

Tutorial 4:

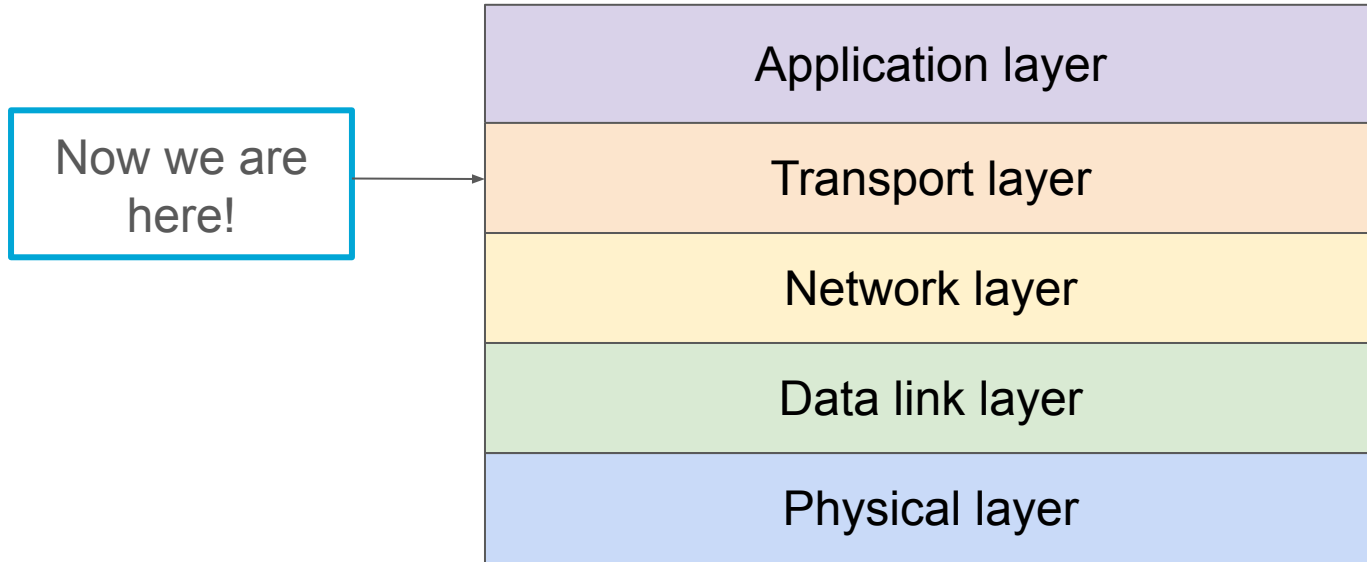
Transport Layer

Konstantinos Timinis

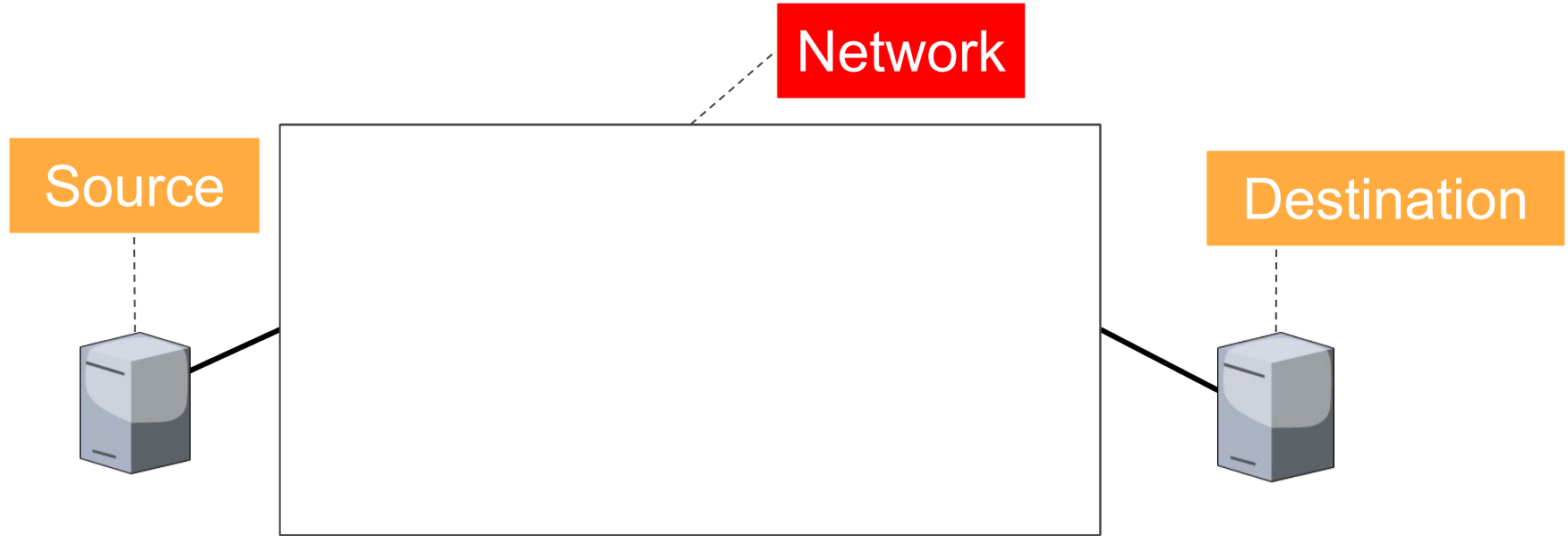
Computer Networks 2025-2026



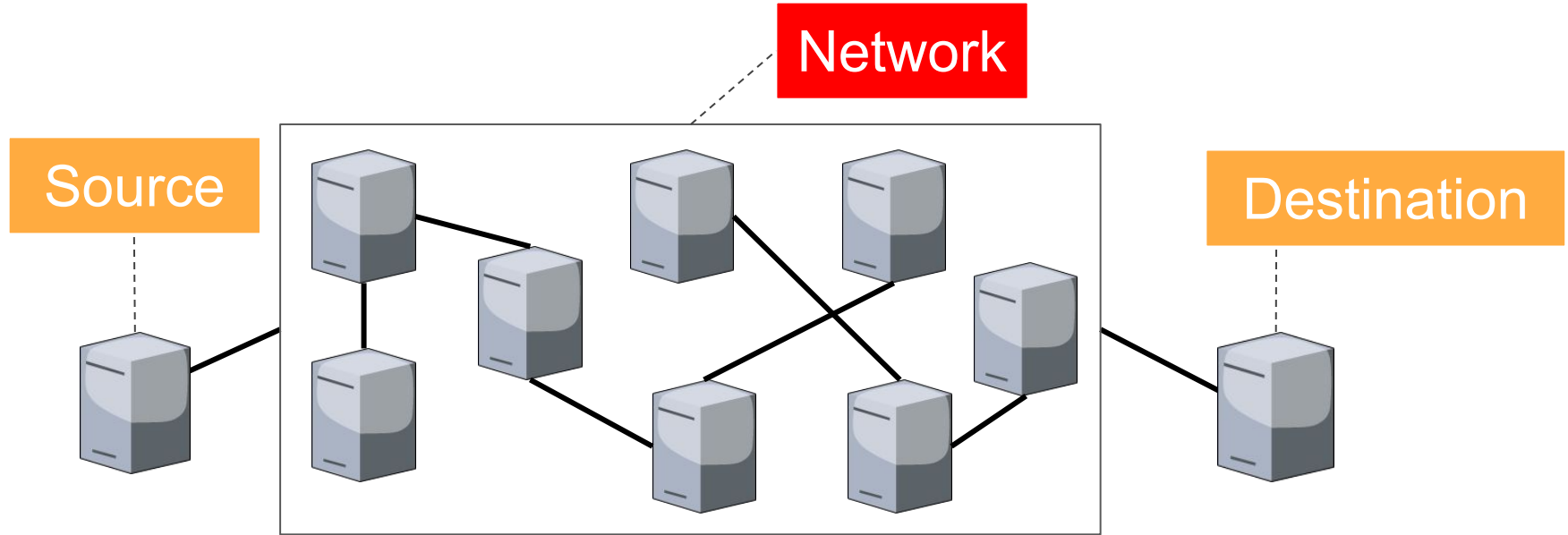
Where Are We Now?



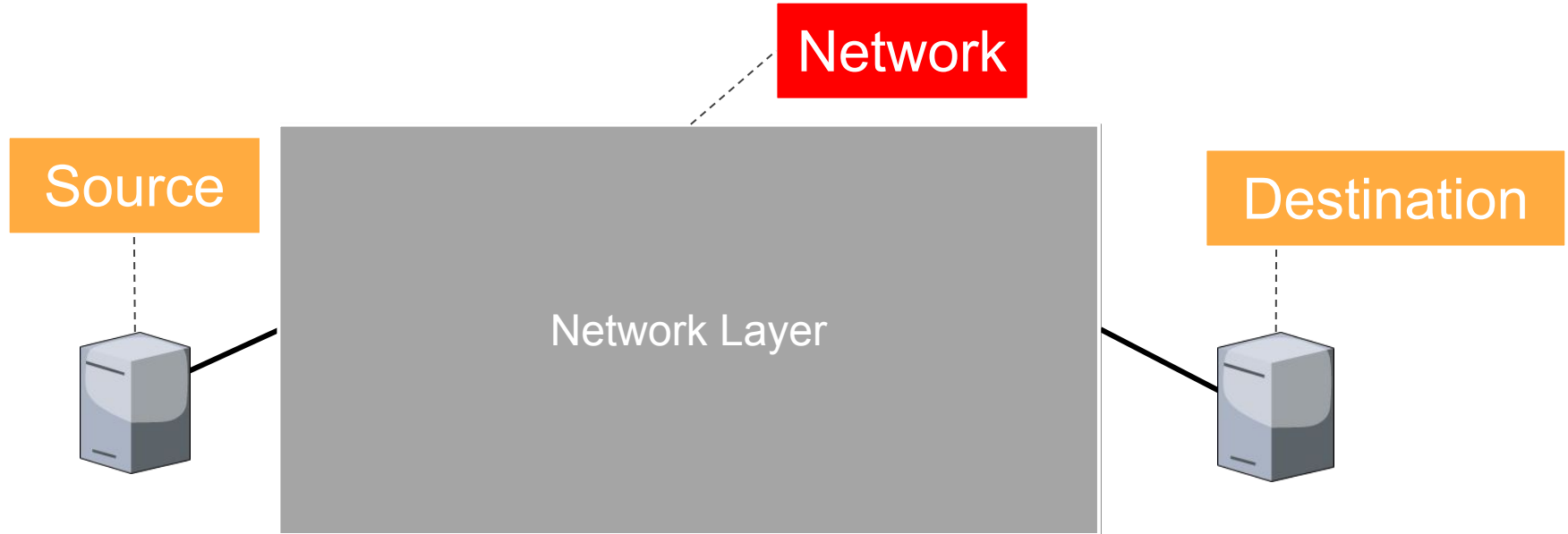
Recap - Why Transport Layer?



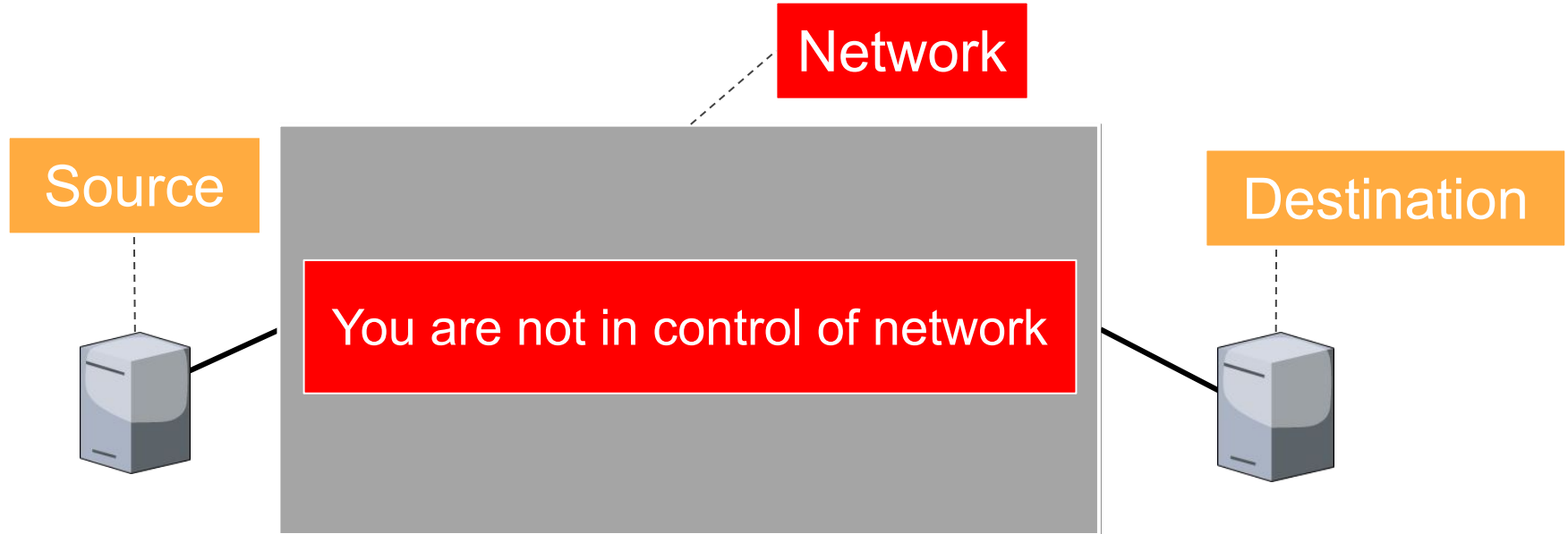
Recap - Why Transport Layer?



