Tutorial 1: Physical & Link Layer

Lukas Kompatscher

Computer Networks 2025-2026

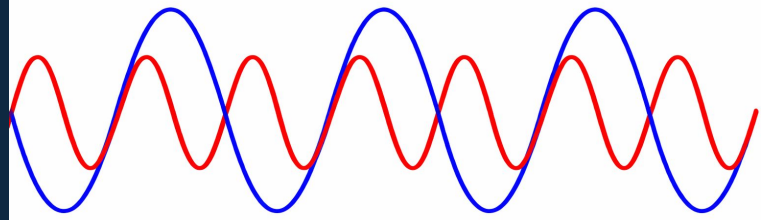
Menu of The Day

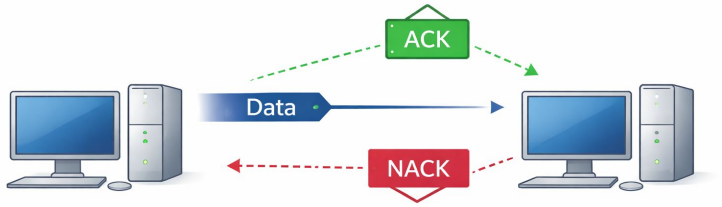
Physical Layer

1. Nyquist vs Shannon

Link Layer

1. Package Framing
   A. Framing Protocols
   B. Acknowledgement handling
2. Error Correction and Detection
   C. Checksum
   D. Cyclic Redundancy check

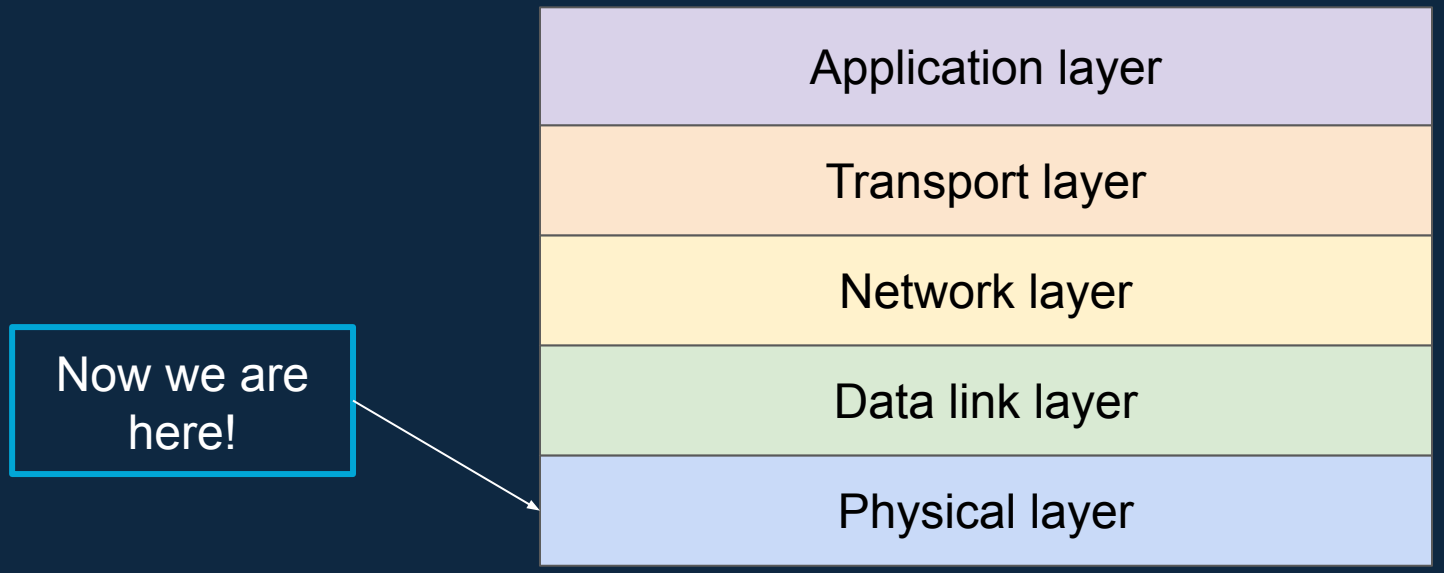




1110
Checksum

1101 0100
CRC

Physical Layer



Communication Limits

Noiseless (Nyquist Theorem)

Shows how data rate increases with bandwidth and signal levels.

Finds: Data Rate

Given: number of signal levels

$$C = 2 \times B \times \log_2(V)$$

C Maximum Data Rate

B Bandwidth

V Number of discrete signals

Noisy (Shannon Theorem)

Give realistic maximum data rate with noise.

Finds: Data Rate

Given: signal to noise ratio

$$C = B \times \log_2\left(1 + \frac{S}{N}\right)$$

C Capacity

B Bandwidth

$\frac{S}{N}$ Signal-to-noise ratio

Communication Limits - Exercise

Consider a physical-layer protocol that uses **256** signal levels on a channel with **100 MHz** bandwidth and an SNR of **30 dB**.

What is the maximum data rate that we can obtain using this protocol and channel?

Nyquist: $C = 2 \times B \times \log_2(V)$

Shannon: $C = B \times \log_2\left(1 + \frac{S}{N}\right)$ $\frac{S}{N} = 10^{\left(\frac{dB}{10}\right)}$

Communication Limits - Solution

256 signal levels, 100 MHz bandwidth, SNR of 30 dB

What is the maximum data rate?

$$\frac{S}{N} = 10^{\left(\frac{dB}{10}\right)}$$

Nyquist

$$2 \times B \times \log_2(V)$$

$$2 \times 100 \times 10^6 \times \log_2(256)$$

$$2 \times 100 \times 10^6 \times 8$$

$$1.6 \times 10^9 \Rightarrow R = 1.6 \text{ Gbps}$$

Shannon

$$B \times \log_2\left(1 + \frac{S}{N}\right)$$

$$100 \times 10^6 \times \log_2(1 + 10^3)$$

$$\approx 10^9 \Rightarrow 1 \text{ Gbps}$$

$$\text{R} = \min(1.6 \text{ Gbps}, 1 \text{ Gbps}) = 1 \text{ Gbps}$$

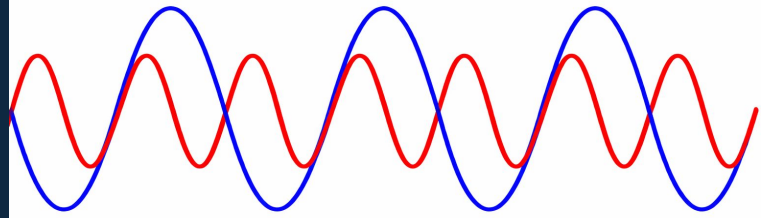
Menu of The Day

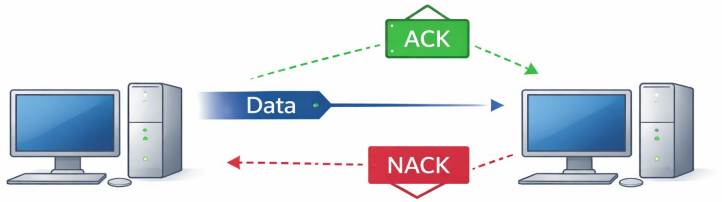
~~**Physical Layer**~~

~~1. Nyquist vs Shannon~~

Link Layer

1. Package Framing
   A. Framing Protocols
   B. Acknowledgement handling
2. Error Correction and Detection
   C. Checksum
   D. Cyclic Redundancy check

