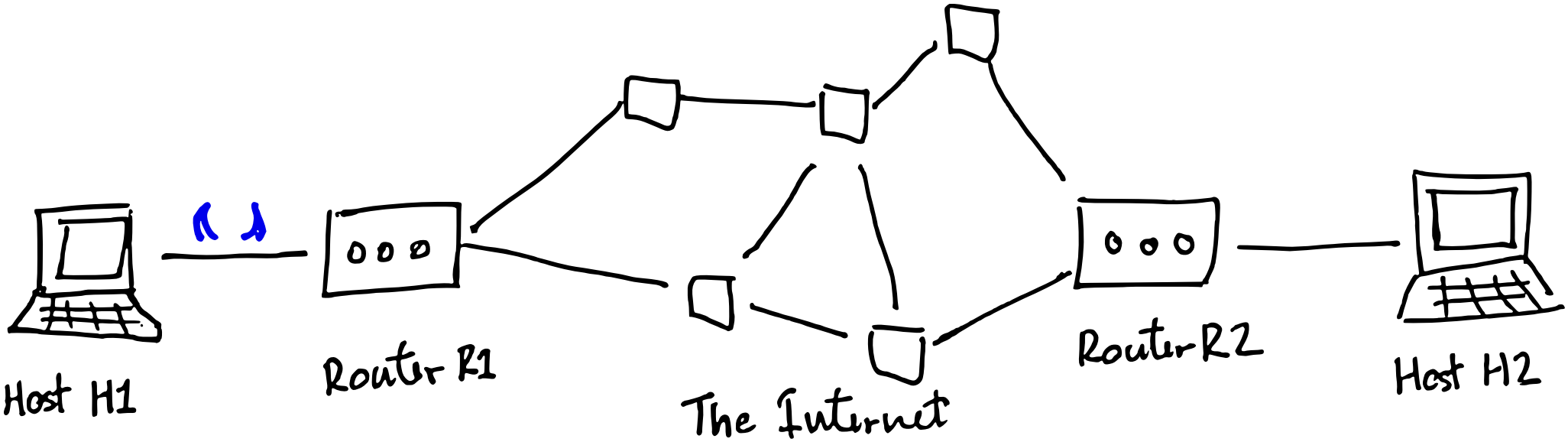
Security Properties

- What security properties does IP have?
- Availability? Confidentiality? Integrity? Authenticity?
- *Depends on attacker capability*
  - Passive Off-Path, Man-in-the-Middle



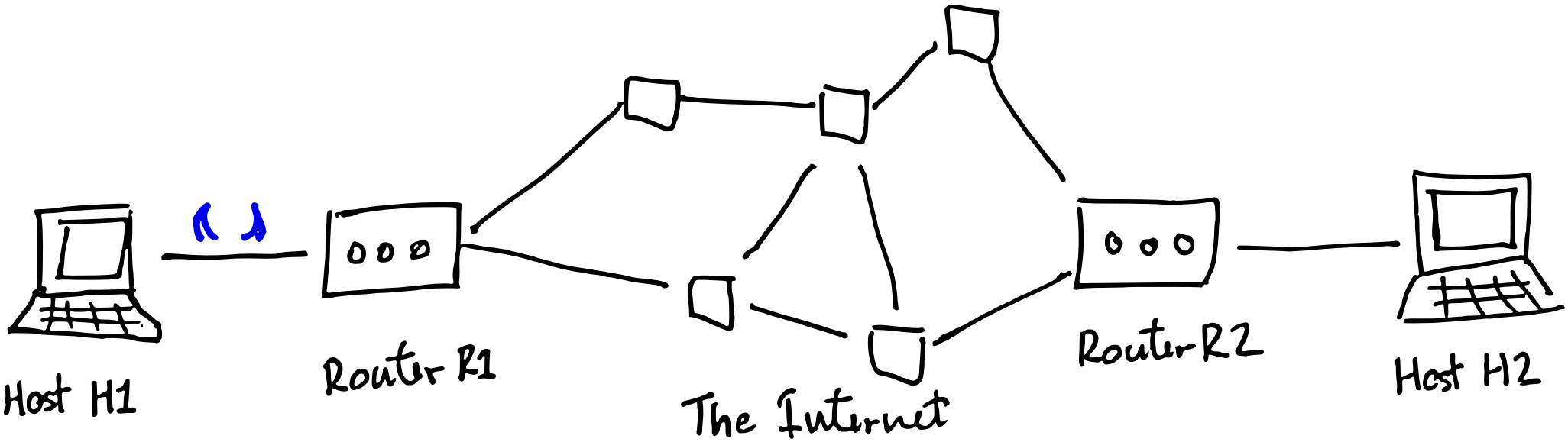
Network Attacker Models

- **Passive attacker:**
  can see all packets but cannot modify them
- Scenario?



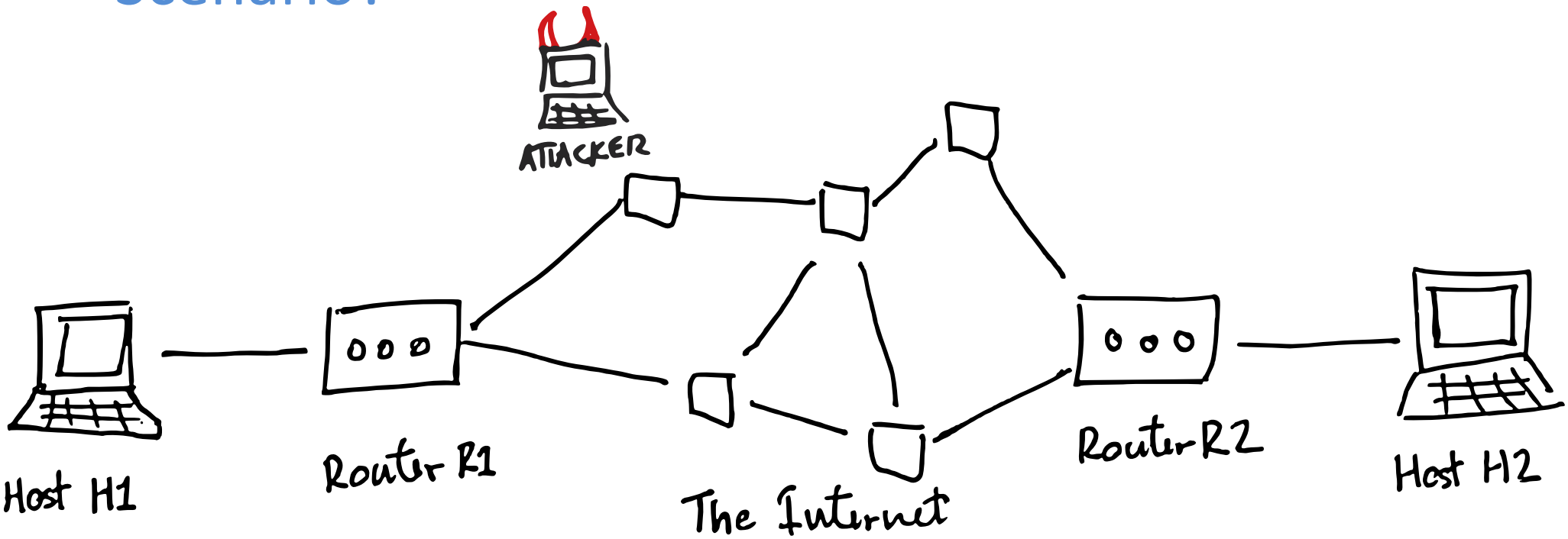
Network Attacker Models

- **Passive attacker:**
  can see all packets but cannot modify them
- Scenario?



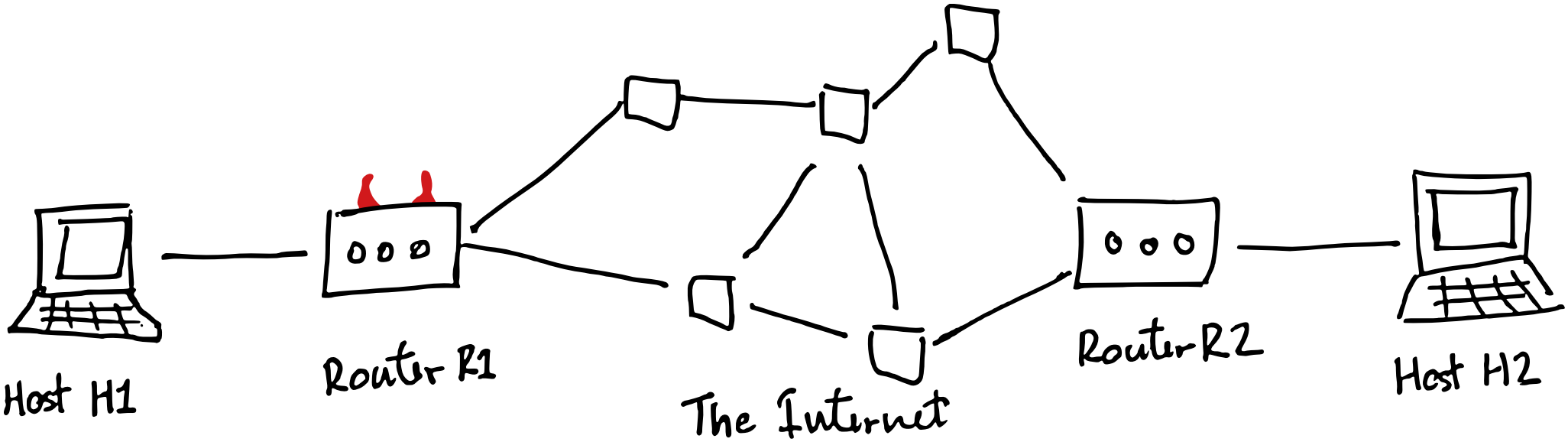
Network Attacker Models

- **Off-Path attacker:**
  can inject packets into network,
  but *cannot* see traffic between other hosts
- Scenario?



Network Attacker Models

- **Man-in-the-Middle attacker:**
  can see, inject, and drop all packets
- Scenario?