Recap - Why Transport Layer?



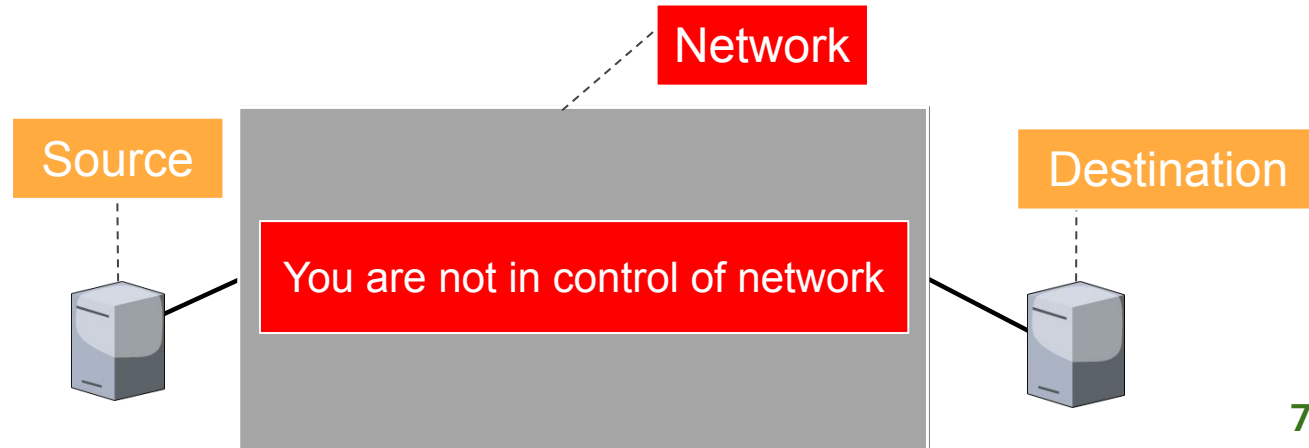
Recap - Why Transport Layer?



Recap - Why Transport Layer?

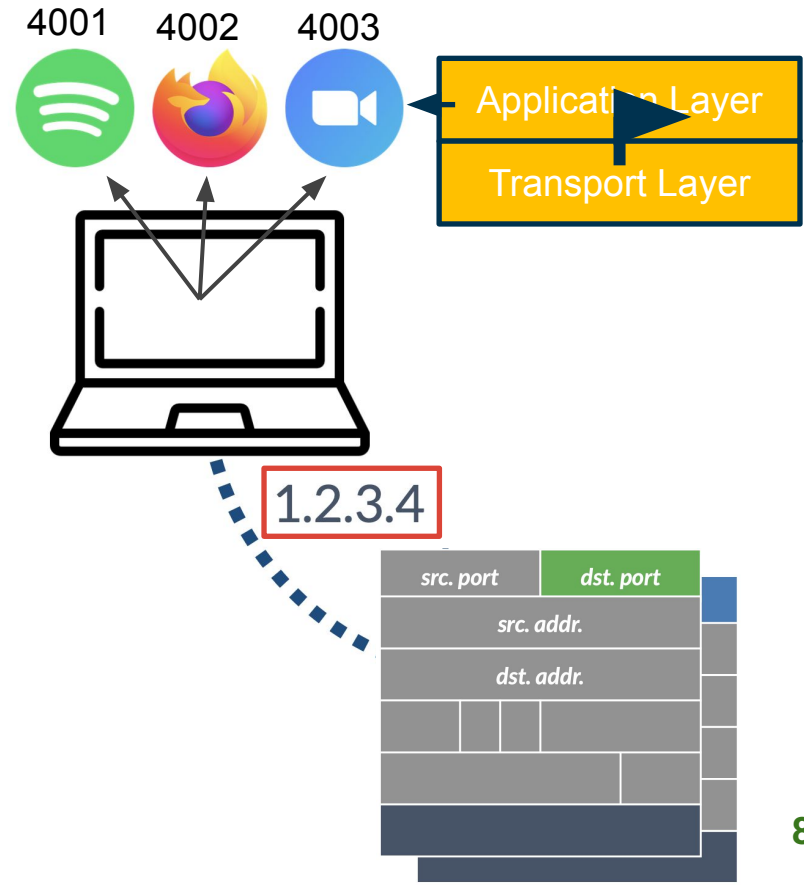
You do not know if:

1. Data will be lost
2. Data will be damaged
3. What is network maximum capacity



Recap – Port Numbers

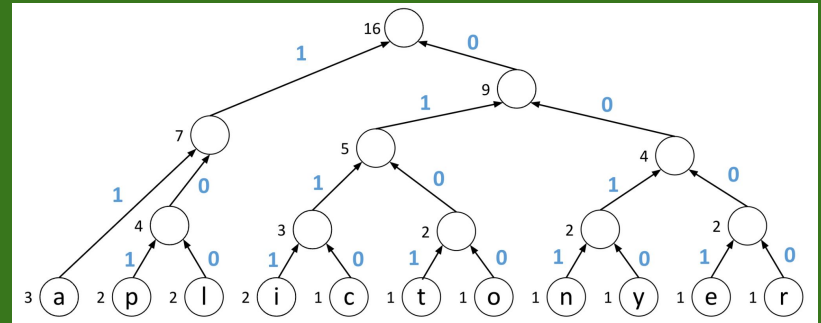
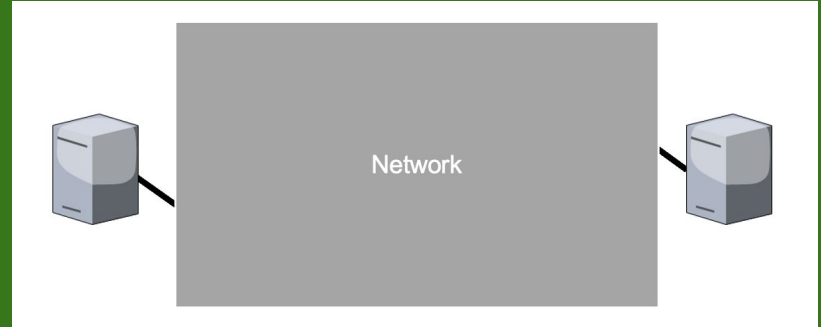
- We have **one** end host, but **multiple** applications.
- All applications are running on the **same** IP address.
- Different applications run on different **ports**.
- Each **segment** has a **source** and **destination** port number.



Menu Of The Day

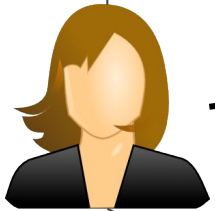
1. Transport Layer
 - a. Sequence Numbers
 - b. Fast Retransmission
 - c. Congestion Control

2. Application Layer Encodings
 - a. Base64 Encoding
 - b. Huffman Encoding

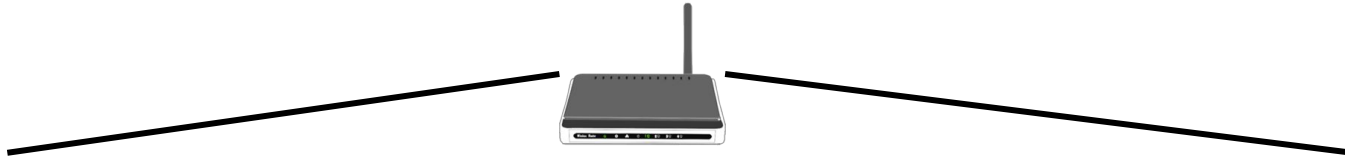
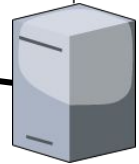


Recap - Sequence Numbers

Source



Destination



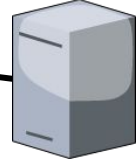
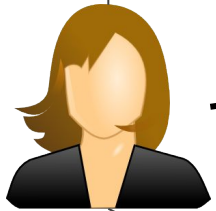
I want to send 4 packets