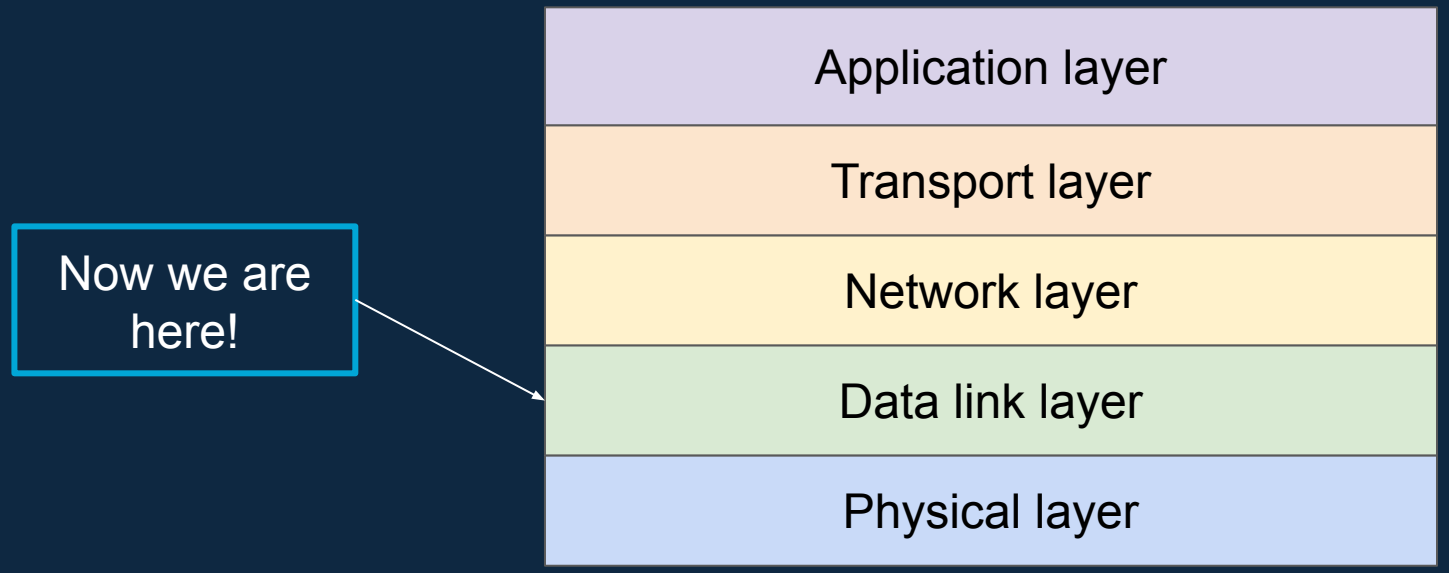




1110
Checksum

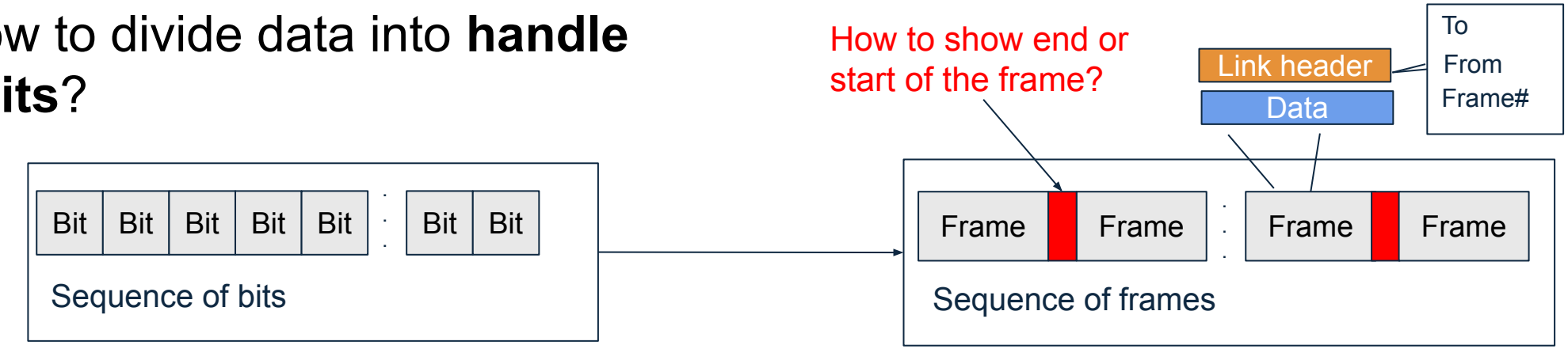
1101 0100
CRC

Link Layer

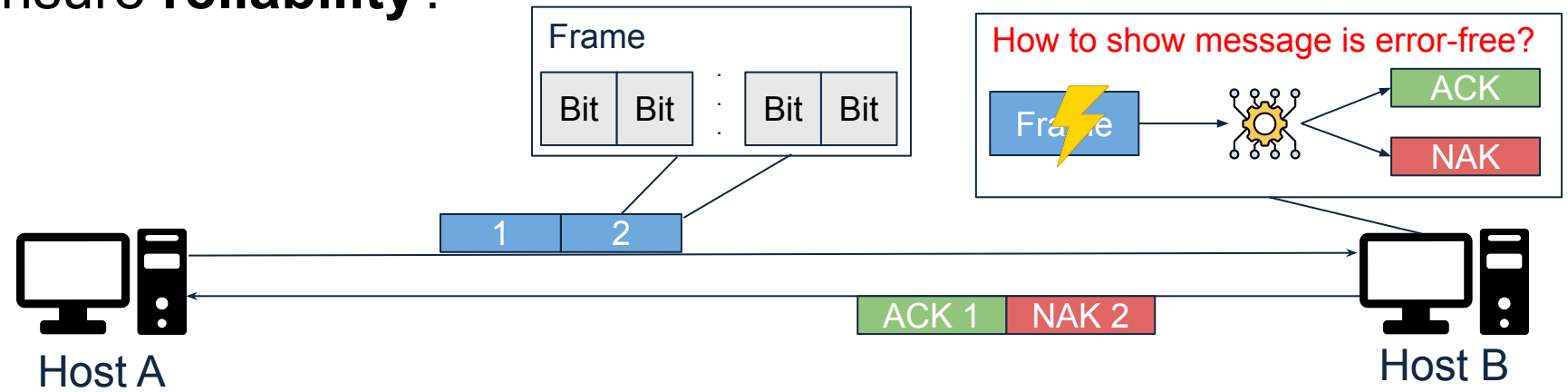

Application layer
Transport layer
Network layer
Data link layer
Physical layer
Now we are here!

Why link layer ?

How to divide data into **handle units**?



How to ensure **reliability**?