Security Properties

- **Availability?**
  attacker can't prevent communication
- **Confidentiality?**
  attacker can't learn protected information
- **Integrity?**
  attacker can't modify communications
- ***Authenticity?***
  attacker can't forge communications

IP Security Properties

	Passive	Off-Path	MitM
Availability			
Confidentiality			
Integrity			
Authenticity			

IP Security Properties

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

- By definition:
  - Passive attacker cannot modify or send packets
  - Off-path attacker cannot see or modify packets
  - MitM attacker can always block packets

IP Security Properties

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

- Confidentiality against a passive attacker?

IP Security Properties

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

- Confidentiality against a passive attacker? ✗
  - MitM can do whatever passive attacker can

IP Security Properties

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

- Integrity against a MitM attacker?
- What about header checksum?

IP Security Properties

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

- Integrity against a MitM attacker? ✗
- Header checksum can be updated by attacker
  - Requires no secret information to compute
  - Does not cover payload

IP Security Properties

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

- Authenticity? Source address indicates who sent the packet…

IP Security Properties

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

- Authenticity? Source address indicates who sent the packet…
- Informational only: not enforced by routers
- Off-path or MitM can set source address to anything

IP Security Properties

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

- Can an off-path attacker affect another host's ability to communicate with any other host?

IP Security Properties

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

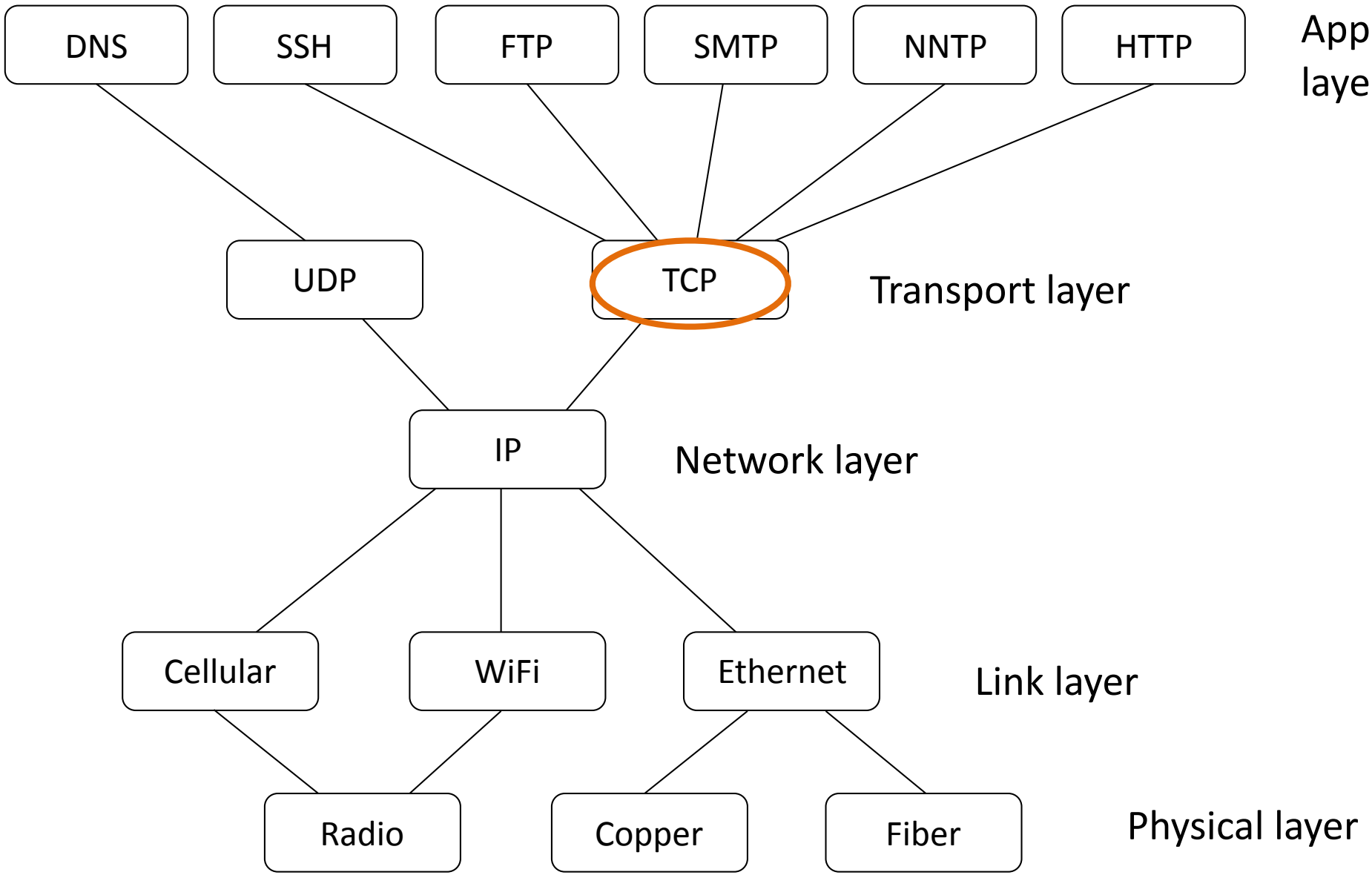
- Network denial-of-service attacks can saturate network preventing other communications
- Hosts and routers may have other limited resources
  - E.g. number of connections (we'll see this later)

IP Security Properties

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

- We'll see how we can build protocols built *on top of* IP to provide some of these security properties

Layering of Protocols

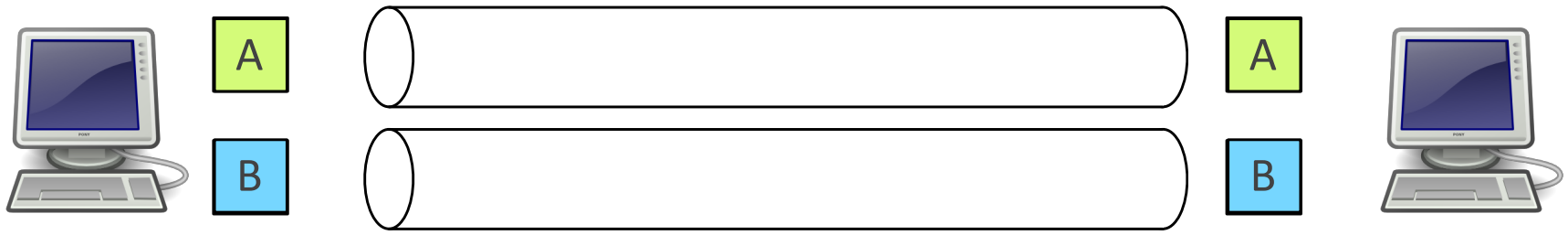


From Packets to Data Streams

- Most Internet applications want a data stream abstraction (not best-effort packets)
- Application on host X wants to send a sequence of bytes to application on host Y
  - Want: reliable, in-order delivery of data
- **Transmission Control Protocol (TCP)** *provides* a data stream abstraction *using* a best-effort packet transport (IP)

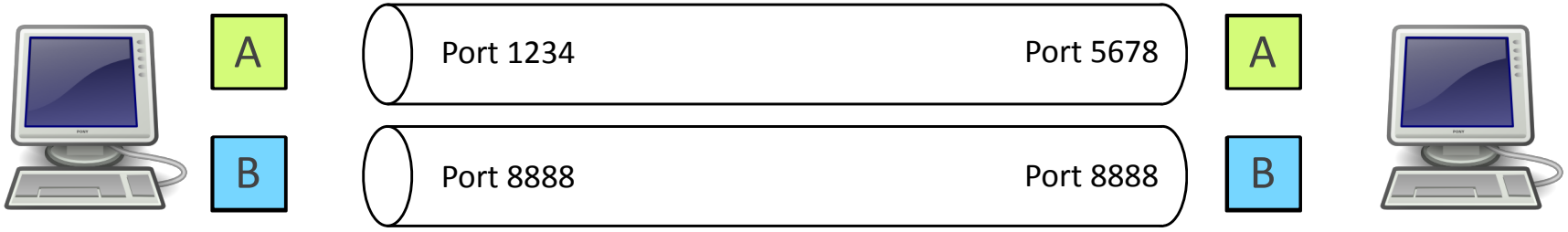
Transmission Control

- Have: network that will deliver packets
  - Packets may be dropped, re-ordered, duplicated
- Want to provide: abstraction of a stream of bytes between applications on different hosts
  - Bytes delivered reliably and in-order



Transmission Control

- Each application 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)
- Some destination port numbers used for specific applications by convention



TCP Port Numbers

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

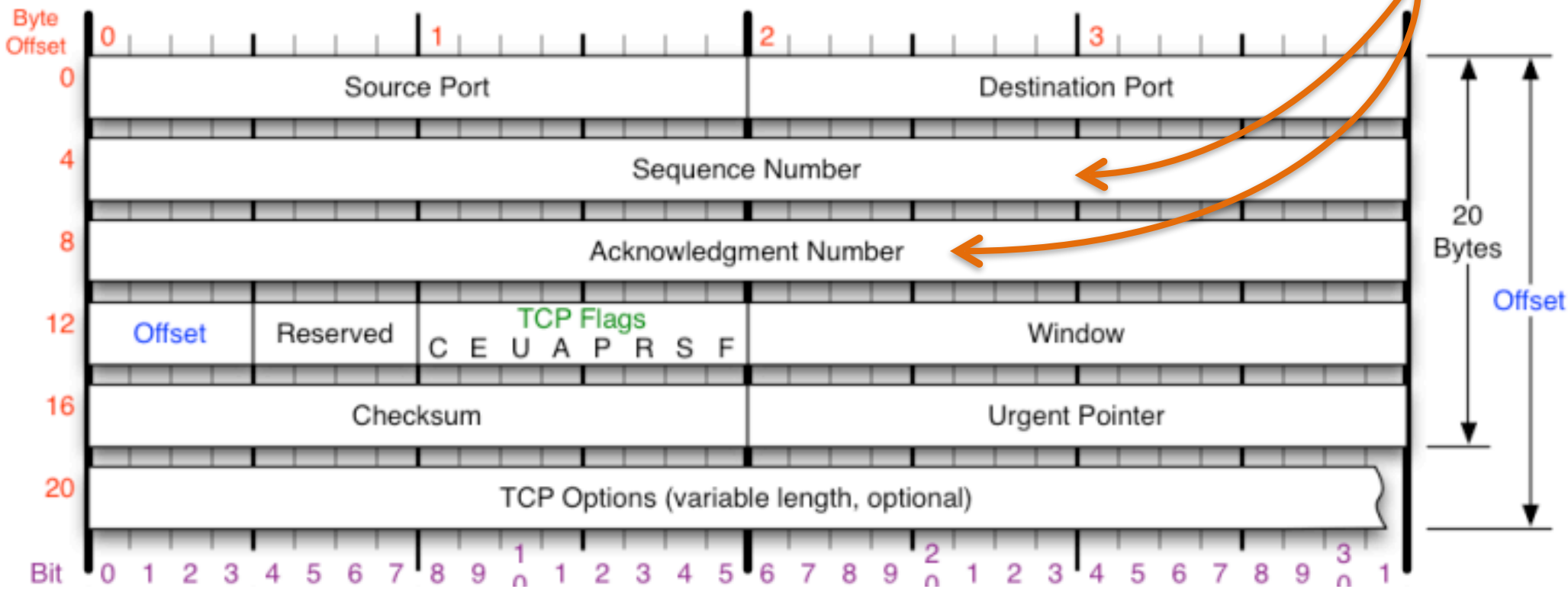
https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