Recap - Sequence Numbers

Available Sequence Numbers: {0, 1}

Source

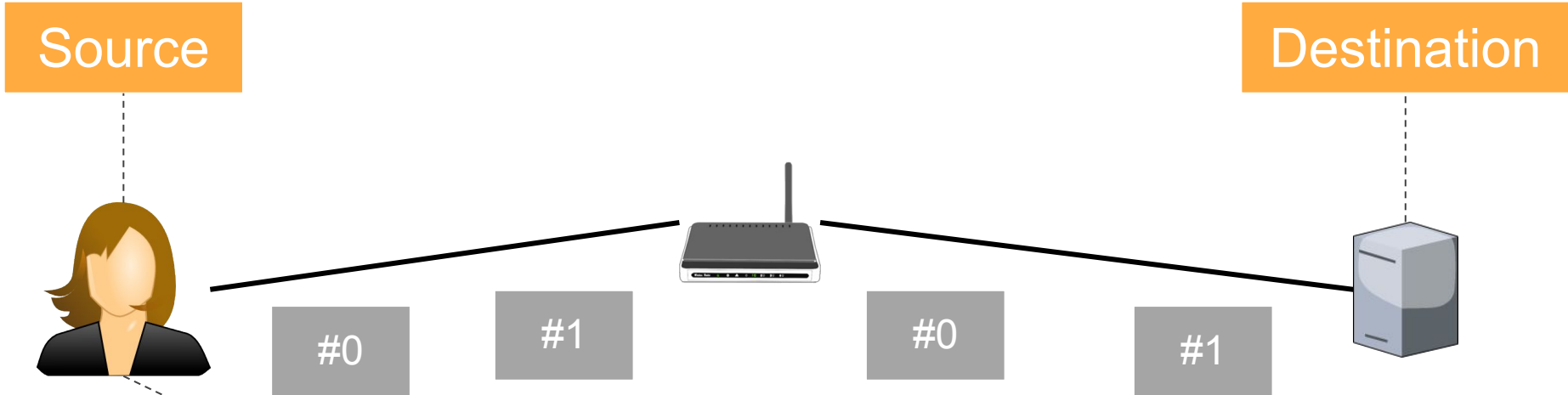
Destination



I want to send 4 packets

Recap - Sequence Numbers

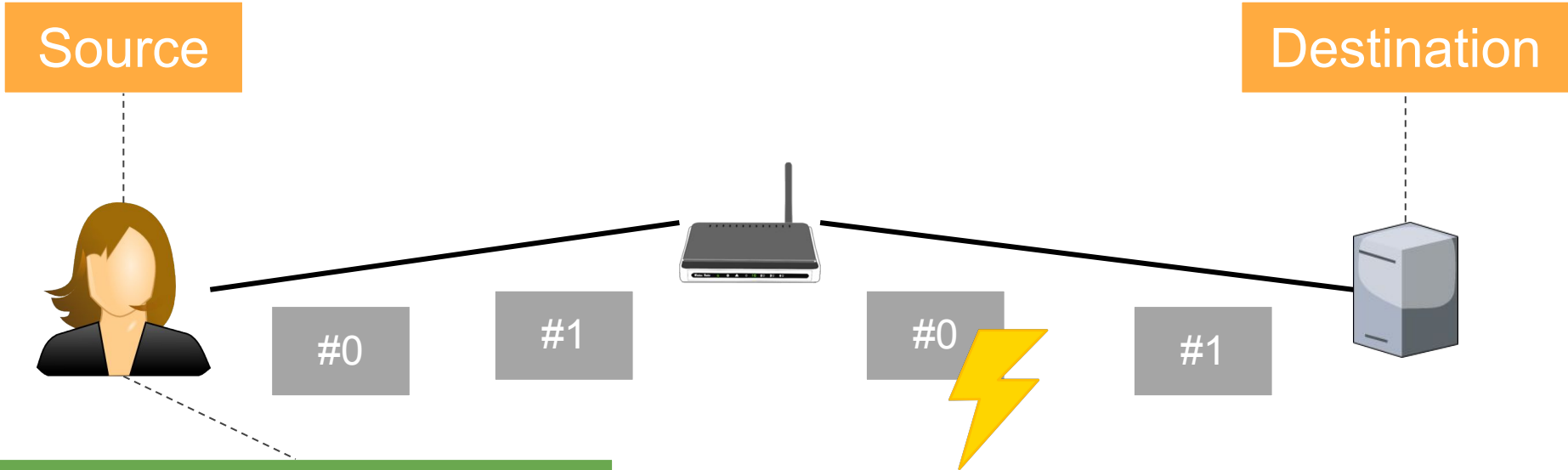
Available Sequence Numbers: {0, 1}



I want to send 4 packets

Recap - Sequence Numbers

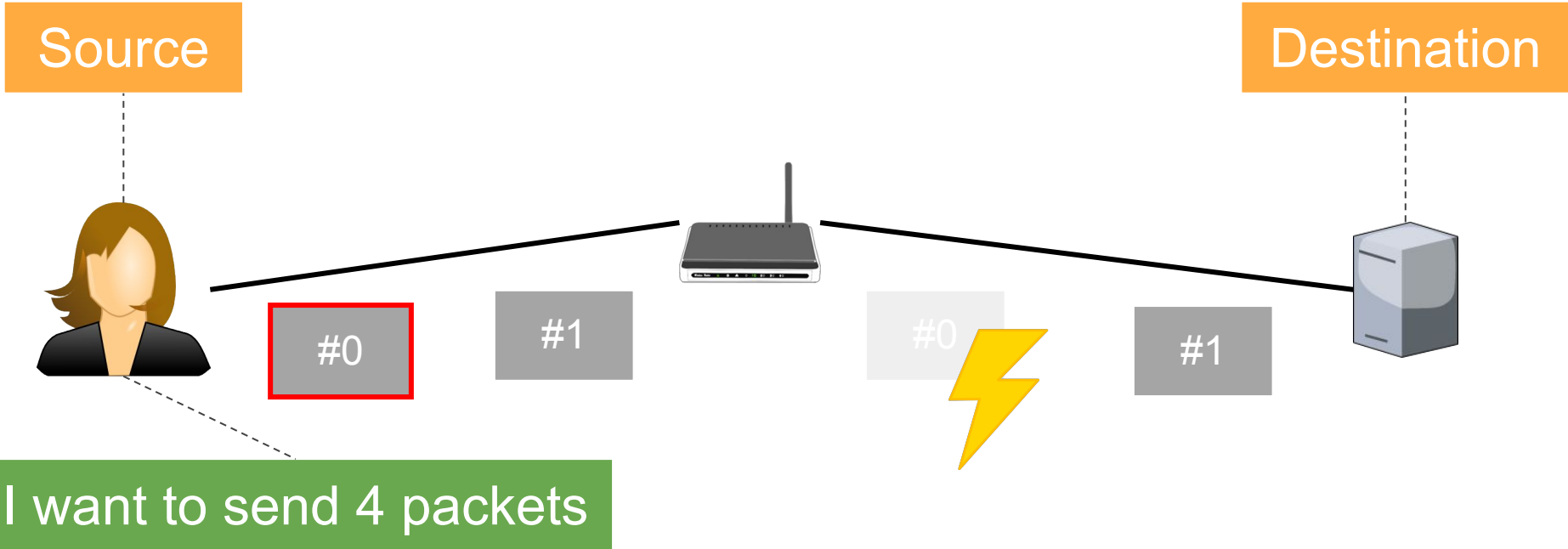
Available Sequence Numbers: {0, 1}



I want to send 4 packets

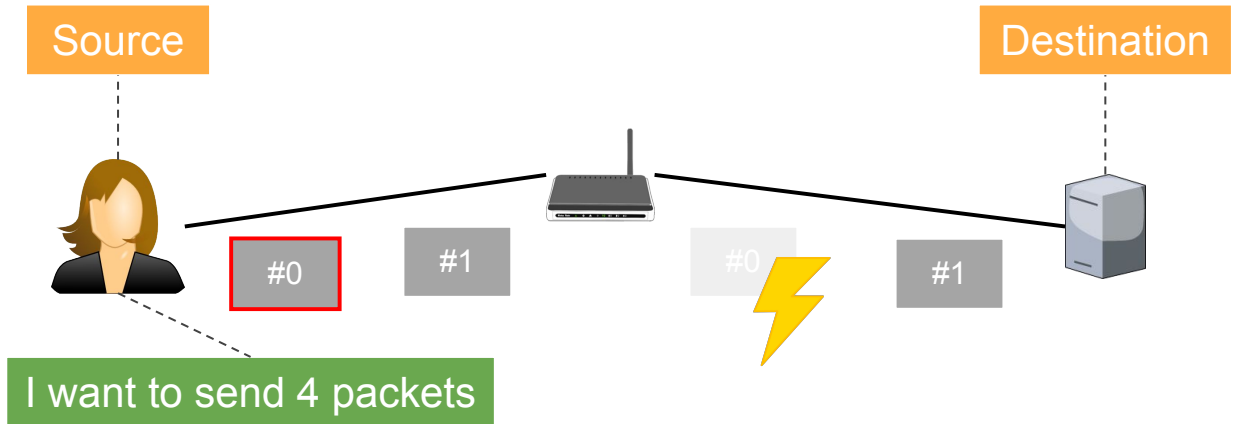
Recap - Sequence Numbers

Available Sequence Numbers: {0, 1}



Recap - Sequence Numbers

- There should **not** be two different packets on the network with the same sequence number (SN)
- After segment lifetime it is safe to reuse sequence numbers



Exercise 1 - World Wrapping Record

In a network with a maximum segment lifetime of **128 seconds**, and a TCP CN protocol that uses **32-bit** sequence numbers **for each byte**, how long does it take for all sequence numbers (SN) to be used when downloading data at a rate of **64 MiB/s**?

Solution - World Wrapping Record

- SN Range = 2^{32}
- Total no. of bytes in segment lifetime = SN Range = 2^{32}

Setup:

- Sequence Numbers (SN): 32 bits
- Download rate: 64 MiB/s
- SN per byte

- Time to use up all = $\frac{\text{SN Range}}{\text{Download rate in Bytes}} = \frac{2^{32}}{2^{20} \times 2^6} = \frac{2^3}{2^2} = 2 = 64 \text{ seconds}$

1MiB = 2^{20} bytes

Answer: 64 seconds

Exercise 2 - Too Fast To Count

In a network with a maximum segment lifetime of **32 seconds**, and a TCP CN protocol that uses **16-bit sequence numbers for each segment**, and has a maximum segment size of **1024 bytes**, what is the maximum mean data rate per connection?

Answer - Too Fast To Count

- SN Range = 2^{16}
- Total no. of segments in segment lifetime = SN Range = 2^{16}

Setup:

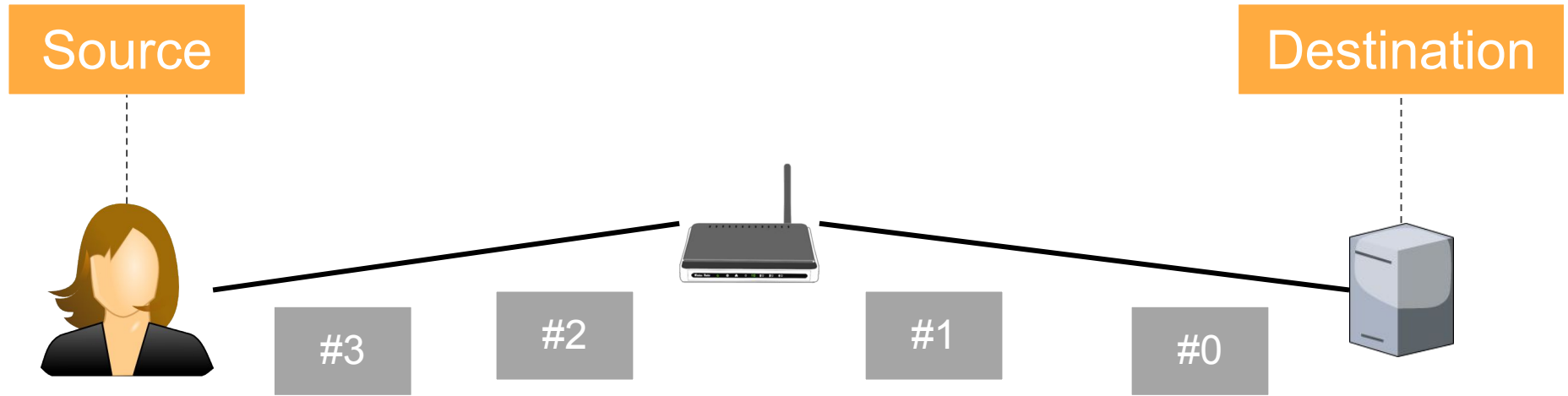
- SN size: 16 bits
- Max segment size: 1024 bytes
- Segment lifetime: 32 seconds
- SN per segment

- Max. amount of data you can transmit in segment lifetime = SN Range \times Max. segment size
 $= 2^{16} \times 1024 = 2^{16} \times 2^{10} = 2^{26} = \mathbf{64 \text{ MiB}}$

- Max. Data Rate = $\frac{\text{max. amount of data you can transmit in segment lifetime}}{\text{segment lifetime}}$
 $= \frac{64 \text{ MiB}}{32 \text{ seconds}} = \mathbf{2 \text{ MiB/s}}$

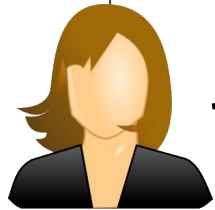
Answer: 2 MiB / s

Recap – Fast Retransmission (FR)



Recap – Fast Retransmission (FR)

```
if sameAckReceivedNTimes():
```



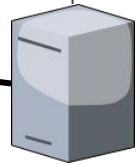
#3

#2



#1

#0



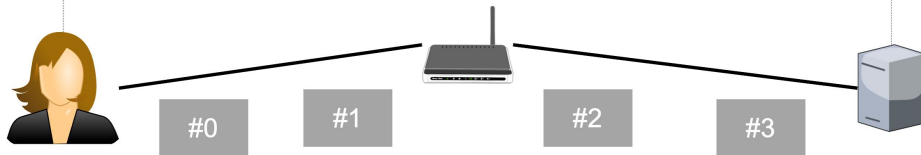
```
if packet.received():  
    if expectedNum:  
        ack(expectedNum)  
    else:  
        ack(lastReceived)
```

Example – Fast Retransmission (FR)

Transmit segments 1 to 4:

- Segment 2 (1st try) - lost
- Segment 2 (2nd try) - received
- FR - upon receiving duplicate acknowledgement twice.
- Receiver buffers packets

```
if sameAckReceivedNTimes():  
  retransmit()
```



Sender

Receiver

0: S1 →

← ACK 1

0: S2 →

Duplicate ACK!

"Where is S2?"

0: S3 →

← ACK 1

0: S2 →

← ACK 3

0: S4 →

← ACK 4

Exercise 3 – Fast Retransmission (FR)

Two hosts are communicating using a TCP CN protocol that uses **fast retransmission** by retransmitting segments when receiving **three** duplicate acks.