Bits and Bytes

$$1\ Kb = 10^3 bits$$

$$1\ KB = 10^3 bytes$$

$$= 1000\ bytes$$

$$1\ MB = 10^6\ bytes$$

$$1\ GB = 10^9\ bytes$$

$$1\ TB = 10^{12}\ bytes$$

$$1\ Kib = 2^{10} bits$$

$$1\ KiB = 2^{10} bytes$$

$$= 1024\ bytes$$

$$1\ MiB = 2^{20}\ bytes$$

$$1\ GiB = 2^{30}\ bytes$$

$$1\ TiB = 2^{40}\ bytes$$

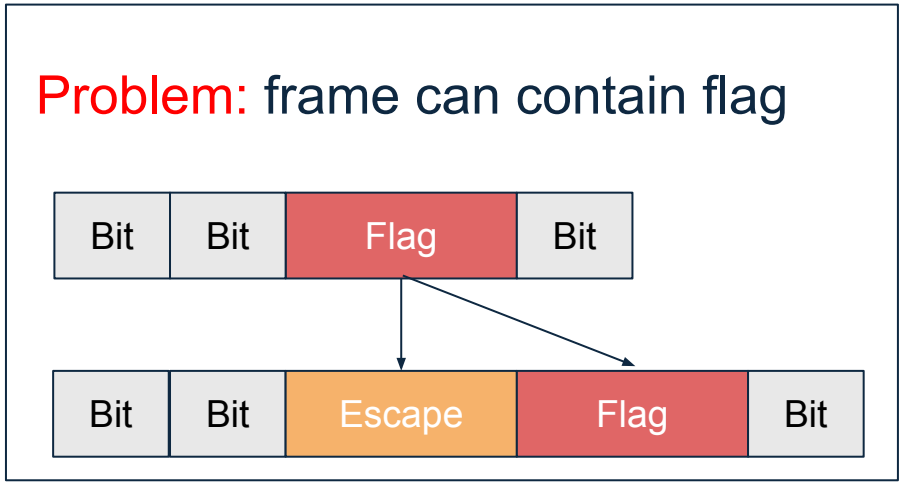
Framing

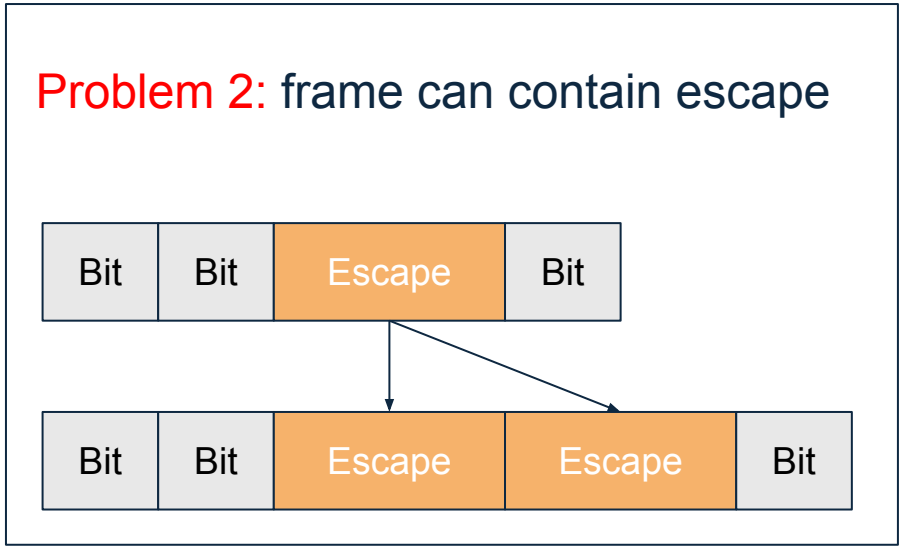
Byte Stuffing

Insert escape bytes before control bytes

Bit Stuffing (very similar)

Insert 0s to prevent confusion with control flags

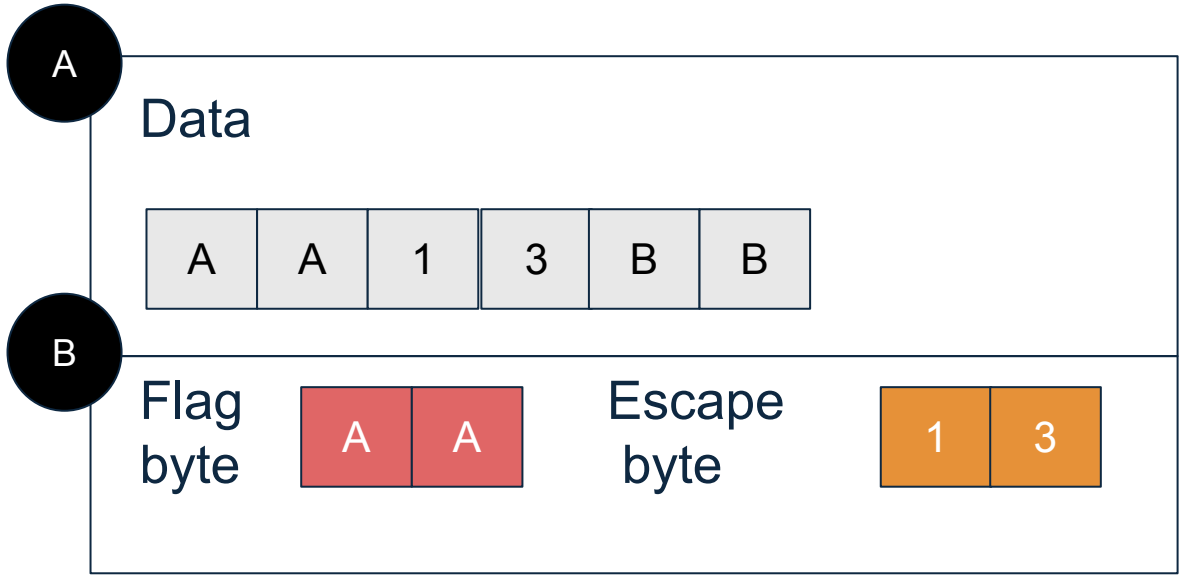




Framing - Exercise

Frame in (A) is transmitted over the wire.

Show what the frame will look like after byte stuffing. Flag and escape symbols are shown respectively (B).



Do not forget to add flags both at the beginning and end of frame.

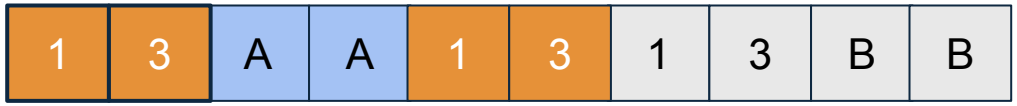
Calculate transmission overhead in bytes. Since symbols are represented in hexadecimal, each symbol is half a byte.