TCP

- Bytes in application data stream numbered with a 32-bit *sequence number*
- Data sent in *segments:* sequences of contiguous bytes sent in a single IP datagram
- There are two logical data streams in a TCP session, one in each direction

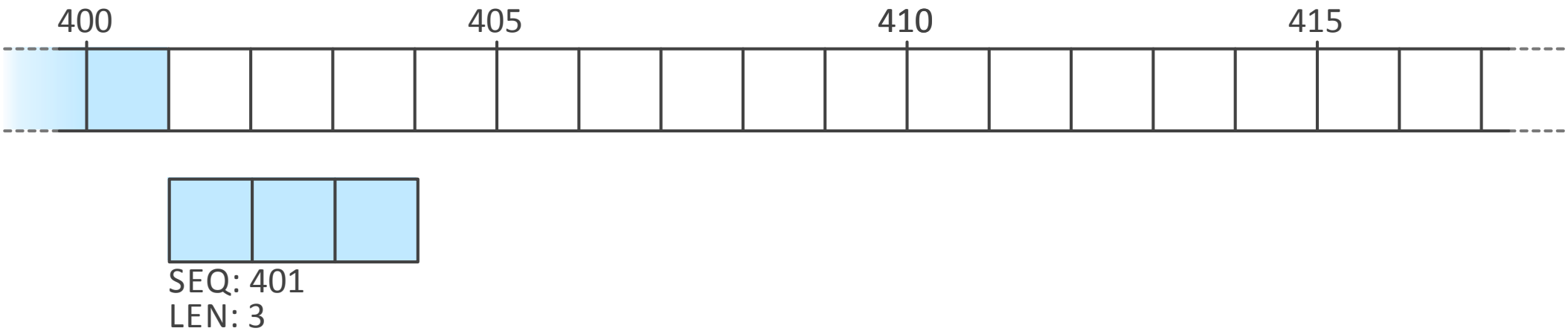
TCP

- Sequence number in packet header is sequence number of first byte of payload
- Acknowledgement number is sequence number of next expected byte number of stream in opposite direction



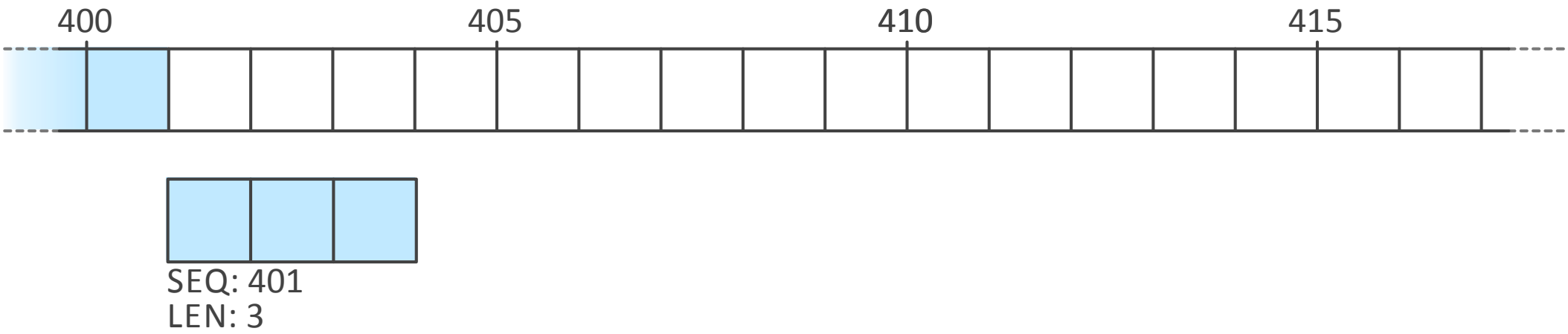
TCP Sequence Number Example

- Sender sends 3 byte segment
- Sequence number indicates where data belongs in byte sequence (at byte 401)
  - *Note:* Wireshark shows *relative* sequence numbers