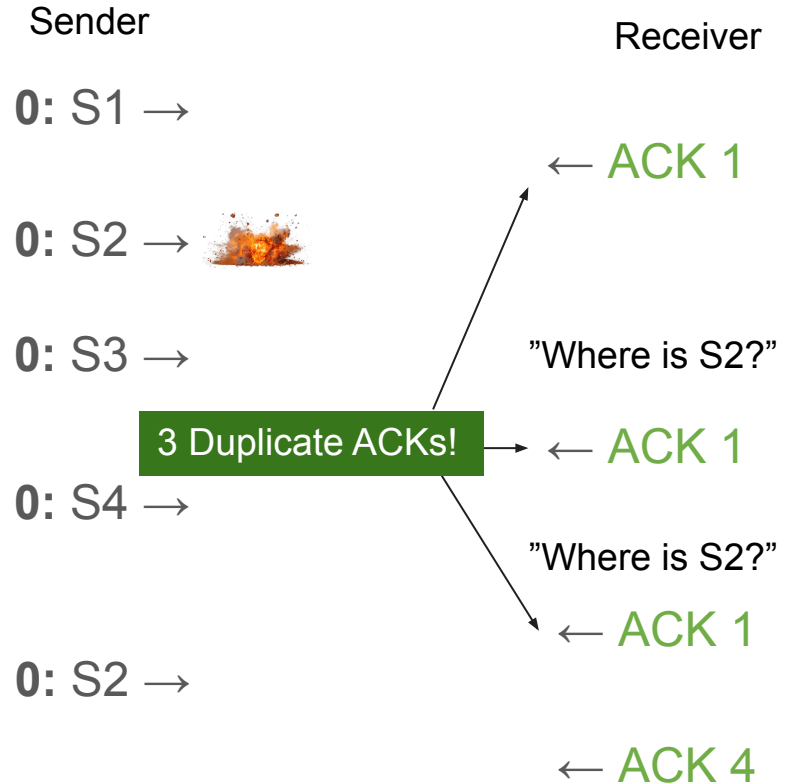
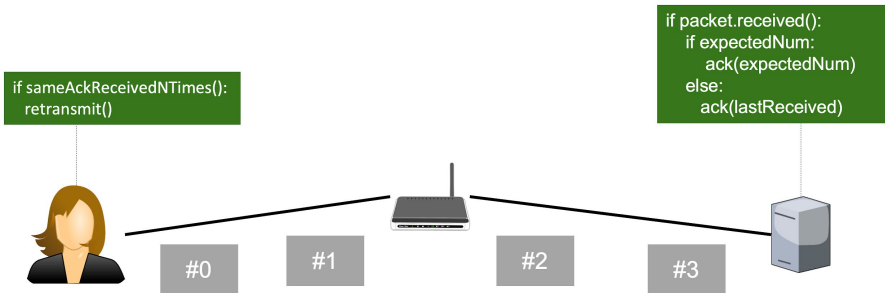
Host A transmits segments S1, S2, S3, S4 in order. Segment S2 is lost and S1, S3, S4 are transmitted successfully.

Assume all acknowledgements are transmitted successfully. Model the transmission behaviour.

Solution – Fast Retransmission (FR)

Setup:

- Retransmit upon 3 duplicate ACKs.
- Segments: S1, S2, S3, S4
- Lost Segments: S2

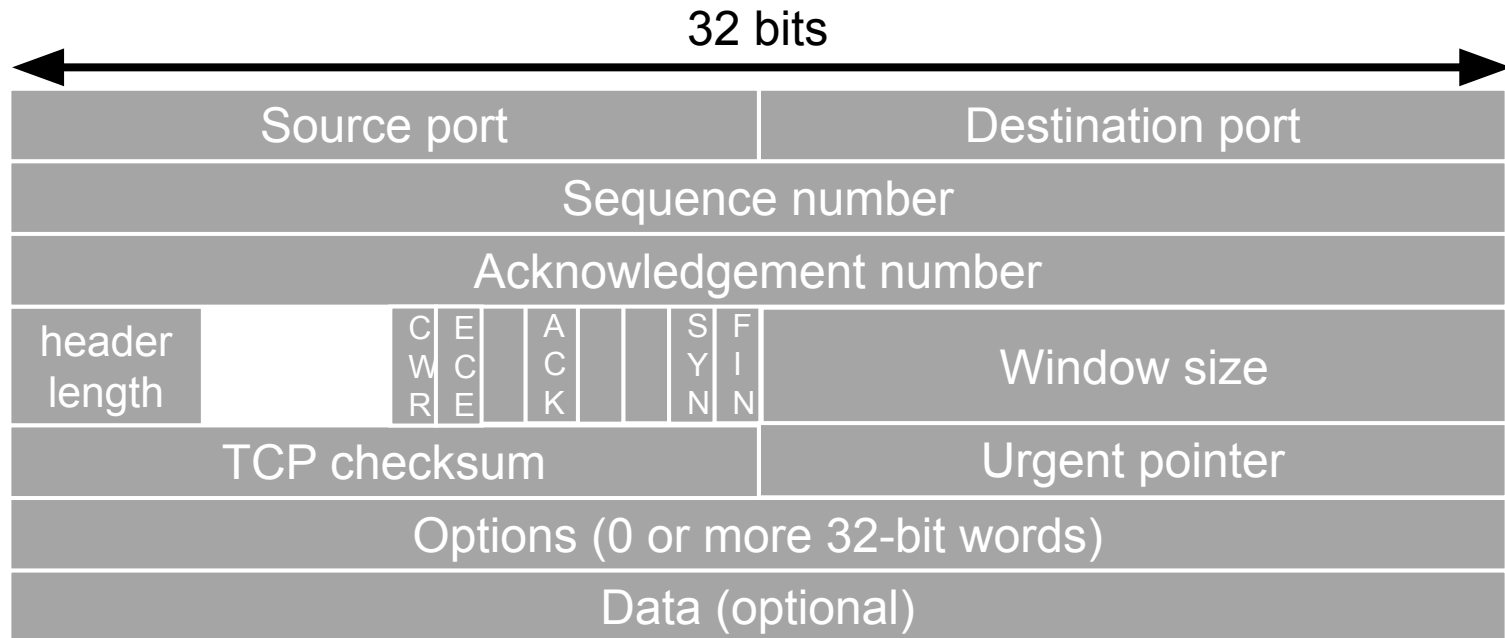


Congestion Control

Recap – TCP Congestion Control



Problem! We do not know the maximum network capacity!

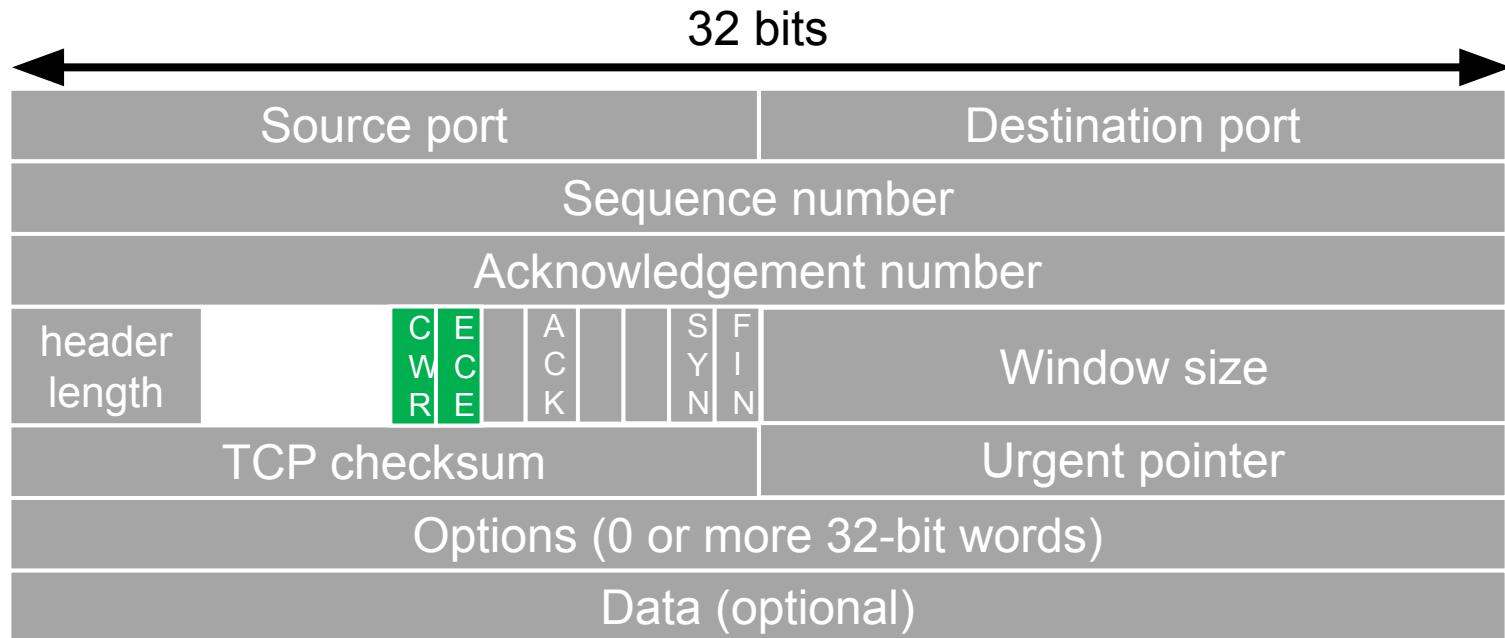


Recap – TCP Congestion Control

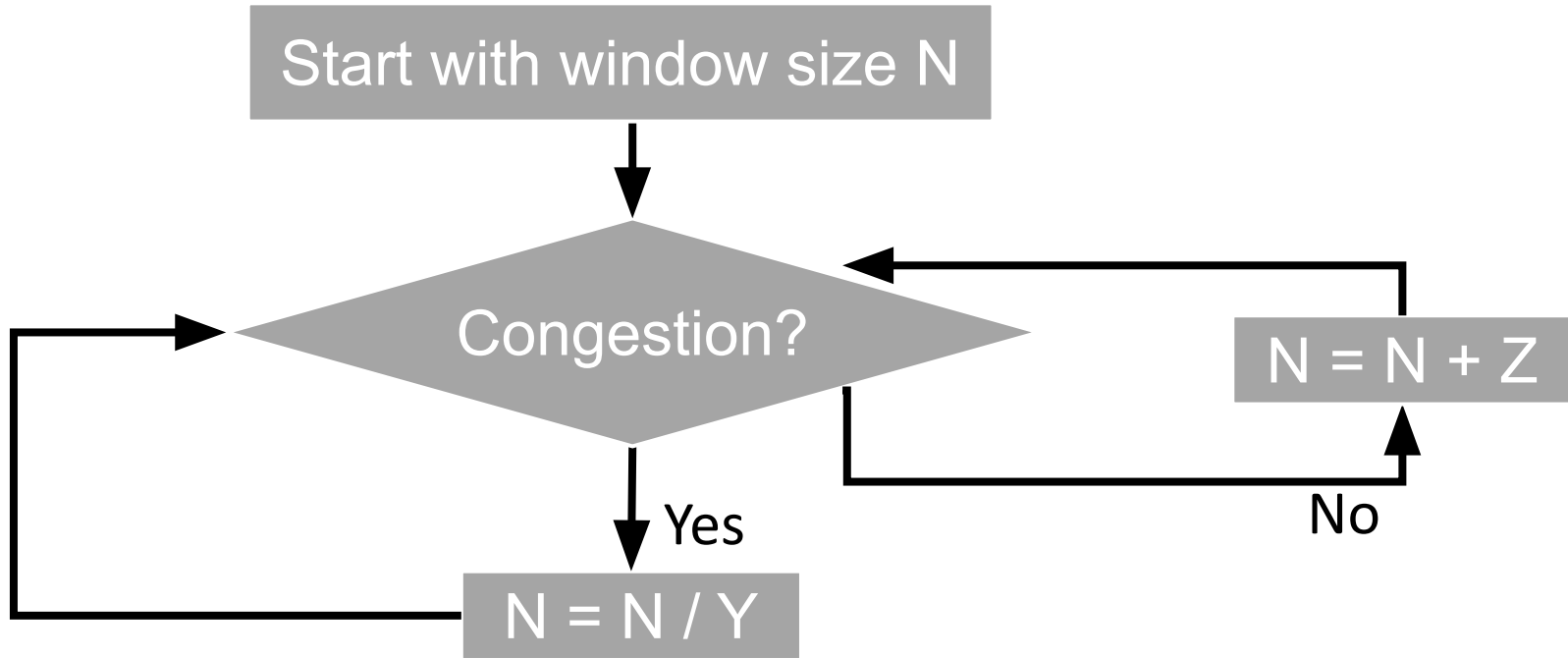


Problem! We do not know the maximum network capacity!

But we do know whether its congested!



Recap – Additive Increase and Multiplicative Decrease

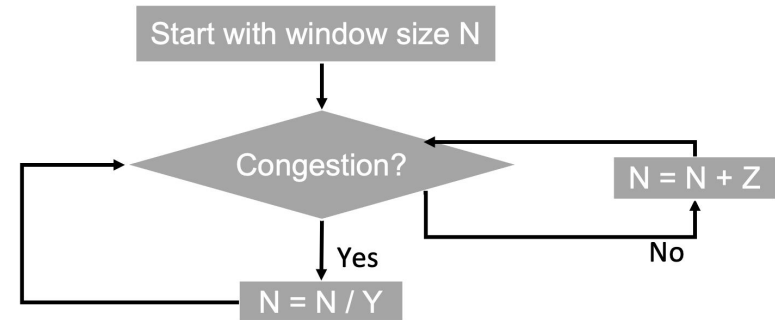


Exercise 4 – TCP CN (AIMD)

An application is sending data using **TCP CN (additive increase, multiplicative decrease (AIMD))** with the following configurations:

1. The **initial** window size is **1 KB**.
2. Additive increase **adds 1 KB** to the window size every round.
3. Multiplicative decrease **divides** window size by **2**.