$$\text{Overhead} = \frac{\text{flag+esc.}}{\text{data}}$$

Framing - Solution

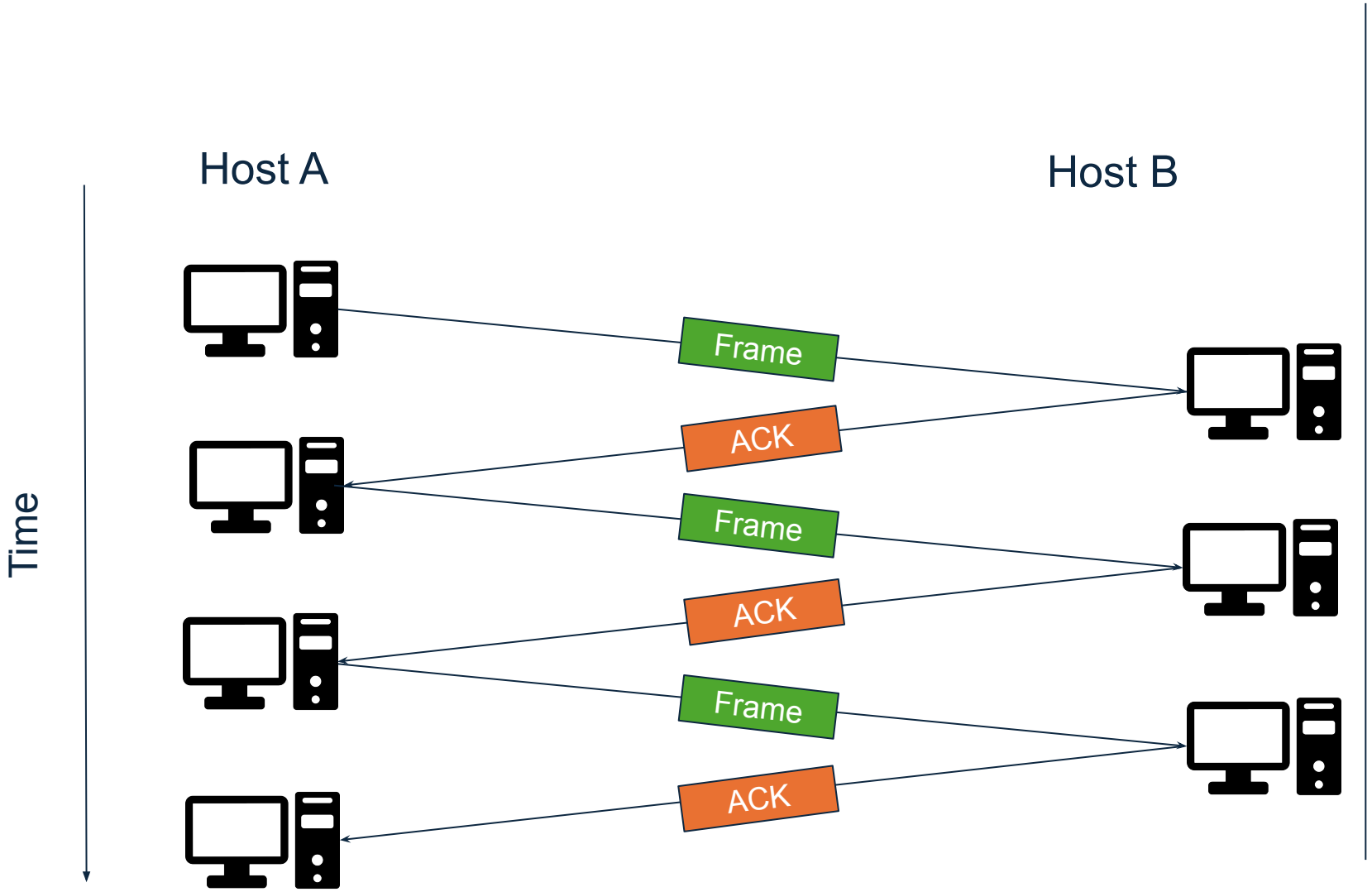
Data

A	A	1	3	B	B

Operation	Resulting frame
Escape escape	
Escape flag	
Add flags to frame	

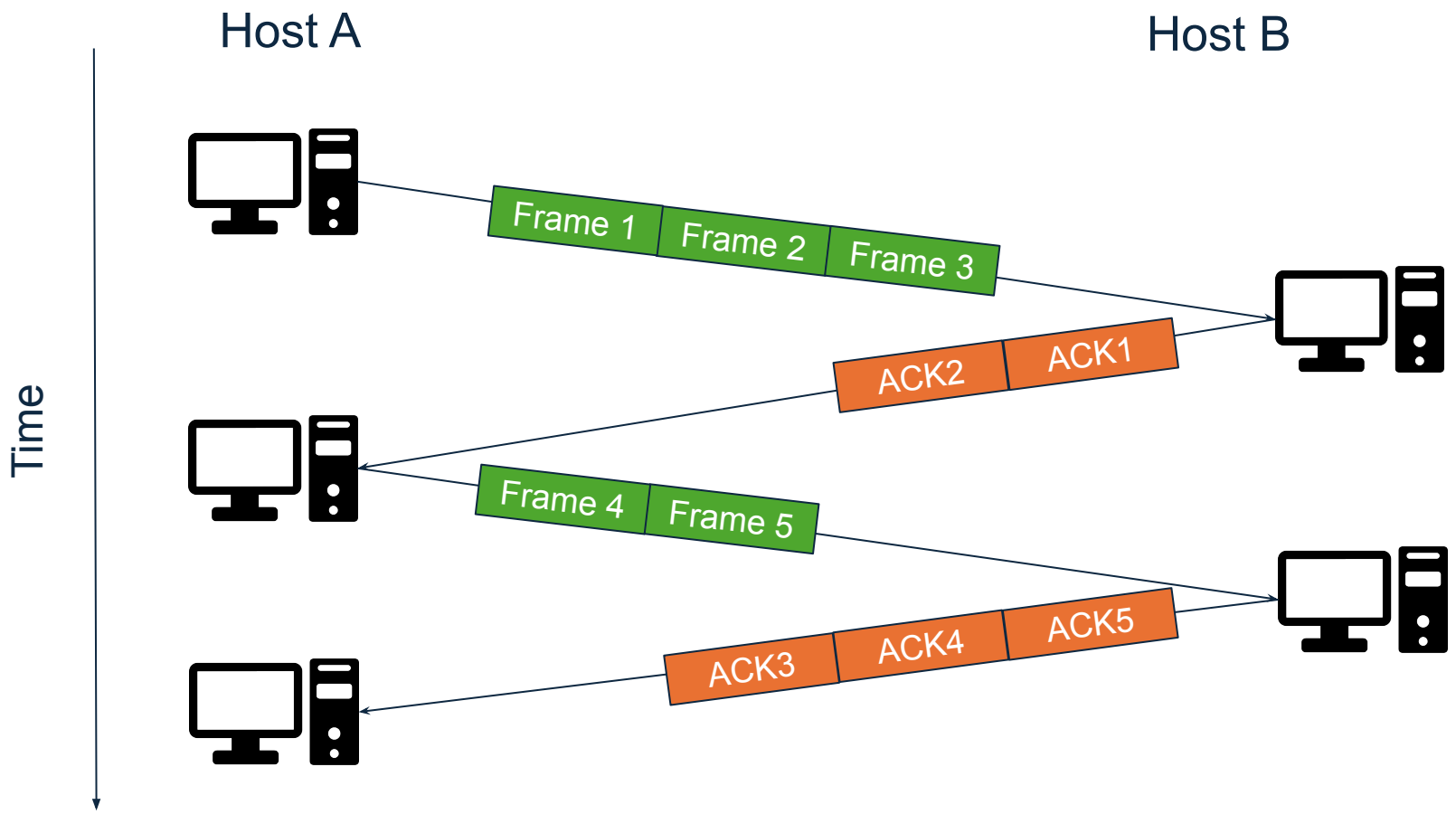
Overhead
Byte stuffing 4 ÷ 3 = 133%

Stop-and-Wait protocol



Disadvantage: low link utilization

Sliding window protocol



Key idea

Do not wait for ack. Pipeline!

Advantage: higher link utilization (LU)

Disadvantage: harder to implement

Formula:

$$lu \leq \frac{w}{1 + 2 * B_f * D}$$

Sliding Window Protocol - Exercise

You use a sliding-window protocol with a window size of **1** (effectively stop and wait protocol) and frame size of **10,000 bits** over a channel with propagation delay of **8 ms**. The max. data rate of the physical channel is **1 Gb/s.** What is the maximum link utilization (LU) you can achieve? Now, what if you use a window size of **100**? What is the maximum LU you can achieve?

$Link\ utilization$:

$$LU \leq \frac{w}{1 + 2 \times B_f \times D}$$

$$B_f = \frac{B_p}{f}$$

- Frame size (in bits/bytes): f
- Window size (in frames): w
- Bandwidth (max. data rate of physical channel): B_p
- Bandwidth (frames per second): B_f
- Propagation delay (in seconds): D

Sliding Window Protocol - Solution

$$B_f = \frac{B_p}{f} = \frac{1 \text{ Gb/s}}{10{,}000 \text{ bits/frame}} = 100{,}000 \; frame/s$$

Window Size = 1

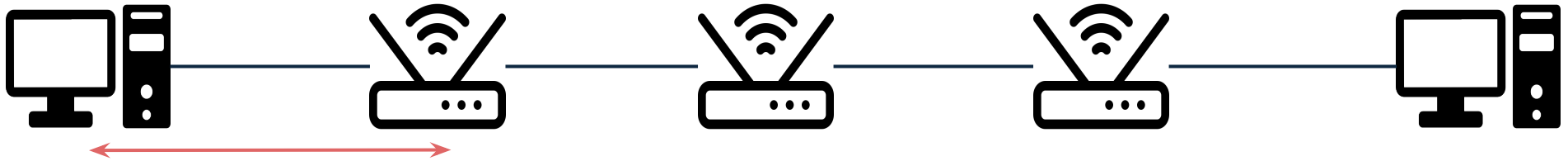
$$LU \leq \frac{w}{1 + B_f \times D} = \frac{1}{1 + 2 * (100{,}000 \text{ frame/s}) \times (8\text{ms})} = \frac{1}{1601} = 0.0006$$

Window Size = 100

$$LU \leq \frac{w}{1 + B_f \times D} = \frac{100}{1 + 2(100{,}000 \text{ frame/s}) \times (8\text{ms})} = \frac{100}{1601} = 0.06$$

Link Layer Transmission - Exercise

The figure below is a model of the networked system. In this case, retransmission is the responsibility of the **link layer**, every router between the hosts detects loss and issues retransmission. That is whenever data is lost only the nearest router retransmits it. **Hop traversal (HT)** is **0.2 sec.** The **retransmission timeout (RT)** between hops is set to **0.4 sec.** How long does it take to successfully send a message across if the reliability of the individual link is **1.0** and **0.5?**

