CSC 458/2209 – Computer Networking Systems

Handout # 3: Link Layer, Error Detection/Correction

Professor Yashar Ganjali
Department of Computer Science
University of Toronto

ganjali7@cs.toronto.edu
http://www.cs.toronto.edu/~yganjali

Announcements

- Programming Assignment 1 (PA1)
  - Will be posted on January 20th (next week).
  - To be completed individually.
  - Due: Feb. 14th at 5PM.
  - Please note there will be a problem set in the middle of PA1.

- Tutorials
  - This week: socket programming.
  - Next week: review of PA1.

- Links posted on class web page (along with PA1)
  - Socket programming
  - Coding guidelines
  - Some useful resources

- Use Piazza if you have any questions.

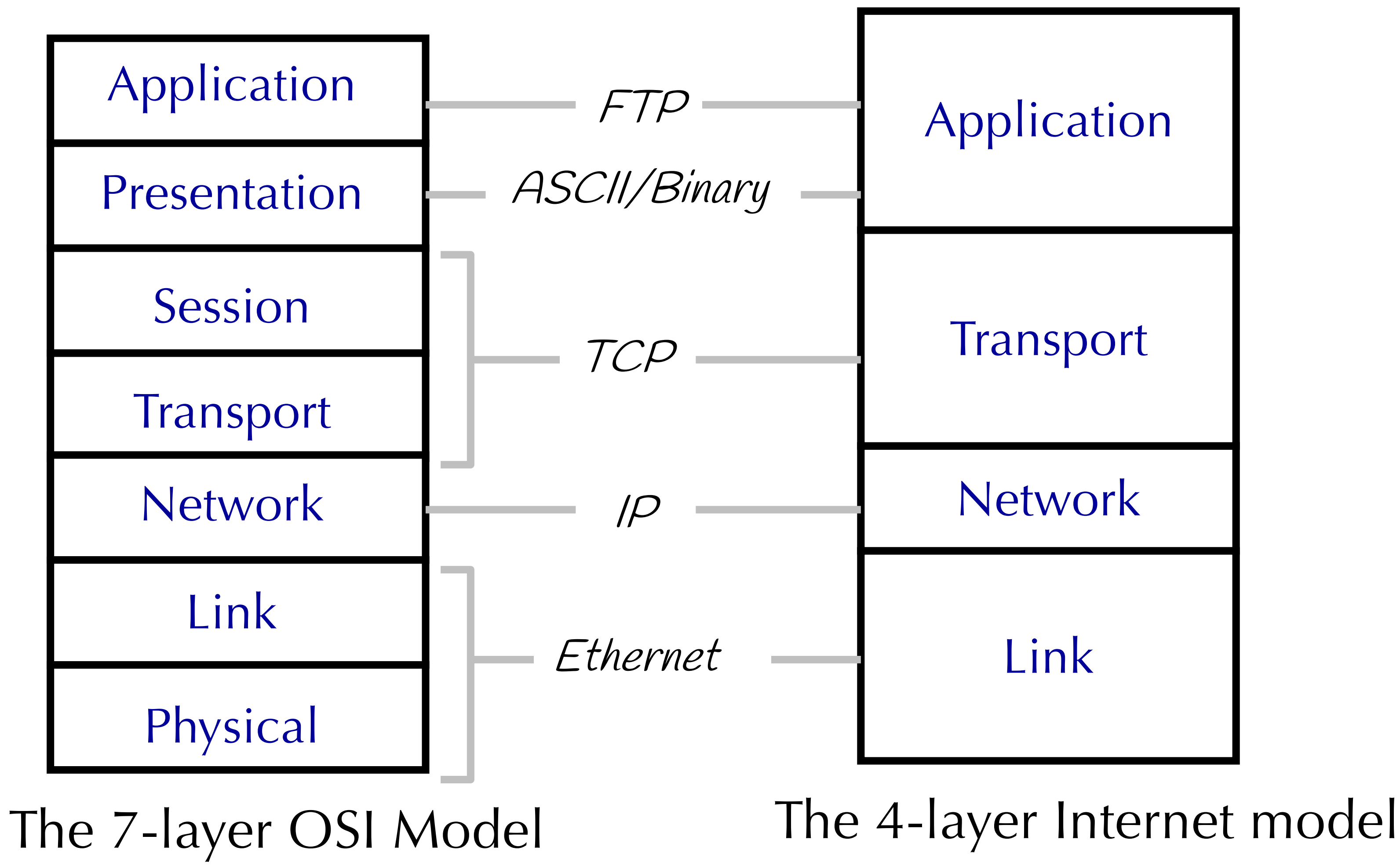
Announcements – Cont'd

- Reading for this week:
  - Chapter 2 of the textbook
  - Last week: Chapter 1

- Next week: Chapter 3

Last Time …

Protocols, layering and reference models



Outline

Part 1. Physical/link layer