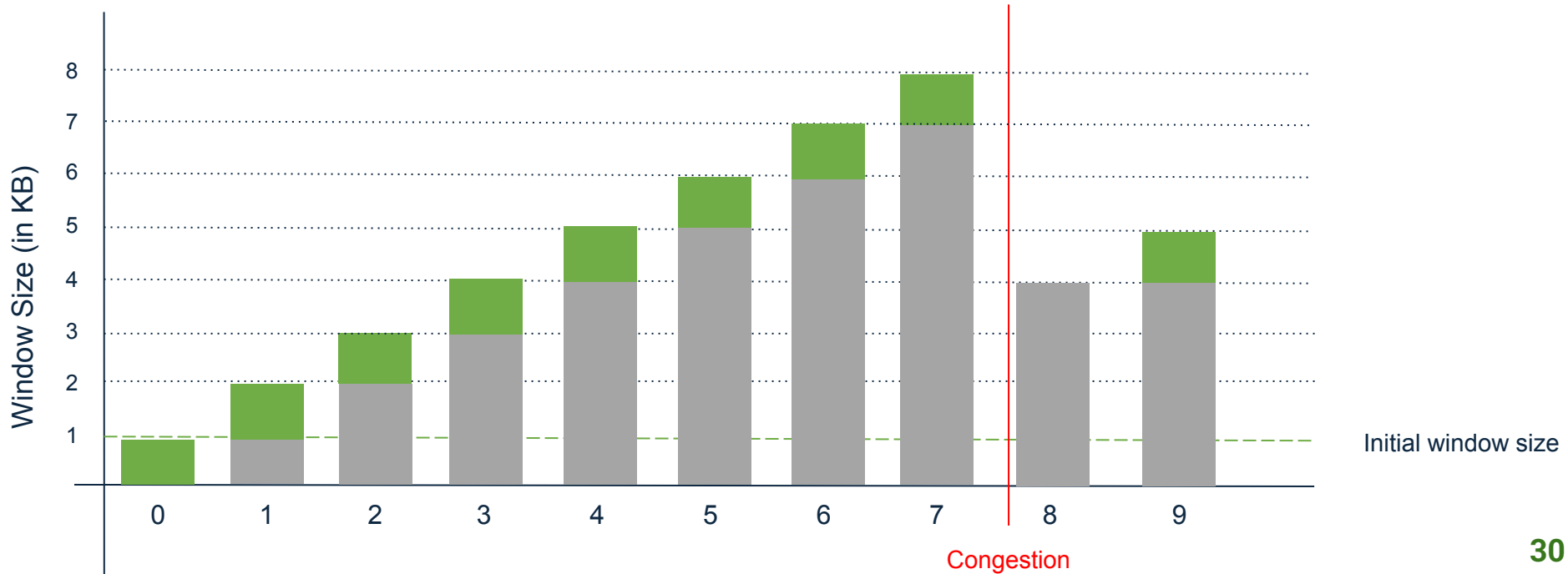
The **congestion** is detected after **round 7**. Model the window size round by round **up to round 9 including**. Start with **round 0** being initial window size and round 1 is the result of round 0 + additive increase.



Solution – TCP CN (AIMD)

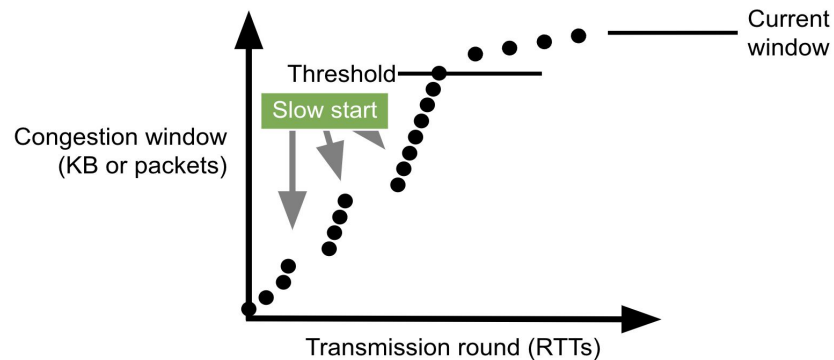
Legend

- Window from previous round
- Added window in the current round



Recap – Improvements TCP Tahoe

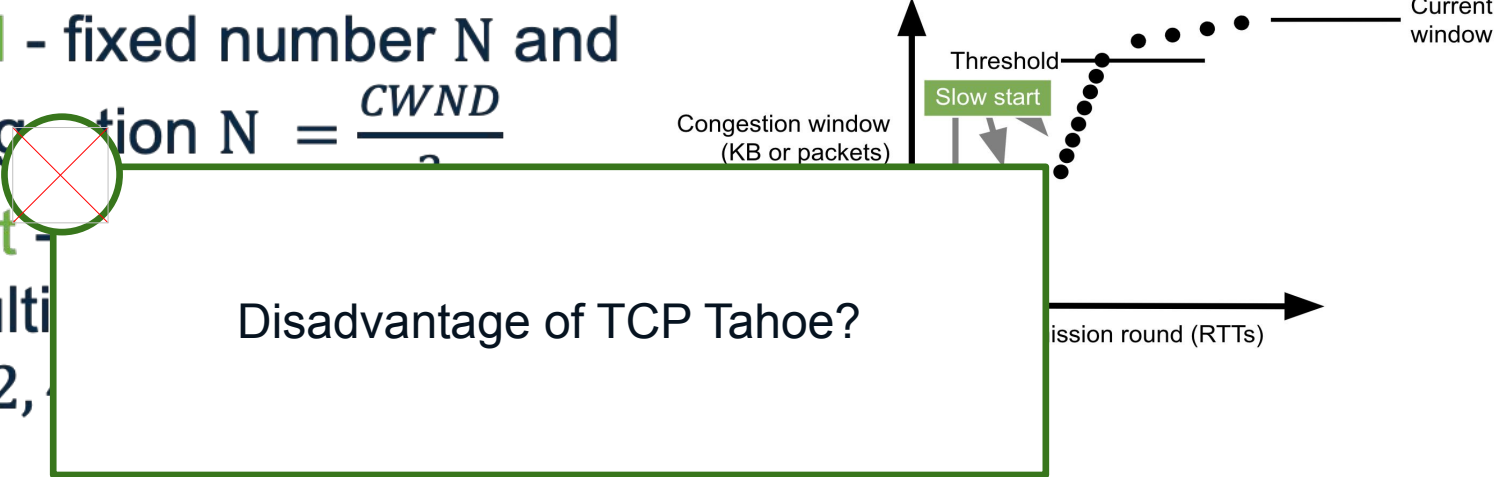
- + **threshold** - fixed number N and after congestion $N = \frac{CWND}{2}$
- + **slow start** - up to threshold apply multiplicative increase by 2. Ex: 1, 2, 4, 8.



* CWND = Congestion Window

Recap – Improvements TCP Tahoe

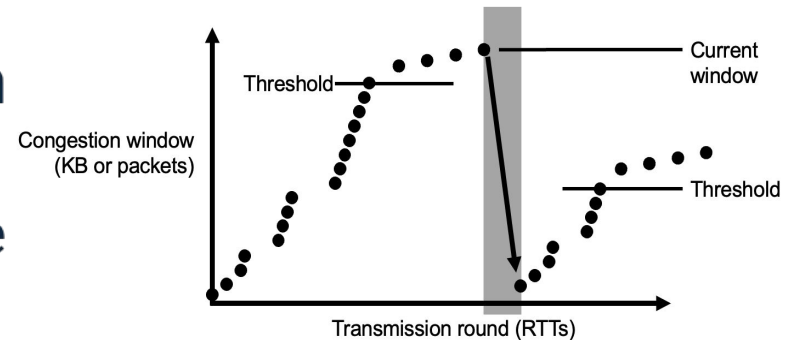
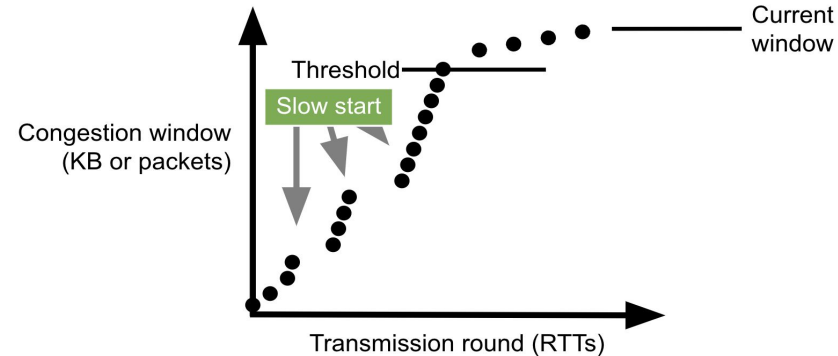
- + **threshold** - fixed number N and after congestion $N = \frac{CWND}{2}$
- + **slow start** - apply multiplier of 2. Ex: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024



* CWND = Congestion Window

Recap – Improvements TCP Tahoe

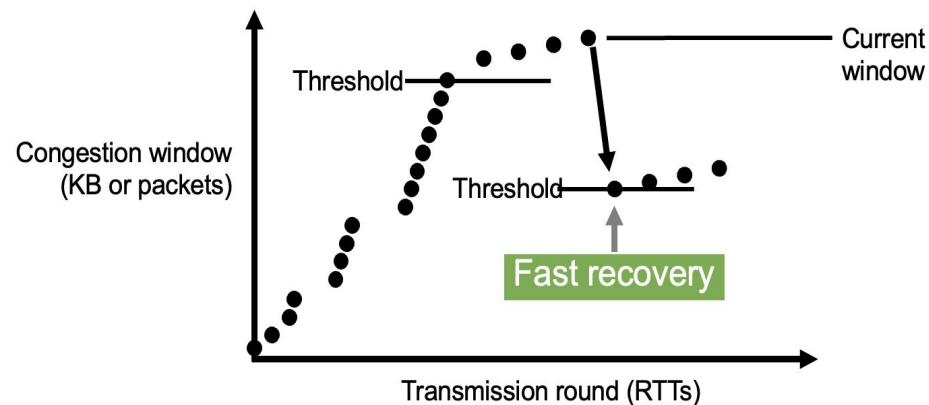
- + **threshold** - fixed number N and after congestion $N = \frac{CWND}{2}$
- + **slow start** - up to threshold apply multiplicative increase by 2. Ex: 1, 2, 4, 8.
- **no fast recovery** - after congestion Tahoe sets window size = initial window size



* CWND = Congestion Window

Recap – Improvements TCP Reno

- + **threshold** - fixed number N and after congestion $N = \frac{CWND}{2}$
- + **slow start** - up to threshold apply multiplicative increase by 2. Ex: 1, 2, 4, 8.
- + **fast recovery** - after congestion: window size = new threshold value



Exercise 5 – Improvements - TCP Reno

An application is sending data using **TCP Reno** (AIMD, fast recovery) with the following configurations:

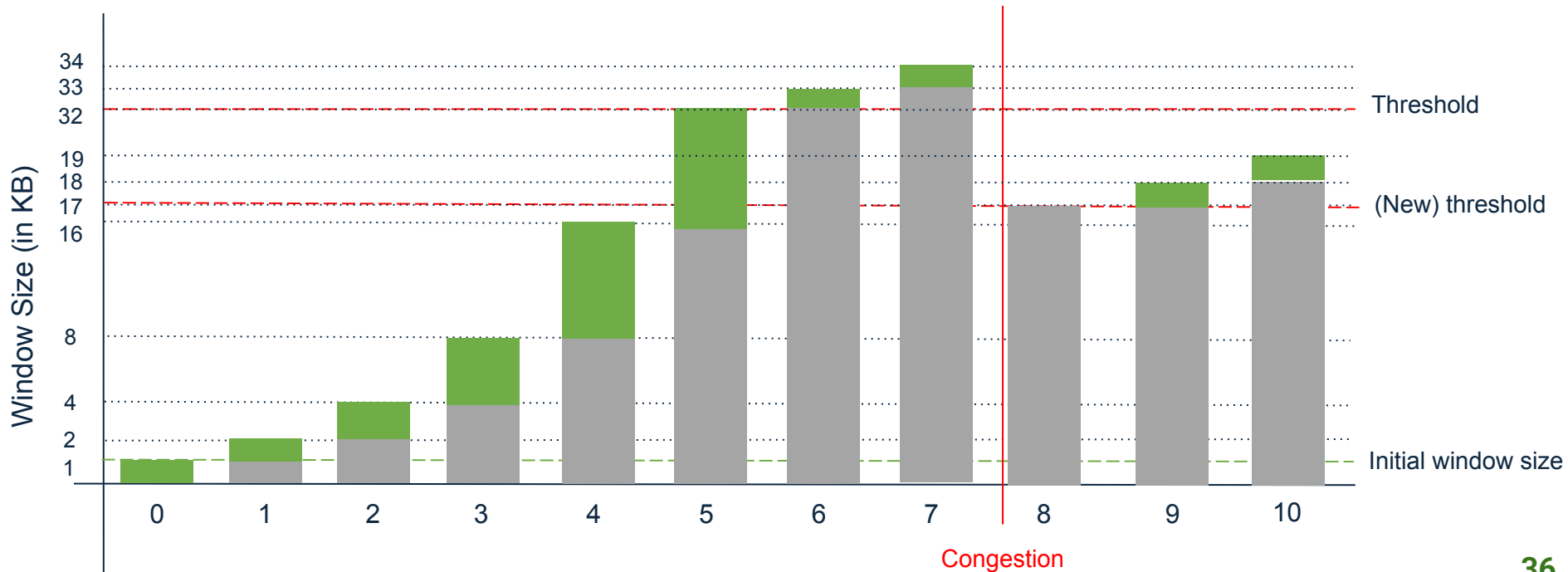
1. The **initial threshold** is set at **32 KB**.
2. The **initial congestion window** size is **1 KB**.
3. Additive increase **adds 1 KB** to the window size every round.