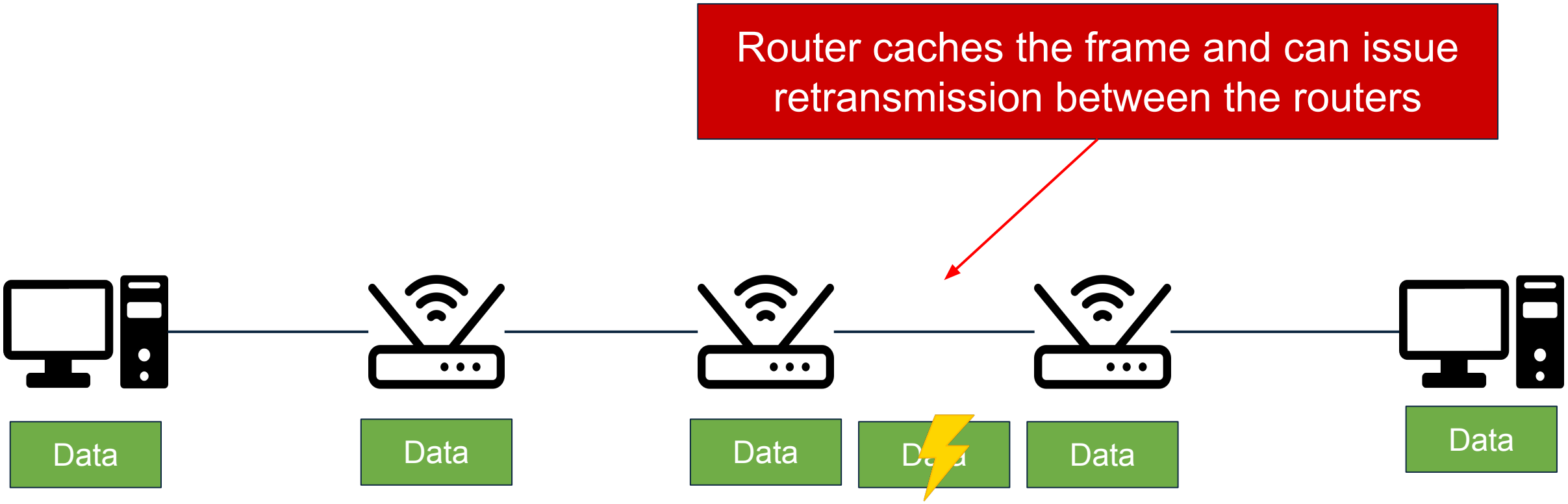


Link layer scope

$$Expected\ transmission\ count\ (EC) = \frac{1}{LR}$$

$$Expected\ transmission\ time = (EC\ - 1) \times RT + HT$$

Link Layer Transmission - Solution (Reference Model)



Link Layer Transmission - Solution

Expected transmission count (EC)= $\frac{1}{Link\ reliablity}$

a) Case 1.0: $\frac{1}{1} = 1$

b) Case 0.5: $\frac{1}{0.5} = 2$

Total time for retransmissions

Time to 1 hop = (EC -1) × retransmission timeout + hop traversal

a) Case 1.0: $(1-1) \times 0.4s + 0.2s = 0.2s$

b) Case 0.5: $(2-1) \times 0.4s + 0.2s = 0.6s$

Link Layer: time to destination = hop count × time for 1 hop

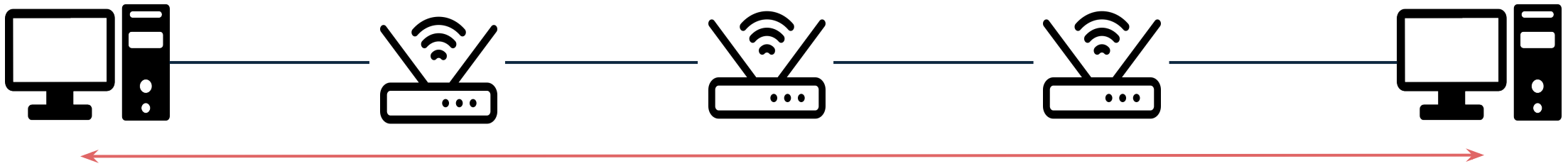
a) Case 1.0: $4 \times 0.2s = 0.8\ s$

b) Case 0.5: $4 \times 0.6s = 2.4\ s$

Transport Layer Transmission - Exercise

The figure below is a model of the networked system. In this case, retransmission is the responsibility of the **transport layer,** only final hosts detect loss, and issues retransmission. Hop travels takes **0.2 s.** The end-to-end retransmission timeout is set to **1.6 s.** How long does it take to successfully send a message across if link reliability is **1.0** and **0.5**?

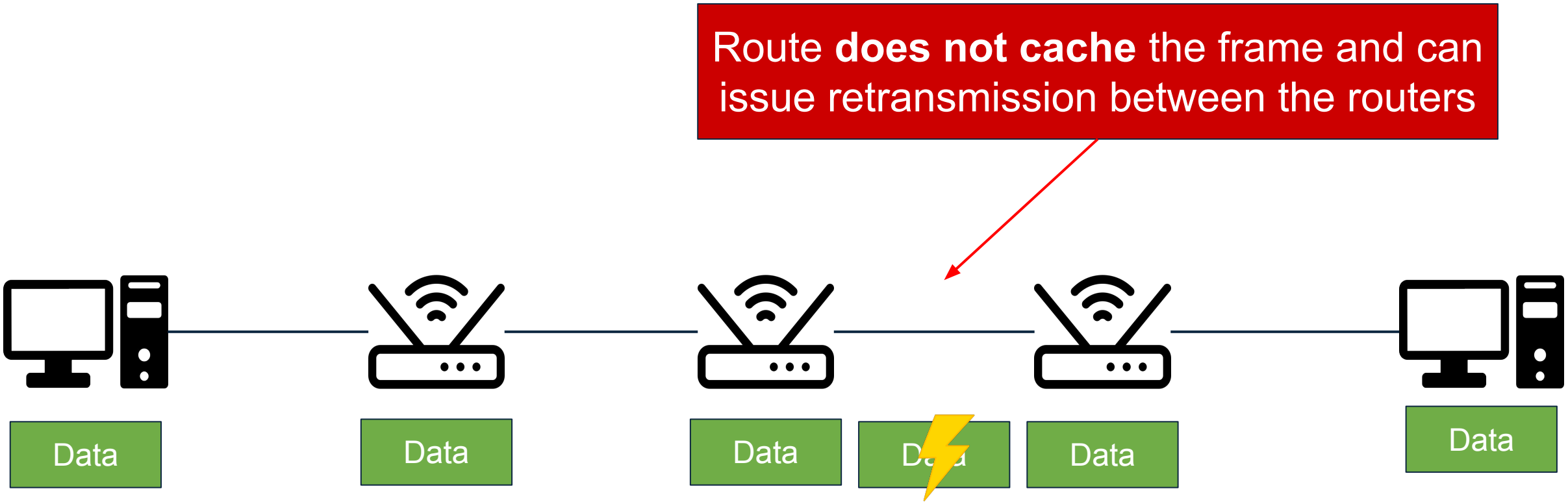
Which solution do you think is used in modern networks and why?



Transport layer scope

$$End\ to\ end\ reliablity = link\ reliablity^{hop\ count}$$

Transport Layer Transmission - Solution (Reference Model)



Transport Layer Transmission - Solution

End-to-end reliability

a) Case 1.0: $1.0^4 = 1.0$

b) Case 0.5: $0.5^4 = 0.06$

Expected transmission count (EC) = $\frac{1}{endtoend\ reliablity}$

a) Case 1.0: $\frac{1}{1.0} = 1$

b) Case 0.5: $\frac{1}{0.06} = 16$

Link layer
Case 1.0: 0.8 s
Case 0.5: 2.4 s

What is better in practice link or transport layer?