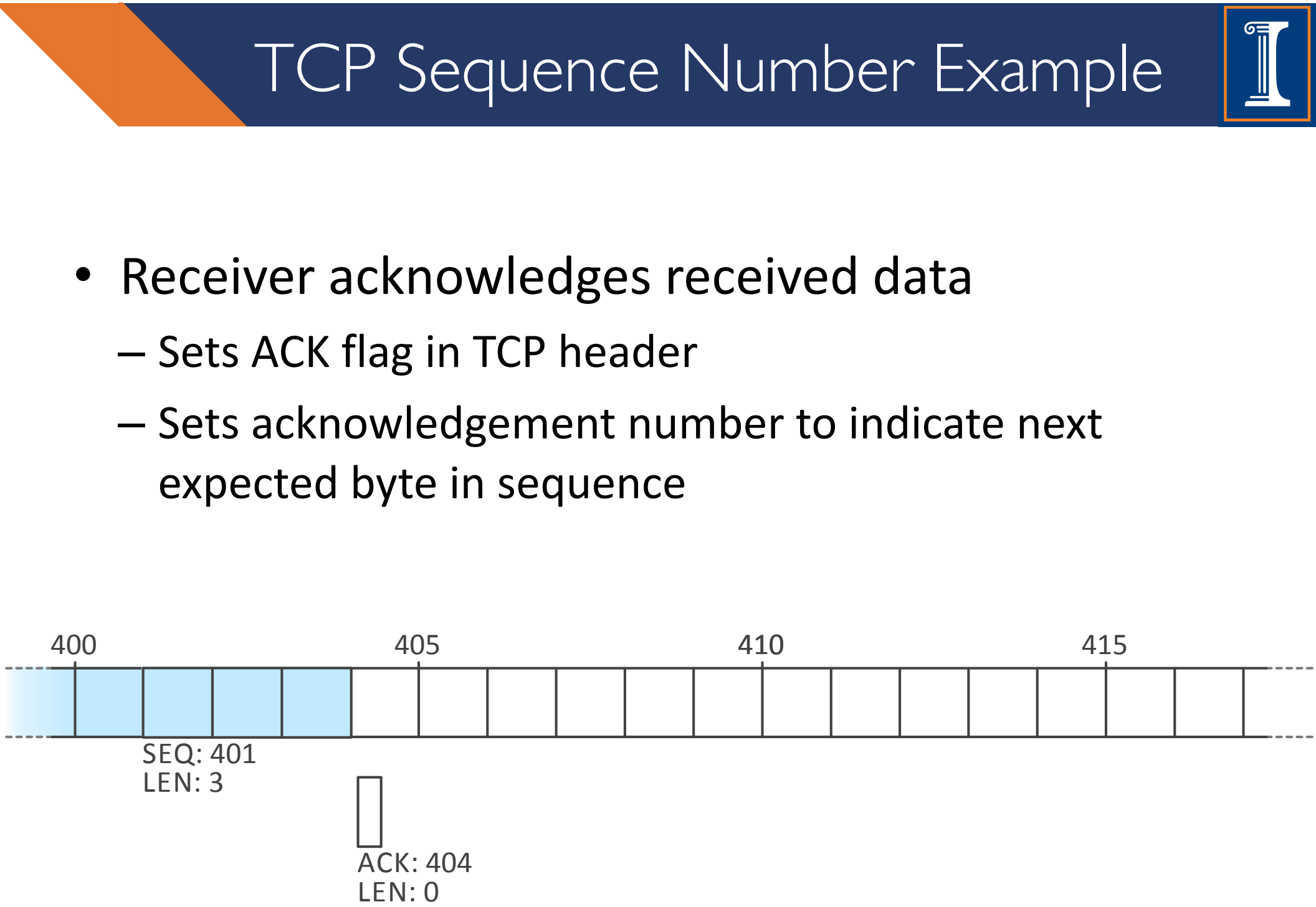


TCP Sequence Number Example

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



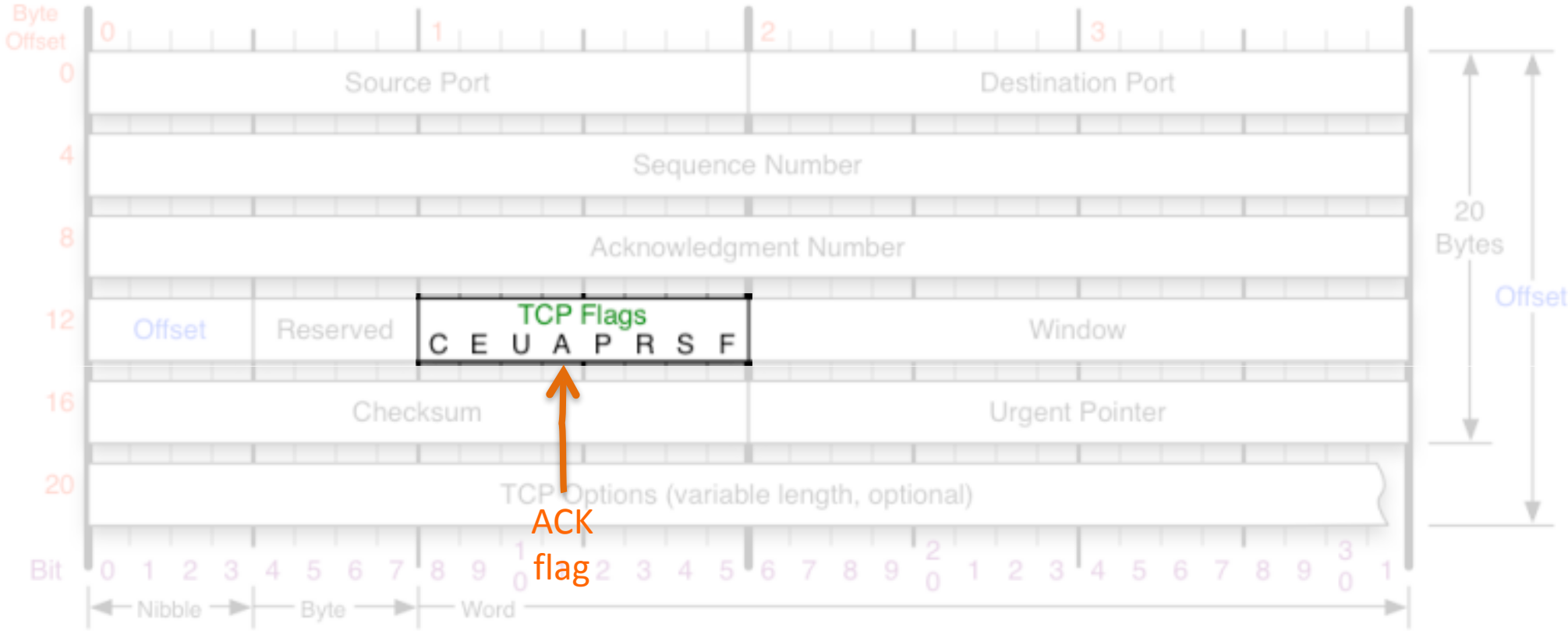
TCP Sequence Number Example



- Receiver acknowledges received data
  - Sets ACK flag in TCP header
  - Sets acknowledgement number to indicate next expected byte in sequence

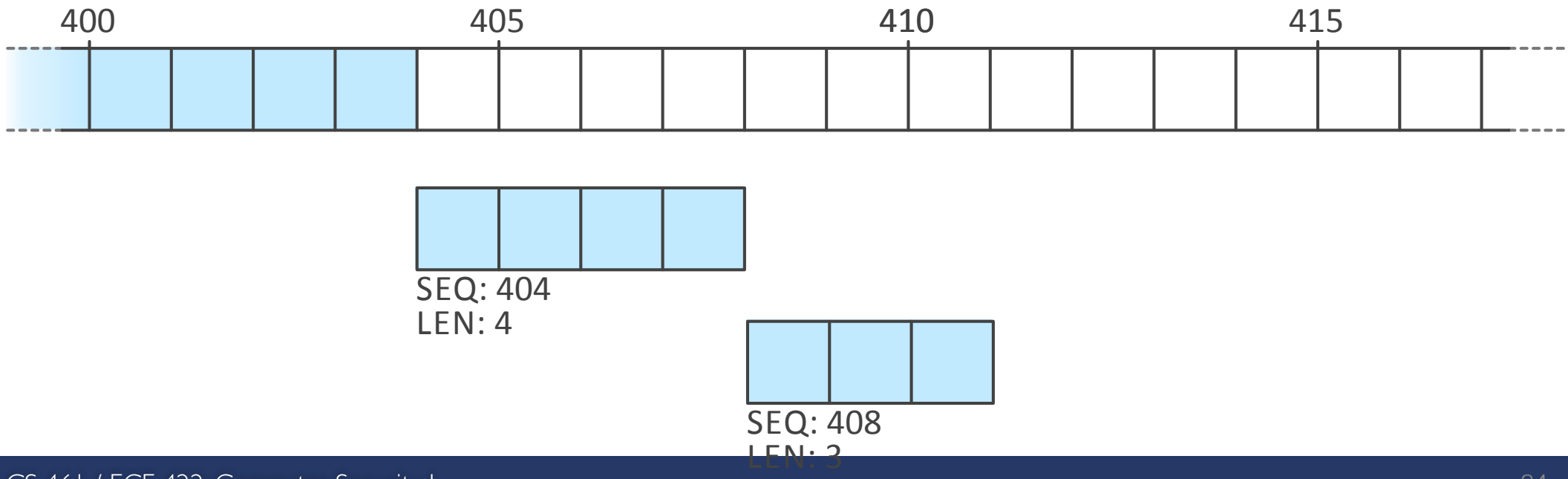
TCP Flags

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



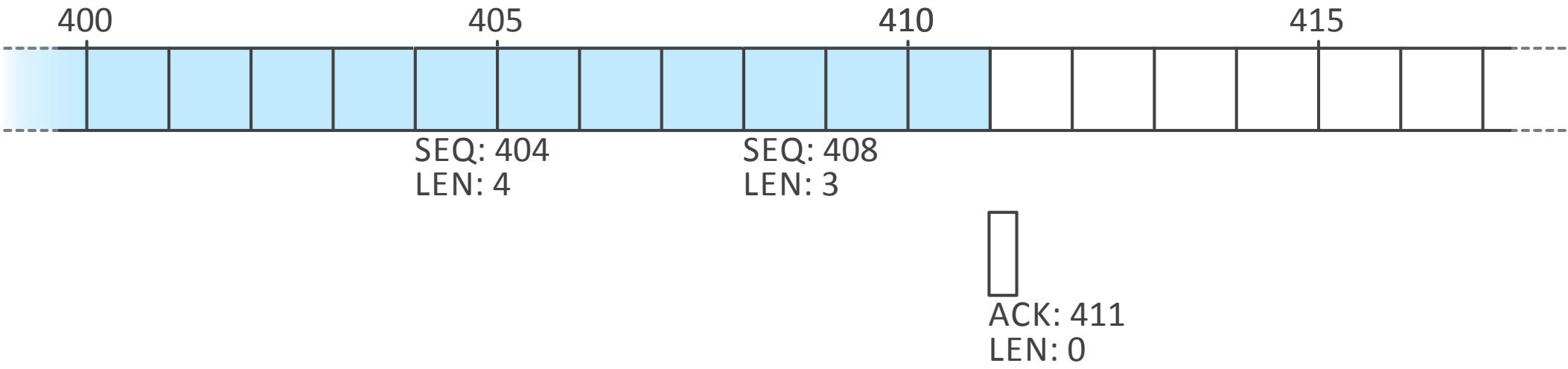
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement



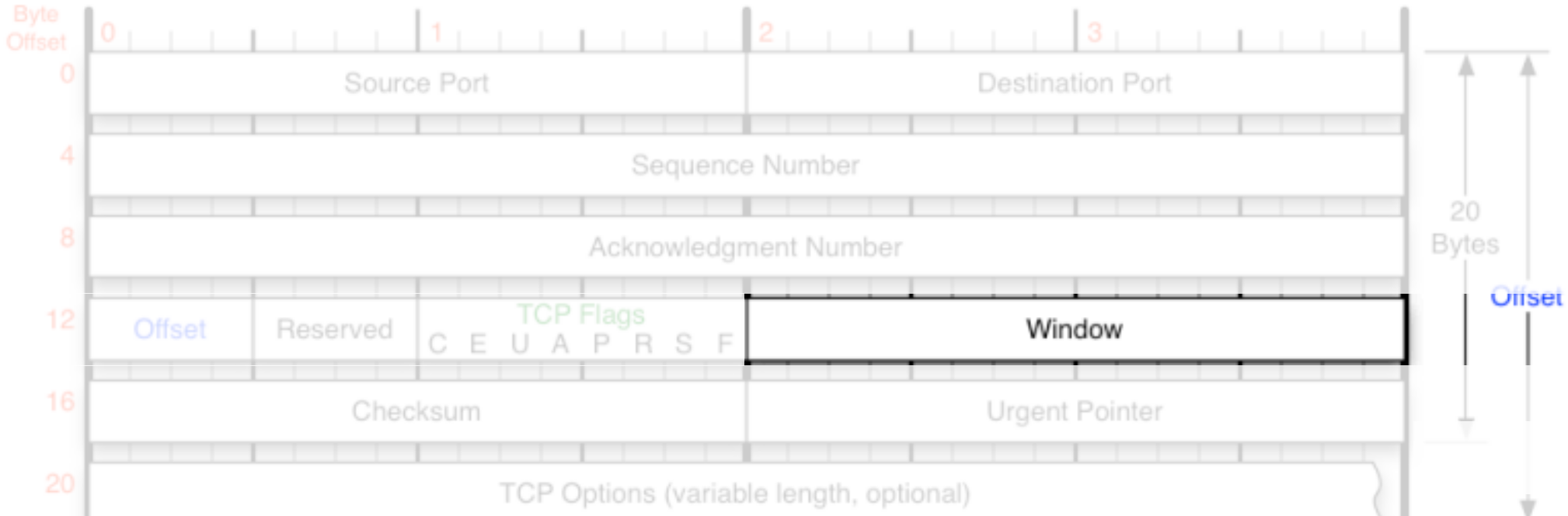
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement
- Receiver always acknowledges with seq. no. of next expected byte