The **congestion** is detected after **round 7**. Model the window size round by round **until round 10**. Start with round 0 and in round 1 is the result of **round 0 + multiplicative increase**.

Solution – Improvements - TCP Reno

Legend

- Window from previous round
- Added window in the current round

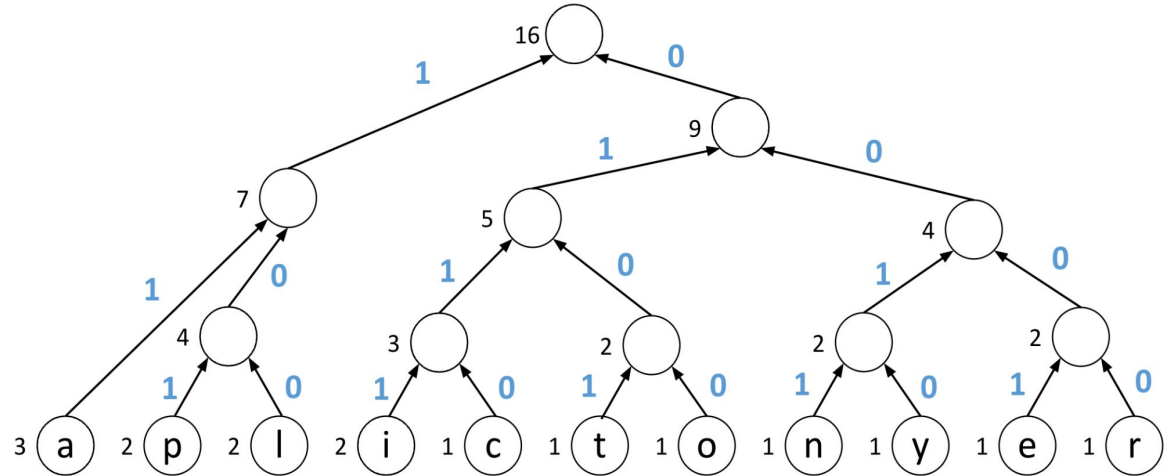


BREAK!

APPLICATION LAYER ENCODINGS

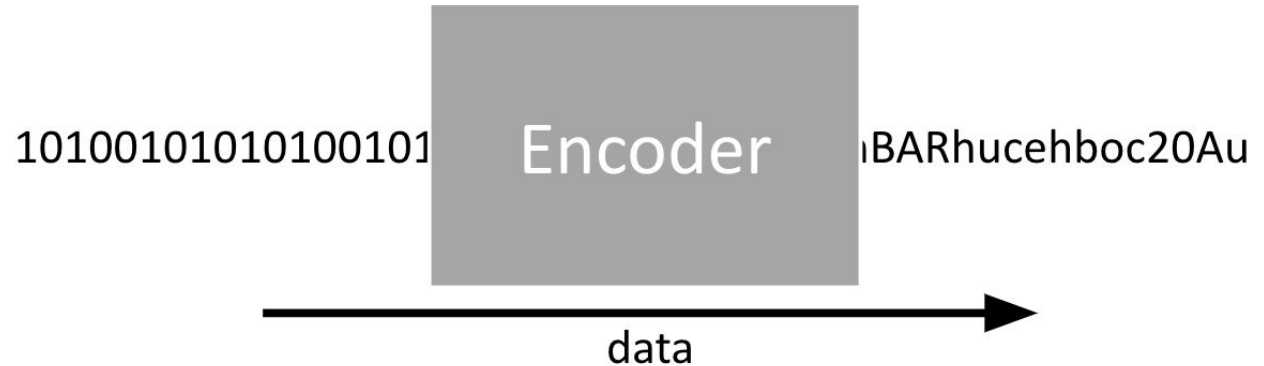
Recap – Why Application Layer Encoding?

- + support of different data types (e.g. images, videos, audio)
- + data compression

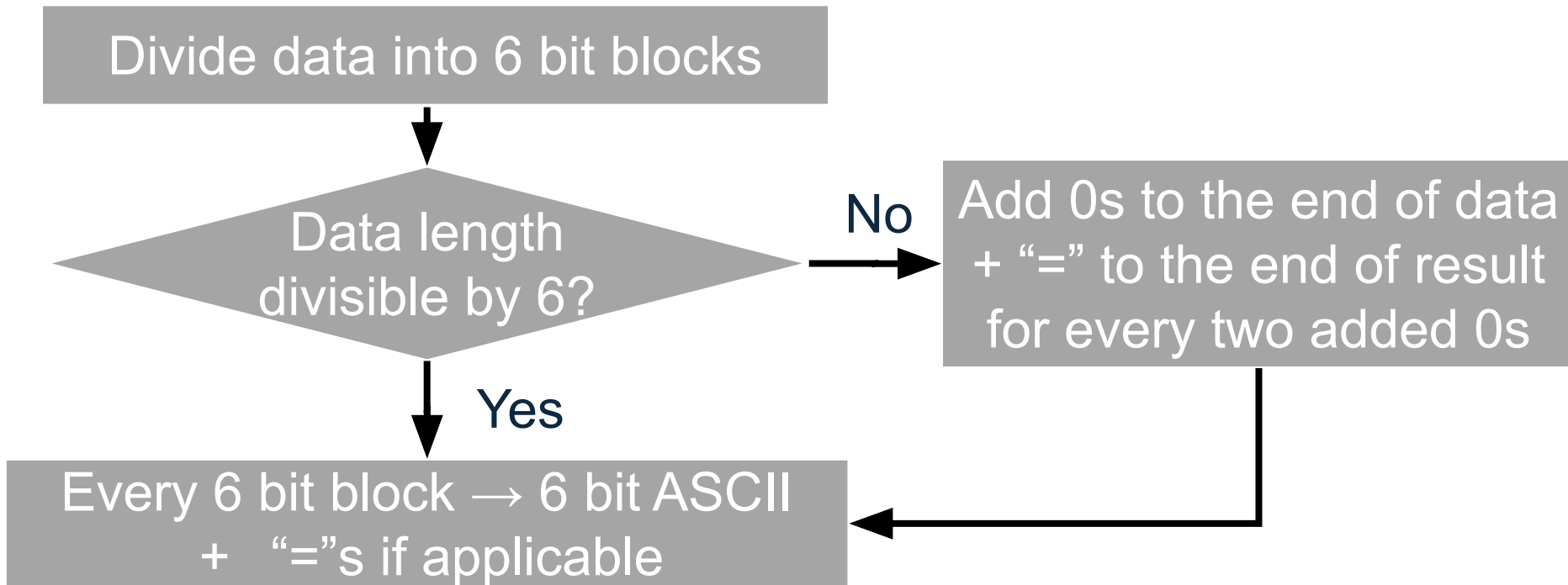


Recap – Base64 Encoding (1/2)

- + multimedia communication
- + on legacy systems designed for ASCII transmission



Recap – Base64 Encoding (2/2)



Exercise 6 – Base64 Encoding

The word “**dogs**” is written in **full ASCII** (table down below).

Encode the word “**dogs**” in **Base64**

- The Base64 alphabet is [A-Za-z0-9+/-].
- E.g.: 000000 → A, 000001 → B, 111111 → /

a	b	c	d	e	f	g	h	i	j	k	l	m
0x61	0x62	0x63	0x64	0x65	0x66	0x67	0x68	0x69	0x6a	0x6b	0x6c	0x6d
n	o	p	q	r	s	t	u	v	w	x	y	z
0x6e	0x6f	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77	0x78	0x79	0x7a

Solution – Base64 Encoding

Setup:

- Encode the word “dogs” in Base64
- The Base64 alphabet is [A-Za-z0-9+/].

d = **0x64** = **0110 0100**

o = **0x6f** = **0110 1111**

g = **0x67** = **0110 0111**

s = **0x73** = **0111 0011**

01100100 01101111 01100111 01110011

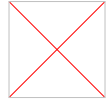
011001 000110 111101 100111 011100 110000

25 6 61 39 28 48

Z G 9 n c w ==

Answer: ZG9ncw==

Recap – Huffman Encoding (1/2)



Lossless compression algorithm.

- + Used in JPEG
- + Enables real time video and audio streaming

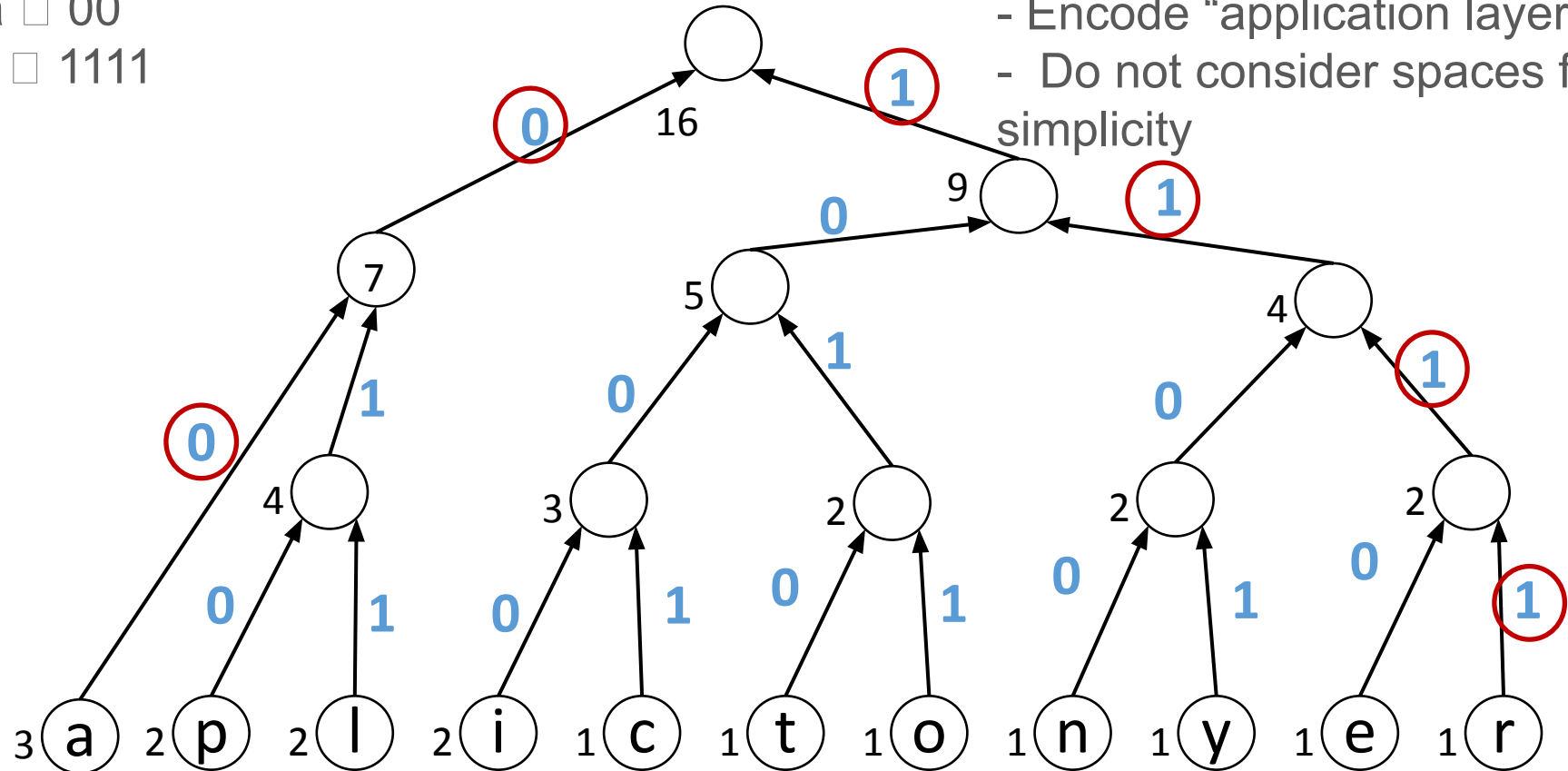
Idea: The more frequent the symbol, the less bits you encode it with