Time to destination = (EC -1) × retransmission timeout + time to receiver

a) Case 1.0: $(1-1) \times 1.6\ s + 0.8\ s = 0.8\ s$

b) Case 0.5: $(16-1) \times 1.6\ s + 0.8\ s = 24.8\ s$

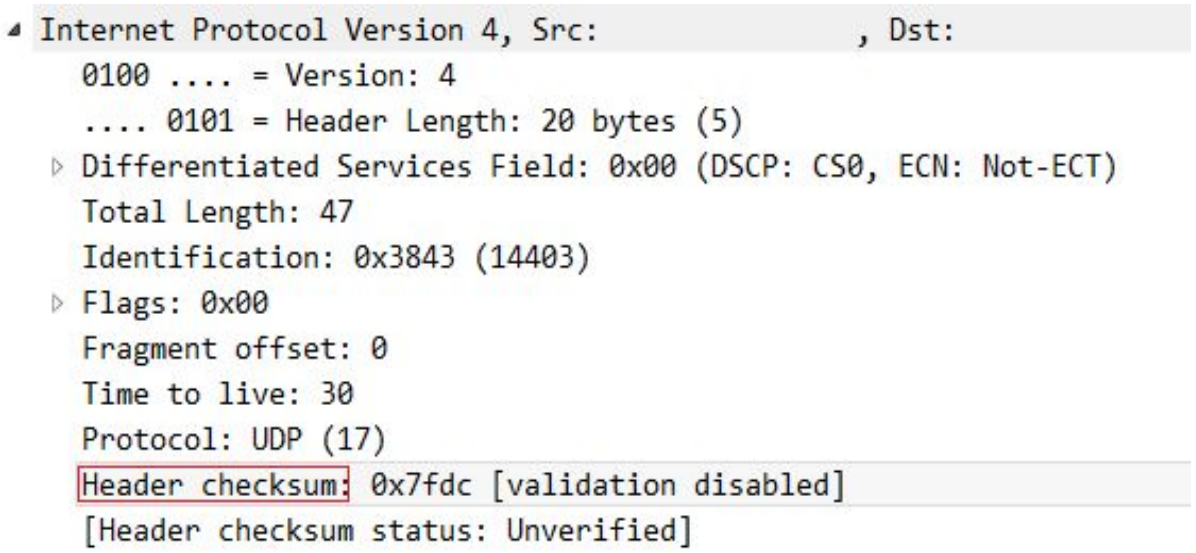
Nr. hops × hop traversal time

BREAK

Part 2: Error Correction and Detection

Error Detection - Checksum

Wireshark Scan



! Can miss some errors (not as strong as CRC)

- Used in IP, UDP and TCP
- Used to detect transmission errors (not perfect, but fast)

Steps

1. Divide data into N-bit words
2. Add words up using one's complement arithmetic
3. Invert the result

Checksum - Example

For example, the data is 1010 1100 and

N = 4

One's complement addition:

1. Divide words into N-bit words
2. Add words up "normally"
3. If carry – add it "to the back"
4. Invert the result
5. Host performs check with sum

1. Words are 1010 and 1100

2.
```
  1010
+ 1100
 10110
```

3.
```
     1
+ 0110
 0111
```

4. Answer 1000

5. Check on host

1000 + 0111 = 1111 (no errors)

Checksum - Exercise

You want to send the data bits 1001 0101 0100.

You use a checksum to enable the receiver to detect errors in the data. You divide the message in groups of four bits and calculate the checksum (using one's complement arithmetic).

What is the checksum that you transmit?

Checksum - Solution

We have data 1001 0101 0100.

1. Divide into words of N =4
   1001, 0101, and 0100
2. Add words up using ones complement
3. Add 1001 to 0101
4. Add result of previous step (1110) to 0100
5. Carry! Add ”to the back”
6. Invert the result

$$\begin{array}{r} 1001 \\ +\ 0101 \\ \hline 1110 \end{array} \rightarrow \begin{array}{r} 1110 \\ +\ 0100 \\ \hline 10010 \end{array} \rightarrow \begin{array}{r} 1 \\ +\ 0010 \\ \hline 0011 \end{array}$$

Answer 1100

Error Detection - Cyclic Redundancy Check (CRC)

Pick a generator polynomial of Nth power and write coefficients down in binary.

Examples:

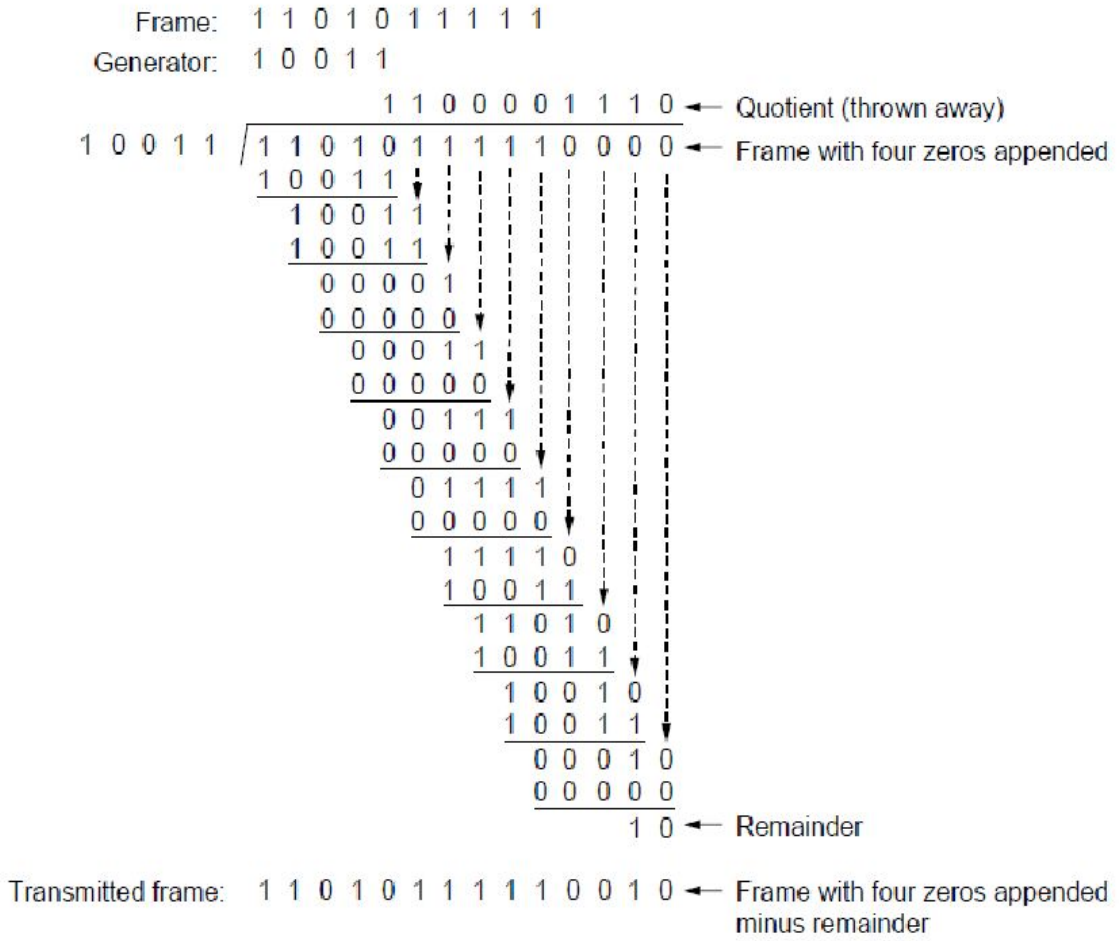
$1x^3 + 1x^2 + 0x + 1 = 1101$

$x^3 + x^2 + 1 = 1101$ (Same as above)

1. Append N 0s to the end of frame.

2. With mod 2 arithmetic divide a frame by the generator until can no more.

3. Substitute the appended 0s with the calculated remainder.



Cyclic Redundancy Check (CRC) - Example

For example, the generator is $x^3 + x^2 + 1$ and message is 1111. $1x^3 + 1x^2 + 0x + 1 = 1101$

1. Add N (three) 0s to the end of the message

2. Using mod 2 arithmetic keep dividing until can no more.

3. Substitute remainder at the end of the message.

How does final host check for correctness?