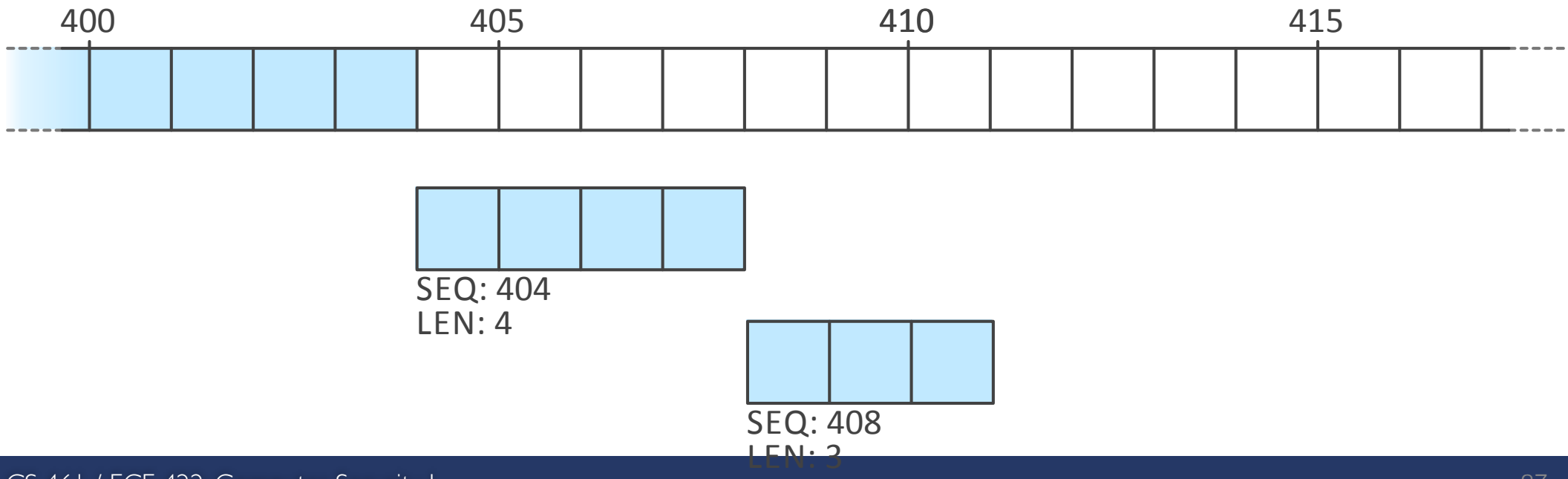
TCP Sequence Number Example

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



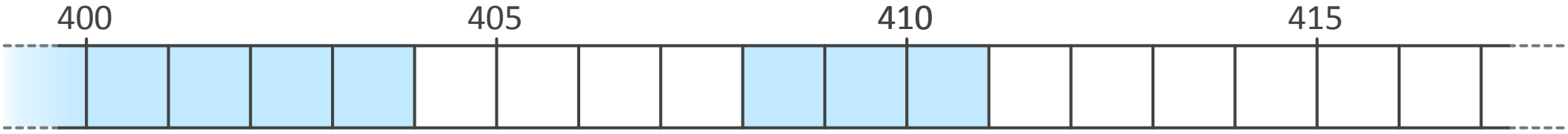
TCP Sequence Number Example

- Sender may send several segments before receiving acknowledgement (up to Window size)



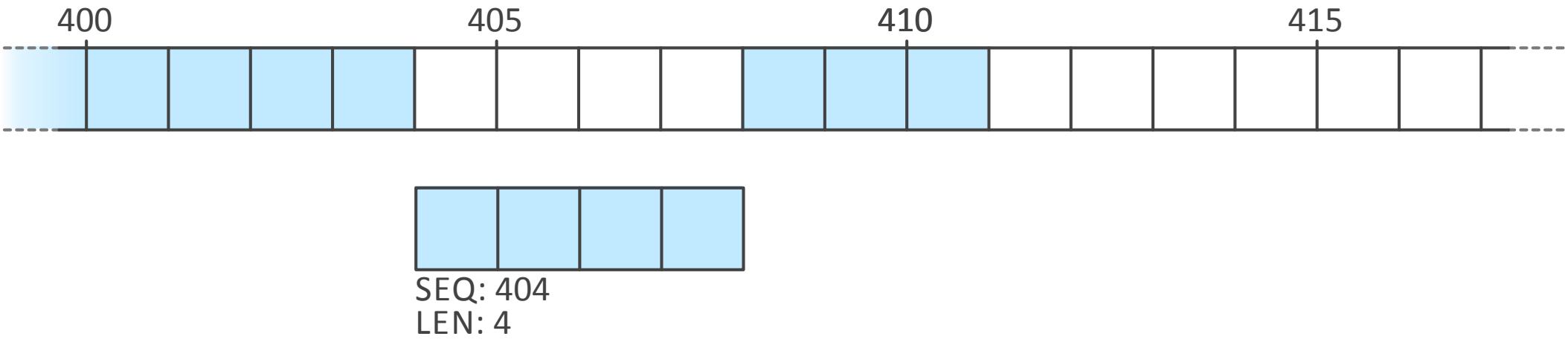
TCP Sequence Number Example

- What if a packet is dropped in the network?
- *Receiver always acknowledges with seq. no. of next expected byte*
- Sender retransmits lost data

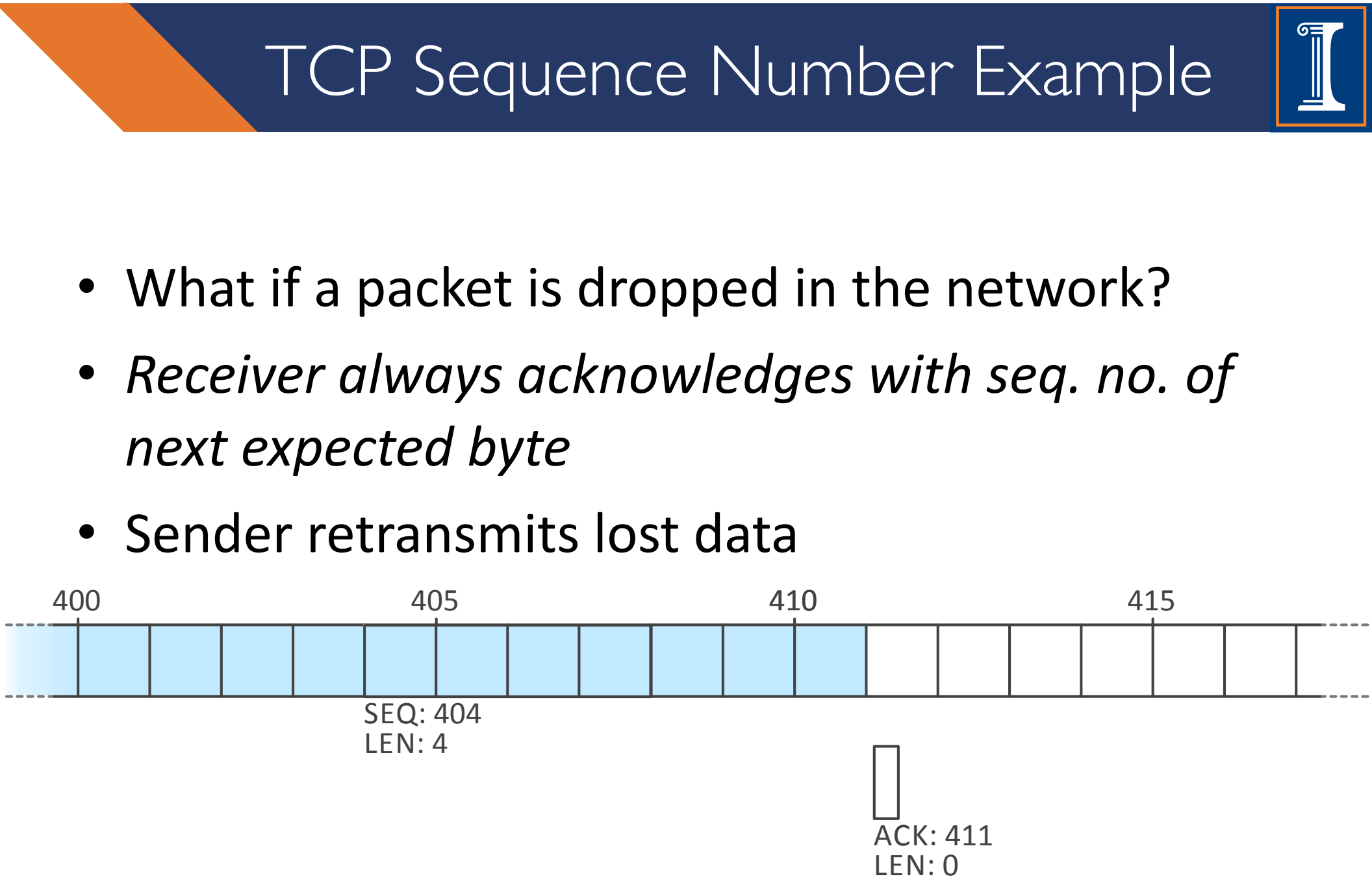


TCP Sequence Number Example

- What if a packet is dropped in the network?
- *Receiver always acknowledges with seq. no. of next expected byte*
- Sender retransmits lost data



TCP Sequence Number Example



- What if a packet is dropped in the network?
- *Receiver always acknowledges with seq. no. of next expected byte*
- Sender retransmits lost data