NETFLIX

Recap – Huffman Encoding (2/2)

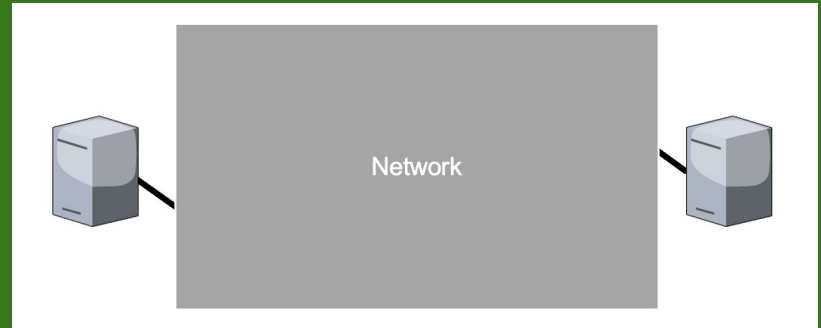
a □ 00
r □ 1111



Overview

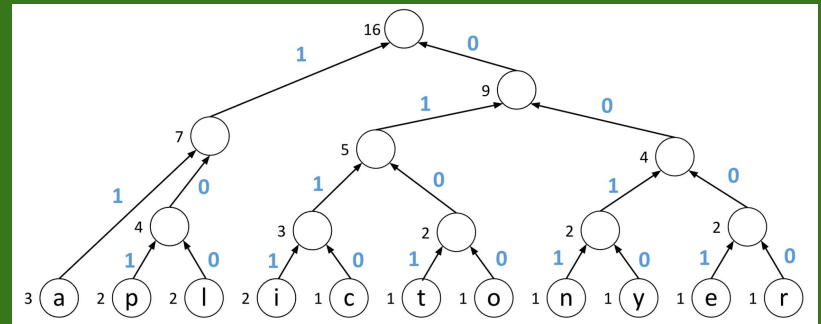
1. Transport Layer

- a. Sequence Numbers
- b. Fast Retransmission
- c. Congestion Control



2. Application Layer Encodings

- a. Base64 Encoding
- b. Huffman Encoding



Thanks to this amazing team!



Quiz Time!

Quiz Instructions

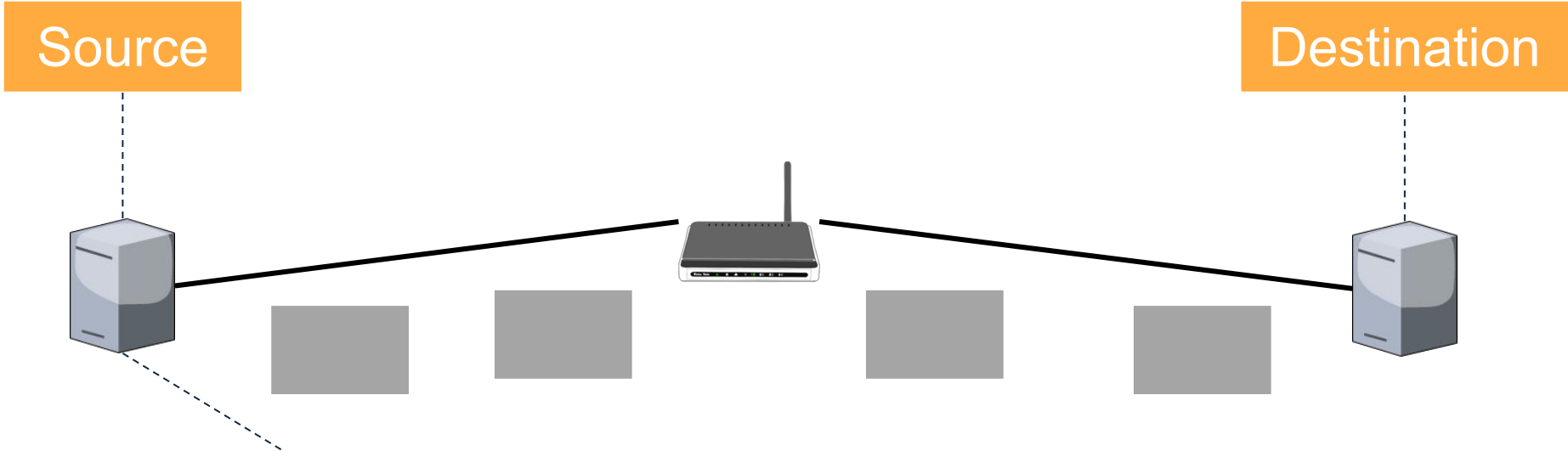
1. Keep quiz **face down** until the quiz starts
2. Do the quiz by **yourself**: no calculators, etc.
3. No talking, phones, etc., during the quiz
4. Select answer by filling in the box
 - a. ✓ Good:
 - b. × Bad:
5. Hand in your quiz at the end



Good Luck on The Final!

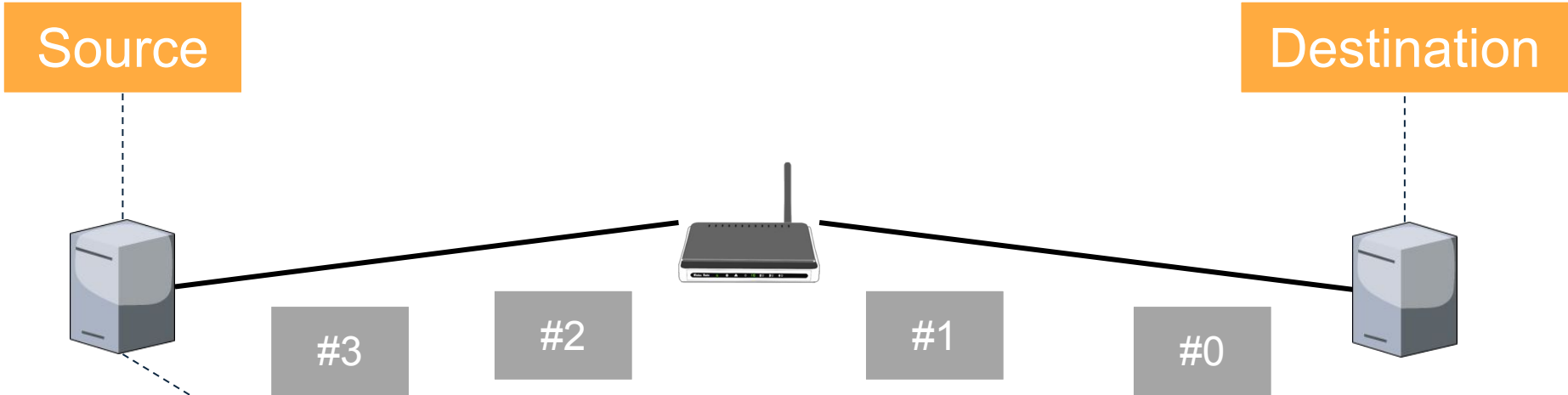
EXTRA MATERIAL

Recap – Initial Sequence Number (ISN)



I want to send 4 packets

Recap – Initial Sequence Number (ISN)



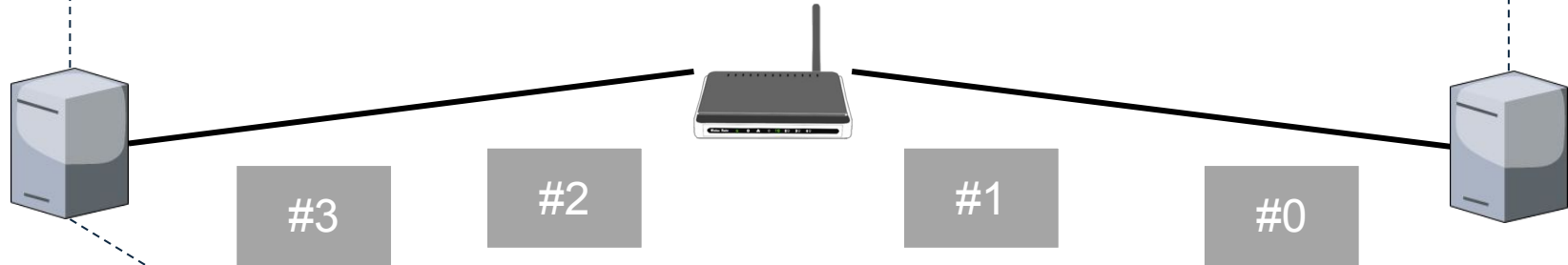
I want to send 4 packets

Recap – Initial Sequence Number (ISN)

Initial Sequence Number (ISN): 0

Source

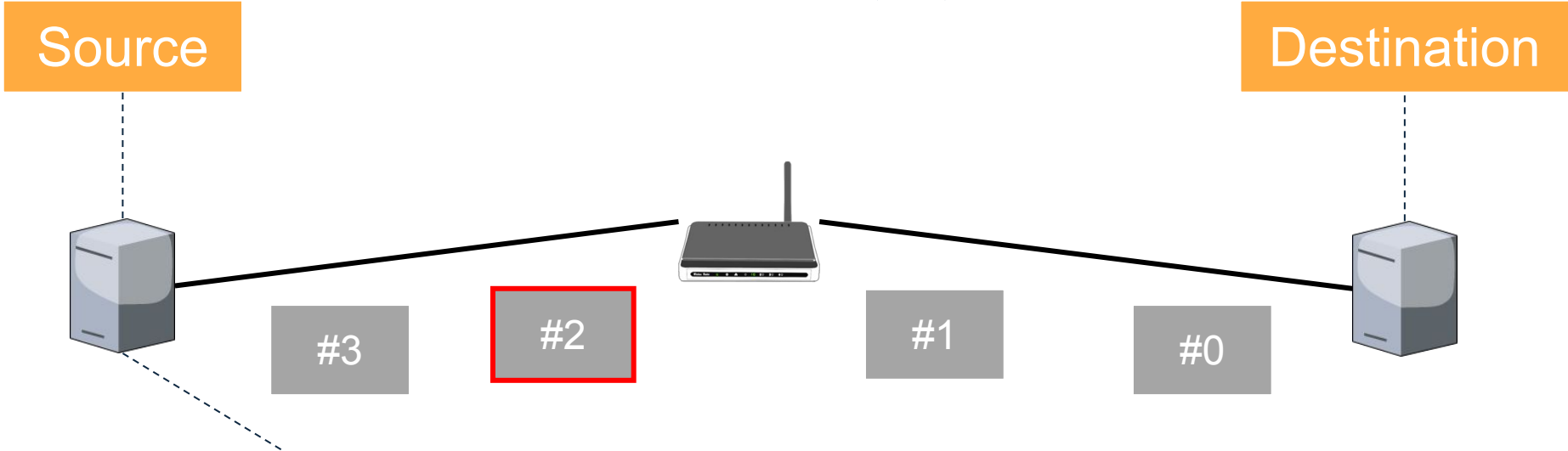
Destination



I want to send 4 packets

Recap – Initial Sequence Number (ISN)

Initial Sequence Number (ISN): 0



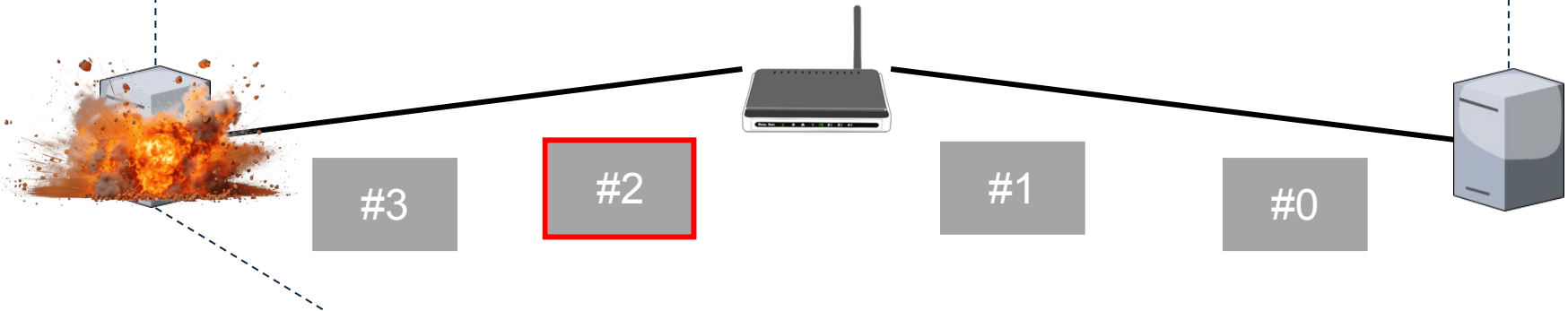
I want to send 4 packets

Recap – Initial Sequence Number (ISN)

Initial Sequence Number (ISN): 0

Source

Destination



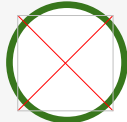
I want to send 4 packets

Recap – Initial Sequence Number (ISN)

Initial Sequence Number (ISN): 0

Source

Destination



What if we crash? How to guarantee that SN we will start with is not on the network?