Final Message: 1111 111

```
1111000
1101000
-------
0010000
  11010
  -----
  01010
   1101
   ----
    111
```

Cyclic Redundancy Check (CRC) - Exercise

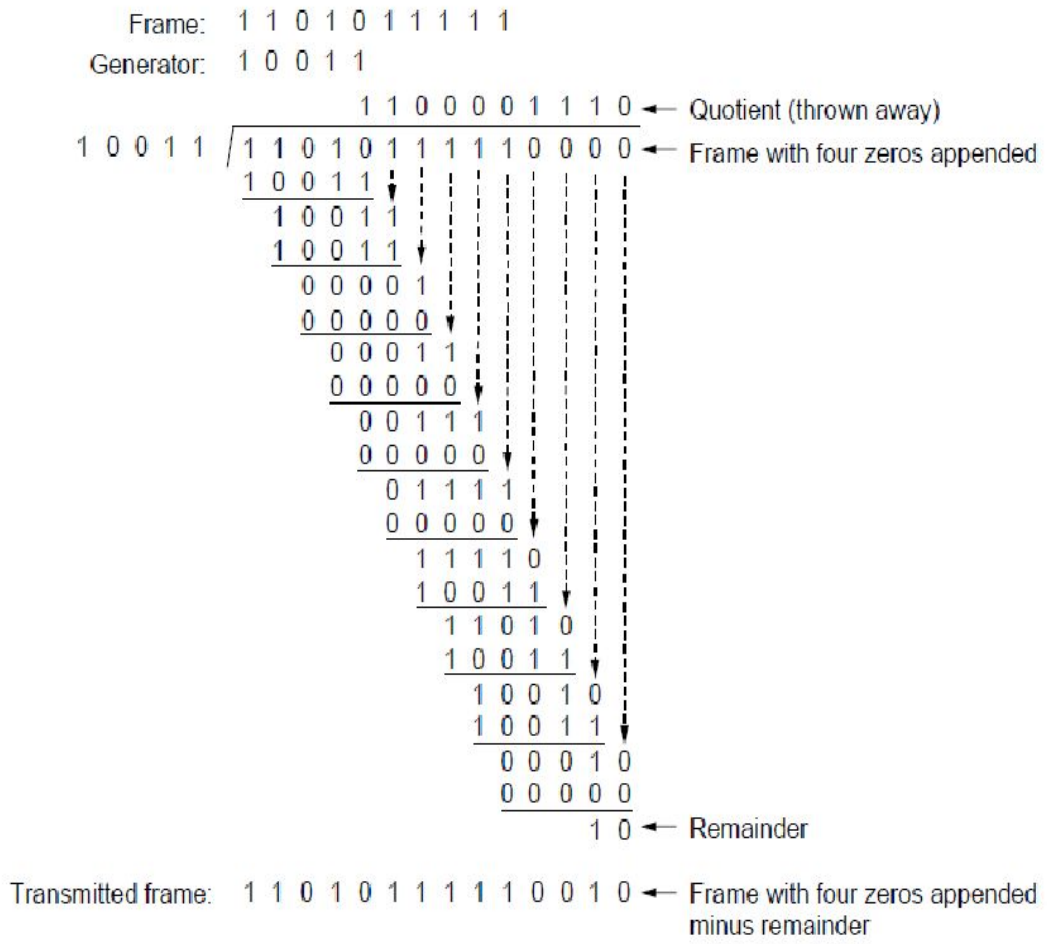
You want to send the data bits

1011 0001 0101 1001.

You use CRC to enable the receiver to detect errors in the data. You use the generator polynomial **1011** (already in binary form).

What is the code word that you transmit?

Include the whole codeword (data bits + CRC check bits)



Cyclic Redundancy Check (CRC) - Solution

Data – 1011 0001 0101 1001

Polynomial – 1011

Add three 0s to the message and divide until you can't no more.

Final message to divide: 1011 0001 0101 1001 000

We also leave 0s out for clarity!

Final Message: 1011 0001 0101 1001 101

```
1011 0001 0101 1001 000
1011
     1 0101 1001 000
     1 011
         1110 01 000
         1011
          101 01 000
          101 1
              11 000
              10 11
               1 110
               1 011
                 101
```

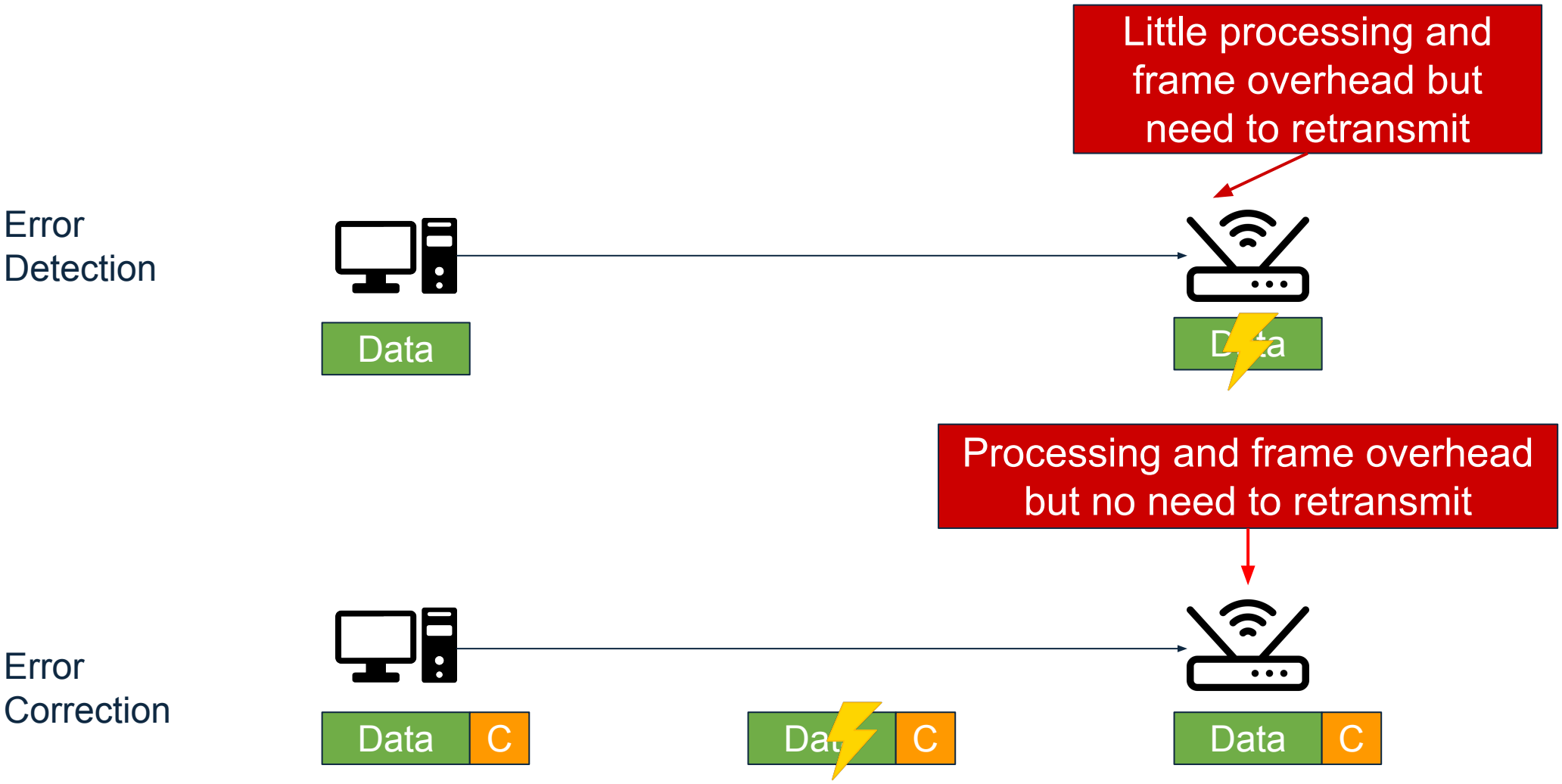
Error Detection vs Correction - Exercise

In this exercise, we work with a wired setup. Assume that, for **every Mb,** the probability of at at least 1 bit-flip is 10^{-6} (PBF). We want to choose whether to use error correction or error detection for our setup. Assume that both always correct and detect successfully. For error correction, the code rate is **0.75** and for error detection it is **1.0.** Error correction introduces an additional **0.01s** processing on the receiver side for every Mb received.