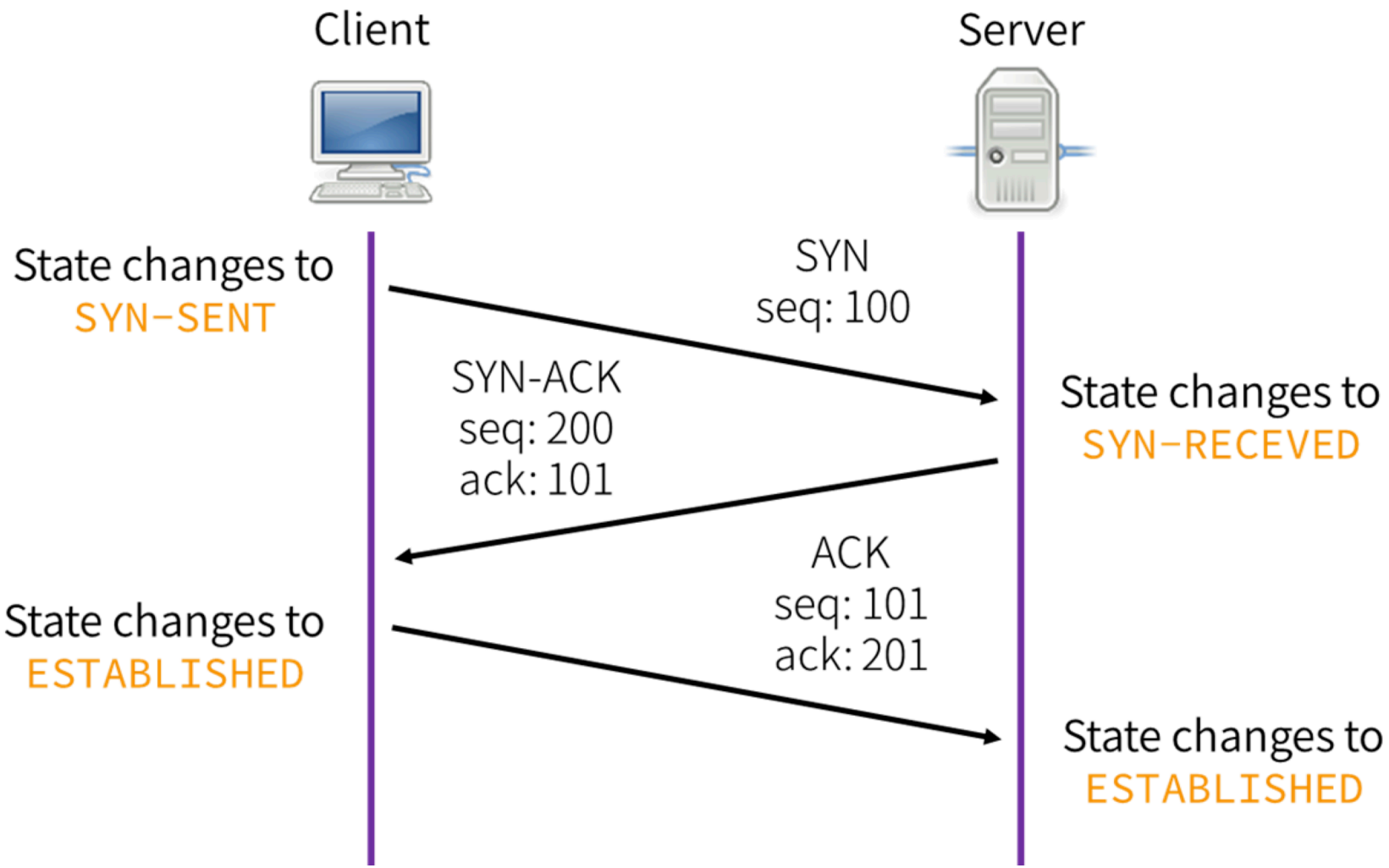
TCP Sequence Numbers

- ACKs may be piggybacked on data flowing in opposite direction or sent without data
- All packets after initial connection setup will carry an acknowledgement
- Sequence numbers wrap around:
  …, 2^{32}-2, 2^{32}-1, 0, 1, 2, …

TCP Connection Establishment

- Connection initiator sends TCP packet with SYN flag set and an initial sequence number
  - Usually the *client* in an application interaction
- Responding host sends TCP packet with both SYN and ACK flags set and its own initial sequence number (for data in opposite dir.)
  - Usually the *server* in an application interaction
  - Acknowledges received sequence number (acknowledgement number field)

TCP Three-Way Handshake



TCP Normal Connection End

- Eventually one side is ready to end the connection: sends packet with FIN flag set
  - Must have ACK flag with valid sequence number
- Peer receiving FIN packet acknowledges receipt of FIN packet with ACK
  - FIN "consumes" one byte of seq. number space
- Eventually other side sends packet with FIN flag set: this terminates the TCP session

TCP Connection Reset

- TCP designed to handle possibility of spurious TCP packets (e.g. from previous connections)
- Packets that are invalid given current state of session generate a reset
  - If a connection exists, it is torn down
  - Packet with RST flag sent in response
- If a host receives a TCP packet with RST flag, it tears down the connection

TCP RST Causes

"As a general rule, reset (RST) must be sent whenever a segment arrives which apparently is not intended for the current connection. A reset must not be sent if it is not clear that this is the case." *(RFC 793)*

- Sequence number outside allowed window
- Acknowledgement number way out of range
- Attempting to open (SYN flag) a connection without a listening process on receiving host
- Any packet when no connection exists
  - Connection identified by tuple of local IP address, remote IP address, local port, remote port

TCP Security Properties

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

- Does TCP provide any additional security beyond what IP alone provides?

TCP Security Properties

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

- Confidentiality against a passive attacker? ✗
  - MitM can always do what passive attacker can do

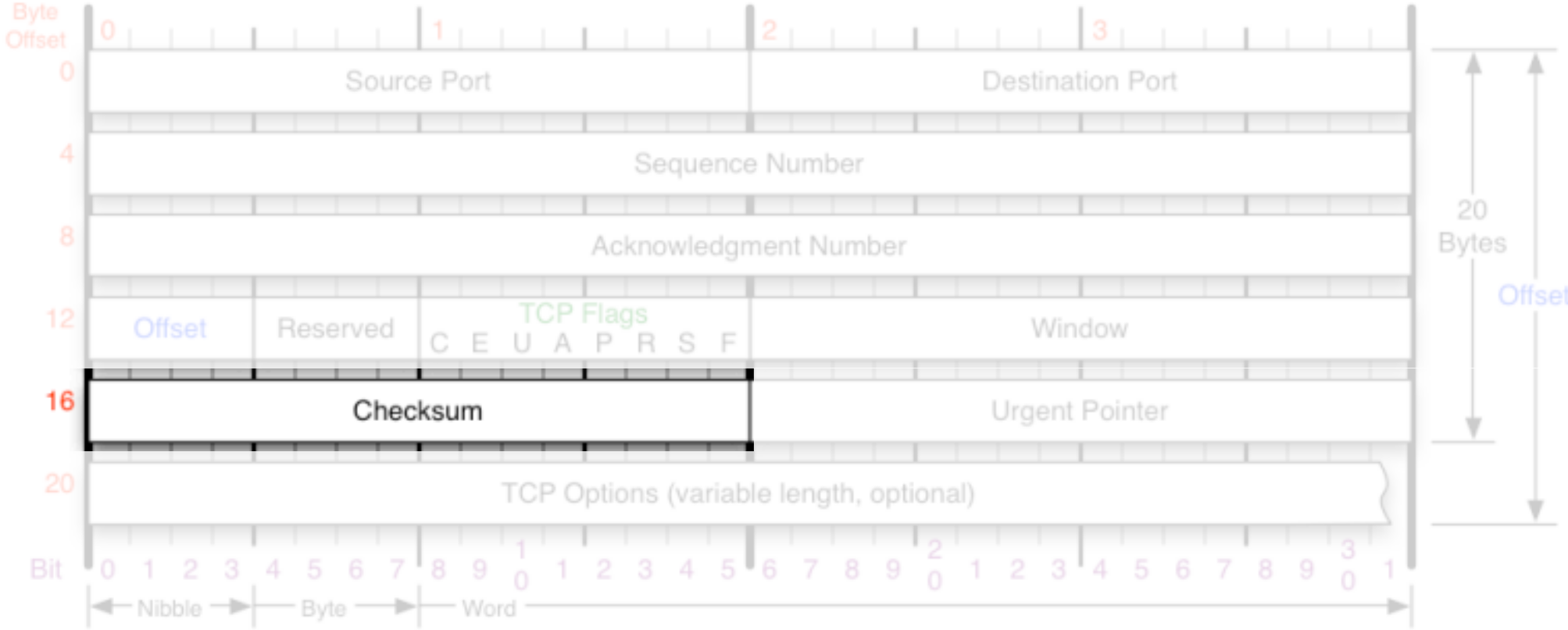
TCP Security Properties

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

- Does TCP ensure Integrity and Authenticity against a man-in-the-middle attacker?

TCP Checksum

- One's complement sum of pseudo IP header, TCP header and data must be zero
  - Choose checksum value to make this so



TCP Checksum

IP Pseudo Header

TCP Header

Segment data

TCP pseudo-header for checksum computation (IPv4)

Bit offset	0–3	4–7	8–15	16–31
0	Source address			
32	Destination address			
64	Zeros		Protocol	TCP length
96	Source port			Destination port
128	Sequence number			
160	Acknowledgement number			
192	Data offset	Reserved	Flags	Window
224	Checksum			Urgent pointer
256	Options (optional)			
256/288+	Data			

TCP Security Properties

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

- Does TCP ensure Integrity and Authenticity against a man-in-the-middle attacker? ✗
  - Attacker can update checksum after modifying packet

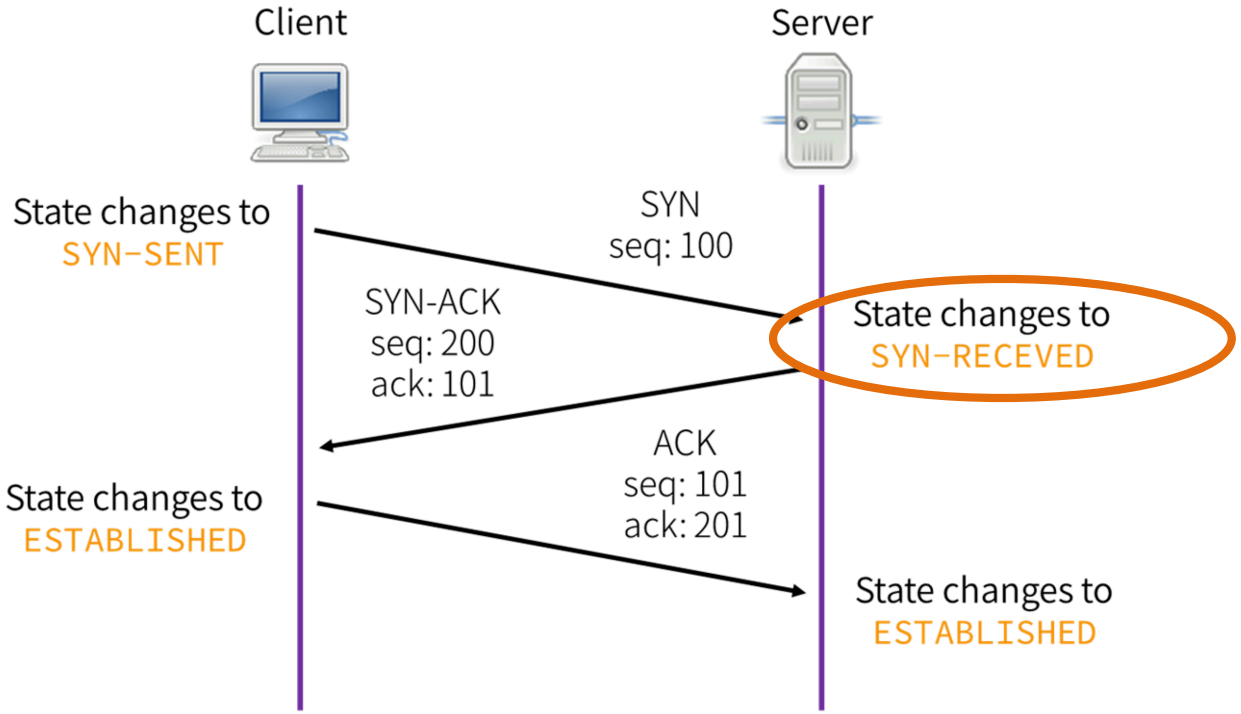
TCP Security Properties

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

- Does TCP have the same problems with availability?
- Does TCP provide any assurance of authenticity?