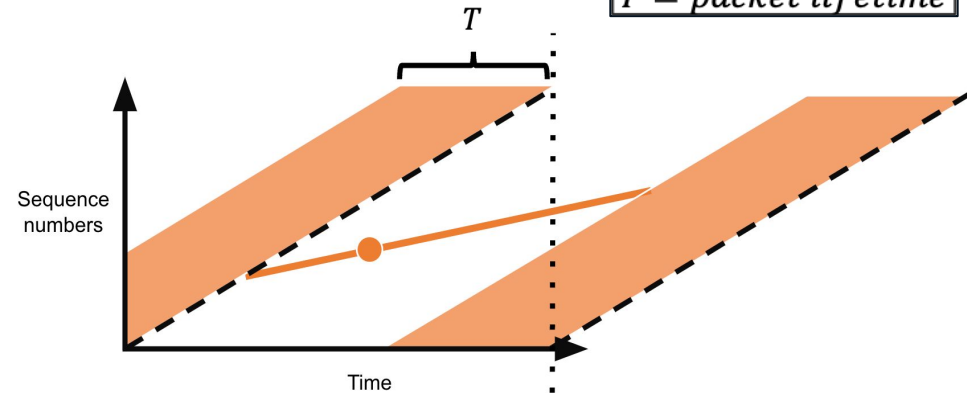
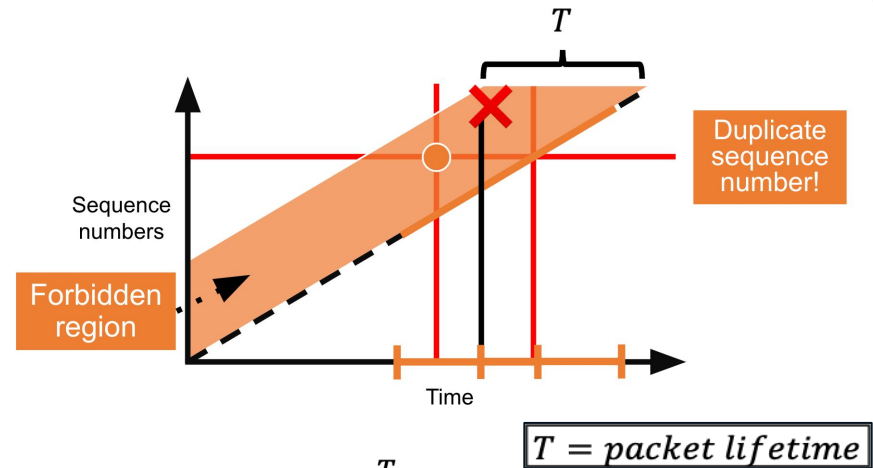
#0

#0

I want to send 4 packets

Recap – Clock-Based Initial Sequence Numbers

1. Base initial SN on some clock in a way that would guarantee **absence of duplicates**.
2. The clock is assumed to continue running even if the host goes down.
3. Forbidden region: A window of time and sequence space that must not overlap with the potential lifetime of previously sent packets.



Exercise 7 – Forbidden Regions

Consider a TCP CN protocol that uses a time-of-day clock to determine initial packet sequence numbers. The clock uses a **10-bit counter**, and ticks once every **125 milliseconds**. The maximum packet lifetime is **64 seconds**.

If the sender sends **4 packets per second** and the sequence number increment per packet sent, how long could the connection last without entering the forbidden region from above?

Solution (1/2) – Forbidden Regions

- SN Range = 2^{10}

- Ticks per second = $\frac{1}{\text{clock tick}} = \frac{1}{0.125\text{s}} = 8$

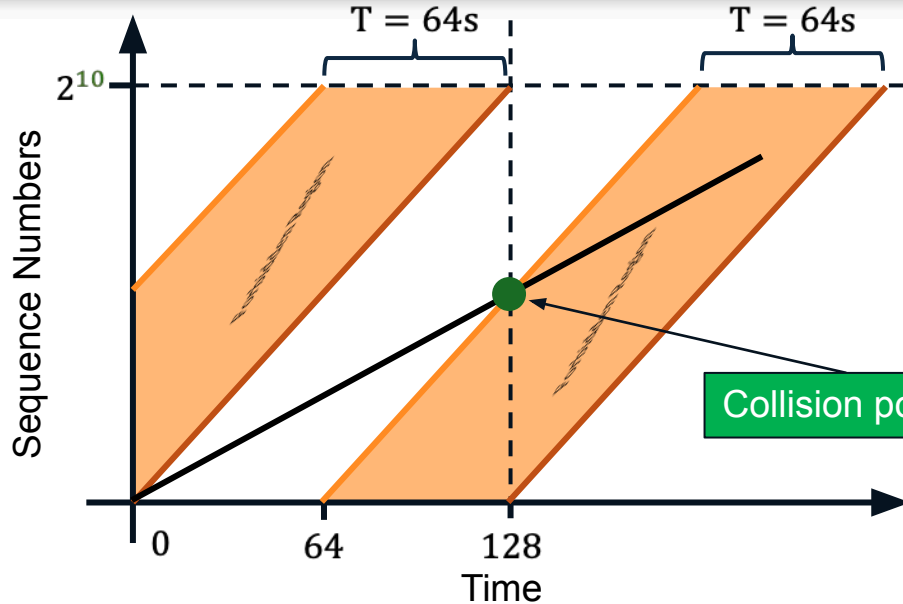
$1\text{ms} = 1 \times 10^{-3}\text{s}$

- Wrap around time = $\frac{\text{SN Range}}{\text{Ticks per second}} = \frac{2^{10}}{8} = \frac{2^{10}}{2^3} = 2^7 = 128$

Setup:

- Clock counter: 10 bits
- Clock tick: every 125 ms
- Max packet lifetime: 64 seconds
- Send rate: 4 packets per second

Solution (2/2) – Forbidden Regions



- Clock: $y = 8x$ or $y = 8(x - 128)$
- Packets SN: $y = 4x$
- Forbidden Region: $y = 8(x + 64)$
or $y = 8(x - 128 + 64) = 8(x - 64)$

$$4x = 8(x - 64)$$

$$4x = 8x - 2^9$$

$$-4x = -2^9$$

$$x = \frac{2^9}{2^2} \rightarrow 2^7 = 128$$




Answer: 128 seconds

Setup:

- Clock counter: 10 bits
- Clock tick: every 125 ms
- Max packet lifetime: 64 seconds
- Send rate: 4 packets per second

- SN Range = 2^{10}
- Ticks per second = $\frac{1}{0.125} = 8$
- Wrap around time = $\frac{2^{10}}{2^3} = 128s$

Legend:

-  Initial Sequence Numbers
-  Forbidden Region
-  Sequence Number

Exercise 8 – Forbidden Regions

Consider a TCP CN protocol that uses a time-of-day clock to determine initial packet sequence numbers. The clock uses a **9-bit counter**, and ticks once every **250 milliseconds**. The maximum packet lifetime is **32 seconds**.

If the sender sends **3 packets per second**, how long could the connection last without entering the forbidden region from above?

Solution (1/2) – Forbidden Regions

- SN Range = 2^9

- Ticks per second = $\frac{1}{\text{clock tick}} = \frac{1}{0.250\text{s}} = 4$

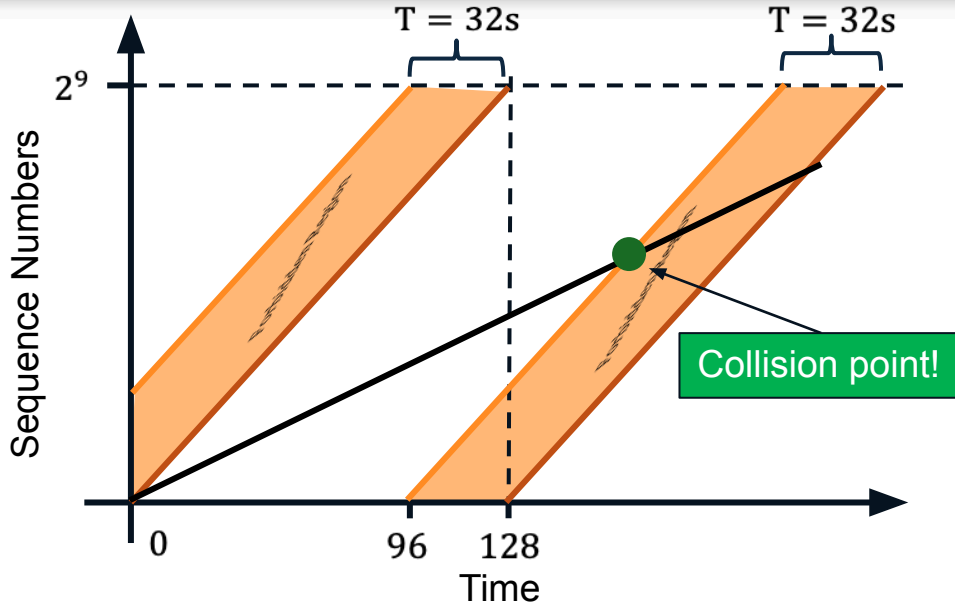
$$\boxed{1\text{ms} = 1 \times 10^{-3}\text{s}}$$

- Wrap around time = $\frac{\text{SN Range}}{\text{Ticks per second}} = \frac{2^9}{4} = \frac{2^9}{2^2} = 2^7 = 128$

Setup:

- Clock counter: 9 bits
- Clock tick: every 250 ms
- Max packet lifetime: 32 seconds
- Send rate: 3 packets per second

Solution (2/2) – Forbidden Regions



- Clock: $y = 4x$ or $y = 4(x - 128)$
- Packets SN: $y = 3x$
- Forbidden Region: $y = 4(x + 32)$
or $y = 4(x - 128 + 32) = 4(x - 96)$

$$3x = 4(x - 96)$$

$$3x = 4x - 384$$

$$-x = -384$$

$$x = 384 \text{ s}$$




Answer: 384 seconds

Setup:

- Clock counter: 9 bits
- Clock tick: every 250 ms
- Max packet lifetime: 32 seconds
- Send rate: 3 packets per second

- SN Range = 2^9
- Ticks per second = $\frac{1}{0.250} = 4$
- Wrap around time = $\frac{2^9}{2^2} = 128\text{s}$

Legend:

-  Initial Sequence Numbers
-  Forbidden Region
-  Sequence Number

Exercise 9 – Base64 Encoding

Translate the following Base64 string to 8 bit ASCII:
YmFzZQ==

a	b	c	d	e	f	g	h	i	j	k	l	m
0x61	0x62	0x63	0x64	0x65	0x66	0x67	0x68	0x69	0x6a	0x6b	0x6c	0x6d
n	o	p	q	r	s	t	u	v	w	x	y	z
0x6e	0x6f	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77	0x78	0x79	0x7a

Solution – Base64 Encoding

Y = 24 = 011000	Z = 25 = 011001
m = 38 = 100110	z = 51 = 110011
F = 5 = 000101	Q = 16 = 010000

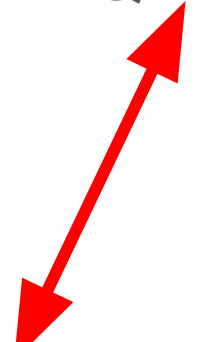
011000 100110 000101 110011 011001 01

011000 100110 000101 110011 011001 010000

01100010 01100001 01110011 01100101

0x62 (b) 0x61 (a) 0x73 (s) 0x65 (e)

YmFzZQ==



Answer: base

Exercise 10 – Huffman Encoding

- Create a Huffman coding for “a cat in a hat” (including spaces).
- What is the length difference between ASCII (8-bit per symbol) encoding and Huffman coding?