In one frame, we want to send **1 MB** of data. Link bandwidth **8 Mb/s,** and latency is **0.2s.** The retransmission timeout is **1s (RT).** How long does it take to successfully send a message across?

$$\text{PEF} = (1 - \text{PBF})^{\text{Mb count}} \qquad \text{transmission time} = \frac{\frac{\text{data size}}{\text{code rate}}}{\text{bandwith}}$$

Error Detection vs Correction - Solution (Reference Model)



Error Detection vs Correction - Solution

Error detection. Probability that a frame arrives error free (PEF):

$$= (1 - \text{PBF})^{\text{Mb count}} = (1 - 10^{-6})^8 = 0.9999999^8 \approx 1$$

Error detection. Expected time:

$$= (\text{EC} - 1) \times \text{retransmission timeout} + \text{successful transmission time} \rightarrow \text{EC} = \frac{1}{\text{PEF}} = \frac{1}{1} = 1$$

$$= (1 - 1) \times 1\text{s} + \frac{\frac{1\text{MB}}{1}}{8\ \text{Mb/s}} + 0.2\text{s} = \mathbf{1.2\ s}$$

$$1\ \text{MB} = 8\ \text{Mb} \rightarrow \frac{8\ \text{Mb}}{0.75} = 10.6\ \text{Mb}$$

Error correction. Time to send:

= transmission time for (frame + overhead) + latency + processing overhead of MB

$$= \frac{\frac{1\text{MB}}{0.75}}{8\ \text{Mb/s}} + 0.2\ \text{s} + 0.01\ \frac{\text{s}}{\text{Mb}} \times 1\text{MB} \approx \mathbf{1.6\ s}$$

Error Detection vs Correction - Exercise

Now we work with a wireless setup. Assume that, for every Mb, the probability of at least 1 bit-flip is 10^{-1} (PBF). Assume that both error correction and detection always correct and detect successfully. For error correction, the code rate is **0.75,** and for error detection, the code rate is **1.0.** Error correction bears additional **0.01s** processing on the receiver side for every Mb received. In one frame, we want to send **1MB** of data. The link bandwidth is **8 Mb/s,** and the latency is **0.2 s.** The retransmission timeout (RT) is **1s.**

How long does it take to successfully send a message across? Which error handling scheme is more performant and for which setup? How does this and the previous exercises contradict our notion of layered architecture?

!Round expected transmission count to nearest integer.

Error Detection vs Correction - Solution

Error detection. Probability that a frame arrives error free (PEF):

$$= (1 - \text{PBF})^{\text{Mb count}} = (1 - 10^{-1})^8 = 0.9^8 \approx 0.5$$

Error detection. Expected time:

$$= (\text{EC} - 1) \times \text{retransmission timeout} + \text{successful transmission time} \rightarrow \text{EC} = \frac{1}{\text{PEF}} = \frac{1}{0.5} = 2$$

$$= (2 - 1) \times 1s + \frac{\frac{1\text{MB}}{1}}{8\ \text{Mb/s}} + 0.2s = \mathbf{2.2\ s}$$

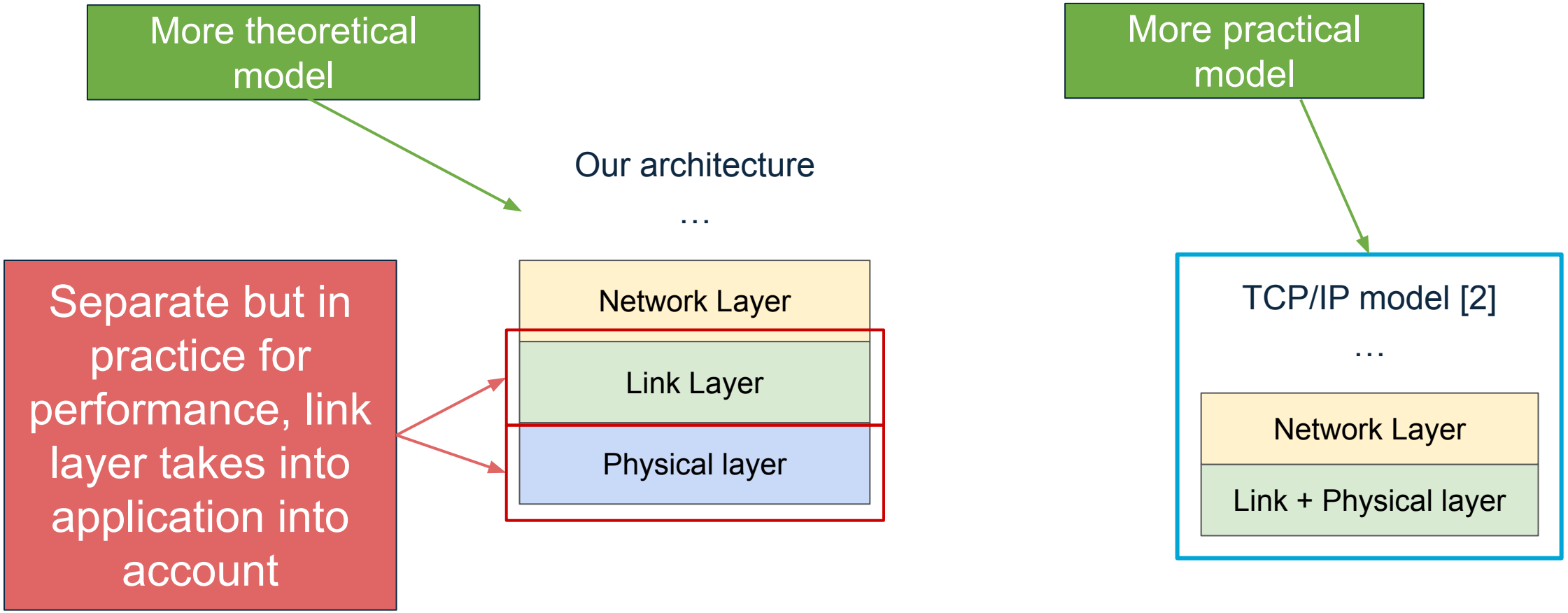
Which setup is better?

Error correction. Time to send:

= transmission time for (frame + overhead) + latency + processing overhead of MB

$$= \frac{\frac{1\text{MB}}{0.75}}{8\ \text{Mb/s}} + 0.2\ \text{s} + 0.01\ \frac{s}{\text{Mb}} \times 1\text{MB} \approx \mathbf{1.6\ s\ (same\ as\ part\ 1)}$$

Does our model hold in practice?



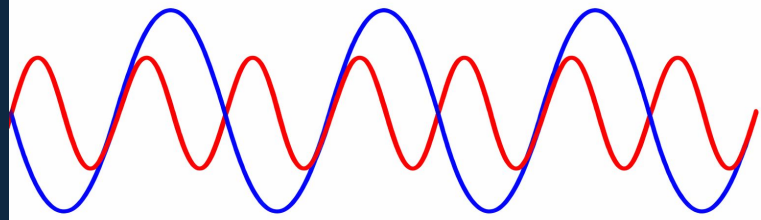
Recap

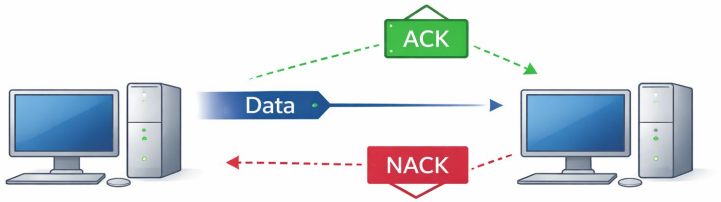
Physical Layer

1. Nyquist vs Shannon

Link Layer

1. Package Framing
   - A. Framing Protocols (Byte & Bit stuffing)
   - B. Acknowledgement handling (Stop-and-Wait vs Sliding Window)
2. Error Correction and Detection
   - C. Checksum (Error detection with 1's complement addition)
   - D. Cyclic Redundancy check (Strong error detection with polynomial division)





1110

Checksum

1101 0100

CRC