Attacking TCP Handshake

- Server allocates a connection record called a Transmission Control Block (TCB) when it receives SYN from client — why?

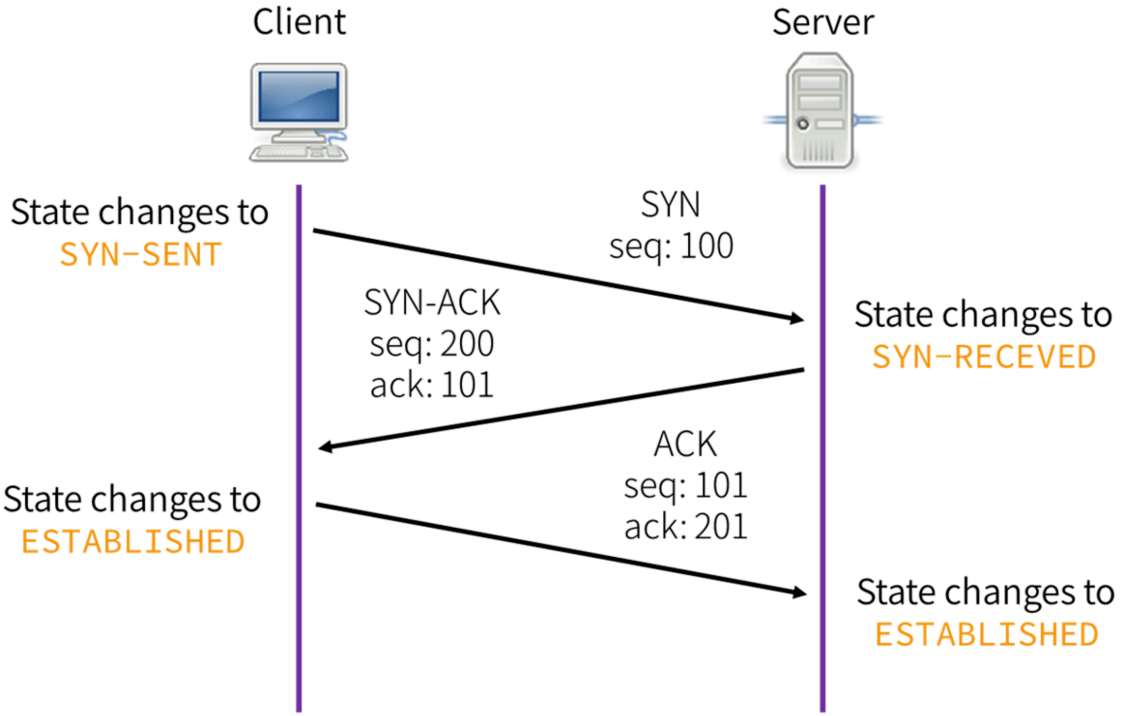


Attacking TCP Handshake

- Server allocates a connection record called a Transmission Control Block (TCB) when it receives SYN from client
- Operating systems limited the number of TCBs
  - Once all TCBs allocated, new requests ignored
- *SYN flooding denial-of-service attack:* attacker sends SYN packets, causing server to use up all TCBs: other clients prevented from connecting

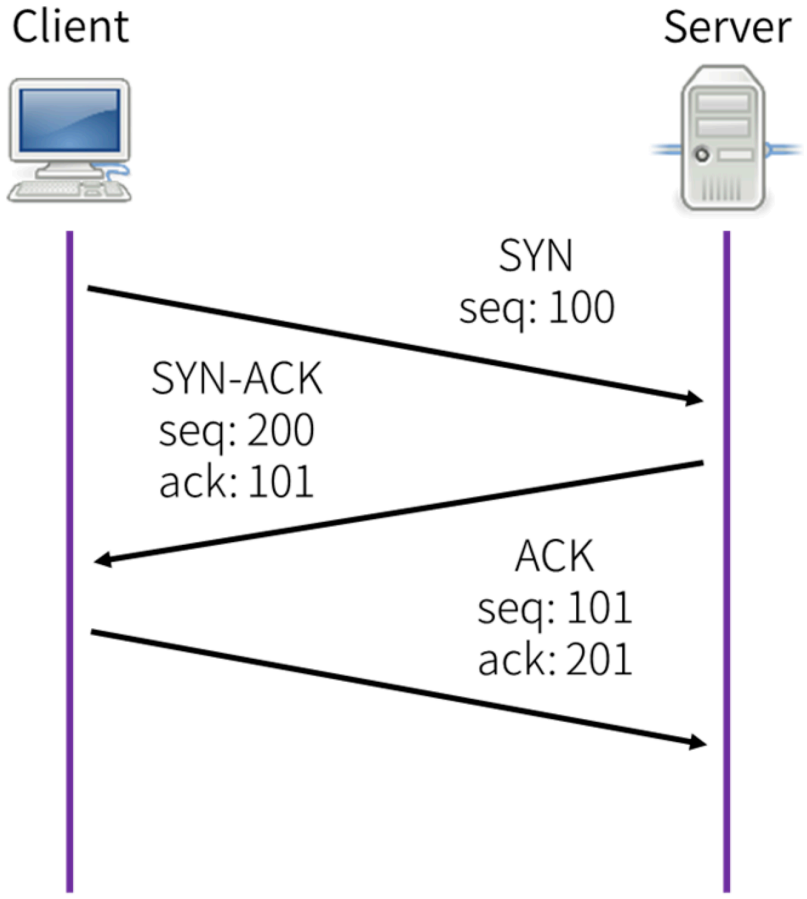
TCP SYN Cookies

- *Do we need to allocate any resources in response to a SYN packet from client?*
  - *What information does server need to know?*



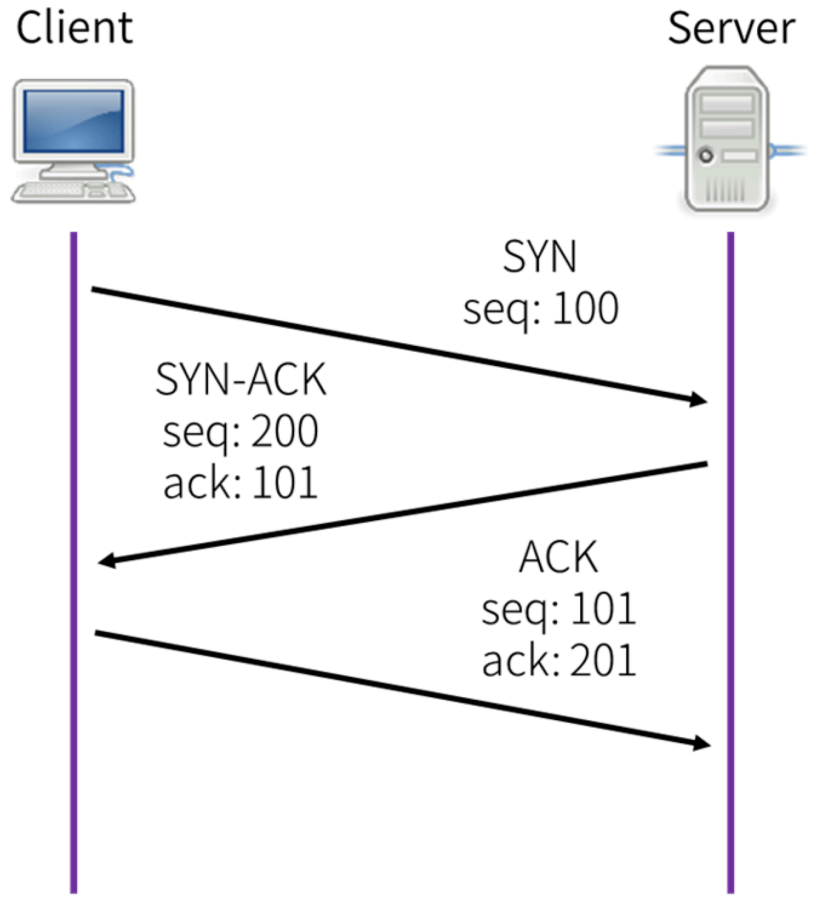
TCP SYN Cookies

- Most of the TCB data appears in next packet from client
  - IP addresses, port numbers, sequence numbers
- Can we wait to allocate TCB until second packet?



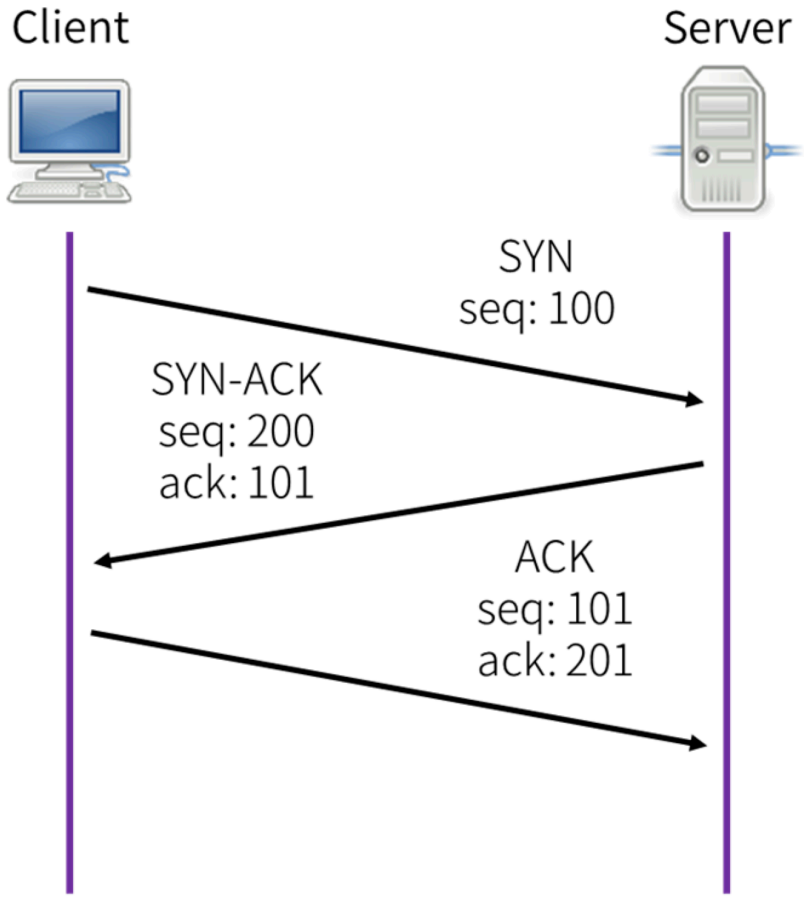
TCP SYN Cookies

- Most of the TCB data appears in next packet from client
  - IP addresses, port numbers, sequence numbers
- Can we wait to allocate TCB until second packet?
- *Not quite:* attacker can flood fake second packet
  - "ACK flooding?"



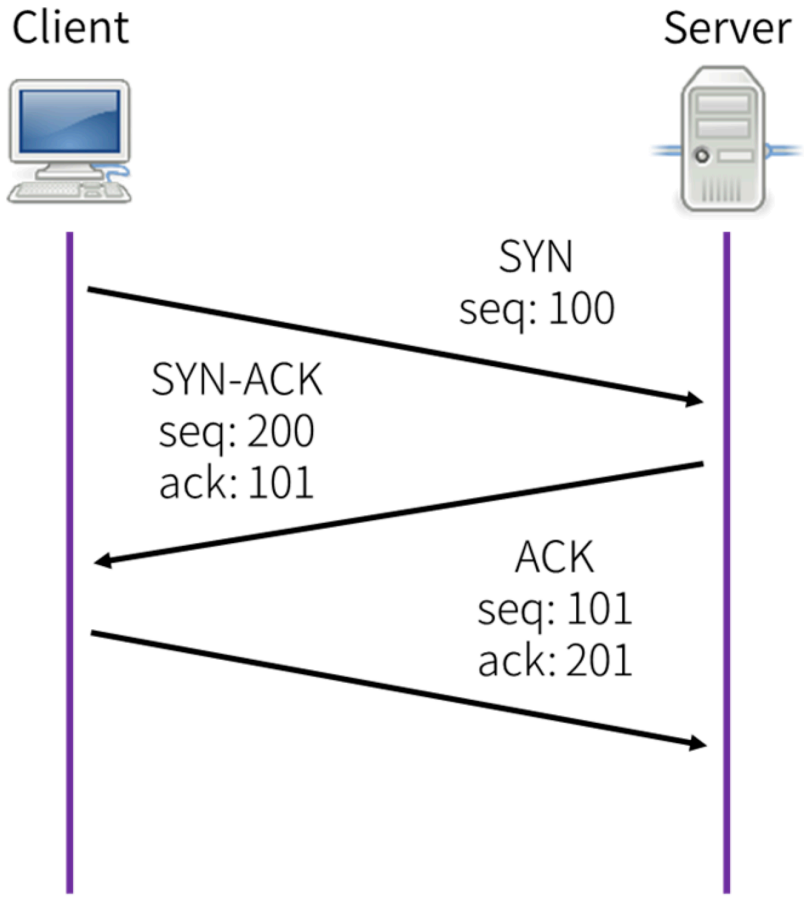
TCP SYN Cookies

- Can we verify that client sent SYN packet without using any server-side memory?
- 32-bit initial sequence number can be chosen arbitrarily
- Use lower 32 bits of $MAC_k(TCB)$ as sequence number
  - MAC is Message Authentication Code, not Ethernet MAC
  - k is server secret



TCP SYN Cookies

- Use lower 32 bits of $MAC_k(TCB)$ as sequence number
- When server receives ACK that does not have an associated connection:
  - Re-compute $MAC_k(TCB)$ using data in client's ACK packet
  - Check if acknowledgement number matches lower 32 bits of $MAC_k(TCB)$



TCP SYN Cookies Details

- In practice cookies slightly different
  - Include current time (prevent future replay)
  - Part of 32-bit seq. no. includes data only sent in SYN packet (e.g. maximum segment size)
- A TCP SYN cookies-like mechanism now implemented in most modern OSes
  - Kicks in once maximum TCB limit reached

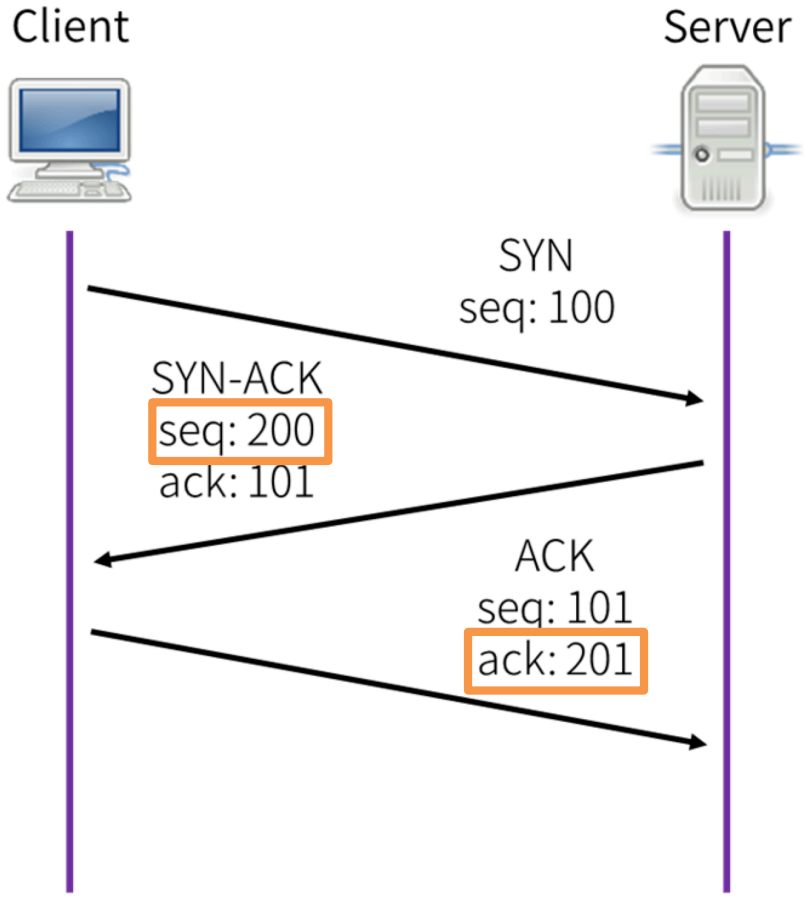
TCP Security Properties

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

- *Without SYN cookies*, TCP particularly vulnerable
  - Used by Mitnick in Christmas Day attack
    - But you will not need to do this for the MP!
- *With SYN cookies*, TCP no worse than IP
  - Denial of service via link capacity exhaustion

TCP Connection Spoofing

- Can we impersonate another host when *initiating* a connection?
- Off-path attacker can send initial SYN to server …
  … but cannot complete three-way handshake without seeing the server's sequence number
- Probability of success 2^{-32} if initial sequence number chosen uniformly at random



TCP Reset Attack

- Can we reset an *existing* TCP connection?
- Need to know port numbers (16 bits)
  - Initiator's port number usually chosen random by OS
  - Responder's port number may be well-known port of service
- There is leeway in sequence numbers B will accept
  - Must be within window size (32-64K on most modern OSes)
- Success probability: $W \times 2^{-(16+32)}$ *(where W is window size)*
  - Maximum value of W is 2^{16}
- See "Slipping in the Window: TCP Reset Attacks" by P. Watson

TCP Challege: ACK Defense

- Only accept a reset if TCP sequence number is exactly equal the last acknowledgement number
- If a RST sequence number does *not* exactly equal the last acknowledgement number but falls inside window, send another ACK with current acknowledgement number
  - Tells valid RST sender what correct sequence number is
  - Would not be observable by off-path attacker
- See RFC 5961 for details

TCP Connection Hijack

- Can we impersonate another host in *existing* TCP connection?
  - E.g. to insert data into the data stream?

TCP Connection Hijack

- Can we impersonate another host in *existing* TCP connection?
  - E.g. to insert data into the data stream?
- Say we want to impersonate *A* to *B* in existing connection
- Need to know port numbers (16 bits)
  - Initiator's port number usually chosen random by OS
  - Responder's port number may be well-known port of service
- B will accept sequence numbers inside window
- B will accept ack. numbers in correct half of 32-bit seq. space
- $W \times 2^{-(16+32+1)}$ (where W is window size) chance to guess right
  - Maximum value of W is 2^{16}
- Protocol must tolerate misaligned data for attack to work

TCP Hijack Defenses

- Limit range of acceptable acknowledgement numbers up to maximum window size behind last sent data
- About $W^2 \times 2^{-(16+32+32)}$ chance to guess right
  - Maximum value of W is 2^{16}
- Can also introduce application protocol checks

TCP Security Properties

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

- TCP (with modern defenses) reduces success probability of off-path attacks to acceptable levels
  - About 2^{-48} for hijack, 2^{-32} for spoofing
- Initiator port number and all sequence numbers must be chosen uniformly at random!

User Datagram Protocol

- Sometimes we *do* only want best-effort delivery
- **User Datagram Protocol (UDP)** is a transport layer protocol that is essentially a wrapper around IP
  - Adds ports to demultiplex traffic by applications
- Checksum similar to TCP
  - Covers IP pseudo-header, UDP header, and data