- Different types of media
- Encoding bits with signals
- Framing
- Model of a link

Part 2. Error detection and correction

- Hamming distance
- Parity, checksums, …

Part 1 – Physical/Link Layer

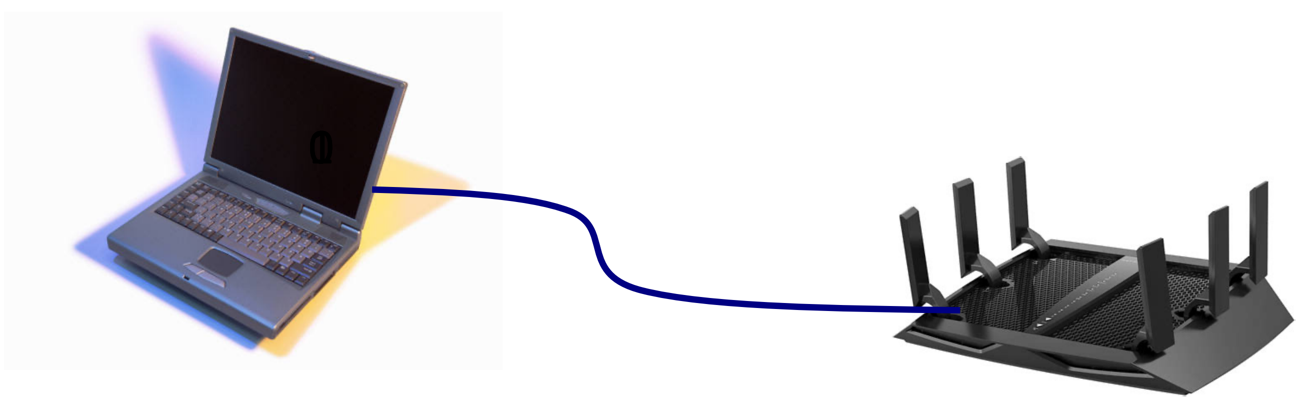
Focus:

How do we send a message across a wire?

The physical / link layers:

1. Different kinds of media
2. Encoding bits, messages
3. Model of a link

Application
Presentation
Session
Transport
Network
Data Link
Physical



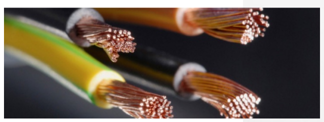
Network Link

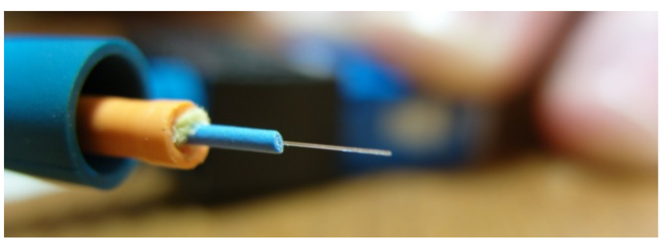
A Communication Medium

and

A Network Adapter

Or a NIC (Network Interface Card)

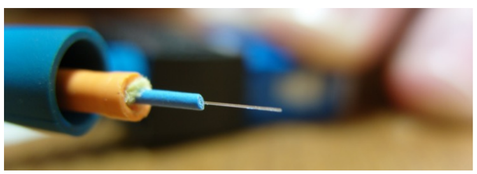




1. Different Types of Media

Wire

- Mostly copper wires, e.g., CAT6 Ethernet Cables
  - Different number of wires
  - Operation frequencies (16MHz to 2GHz)
- Shielding (insulating the wire)
- Distance: 10s of meters to kilometers
- 10Mbps to 100Gbps



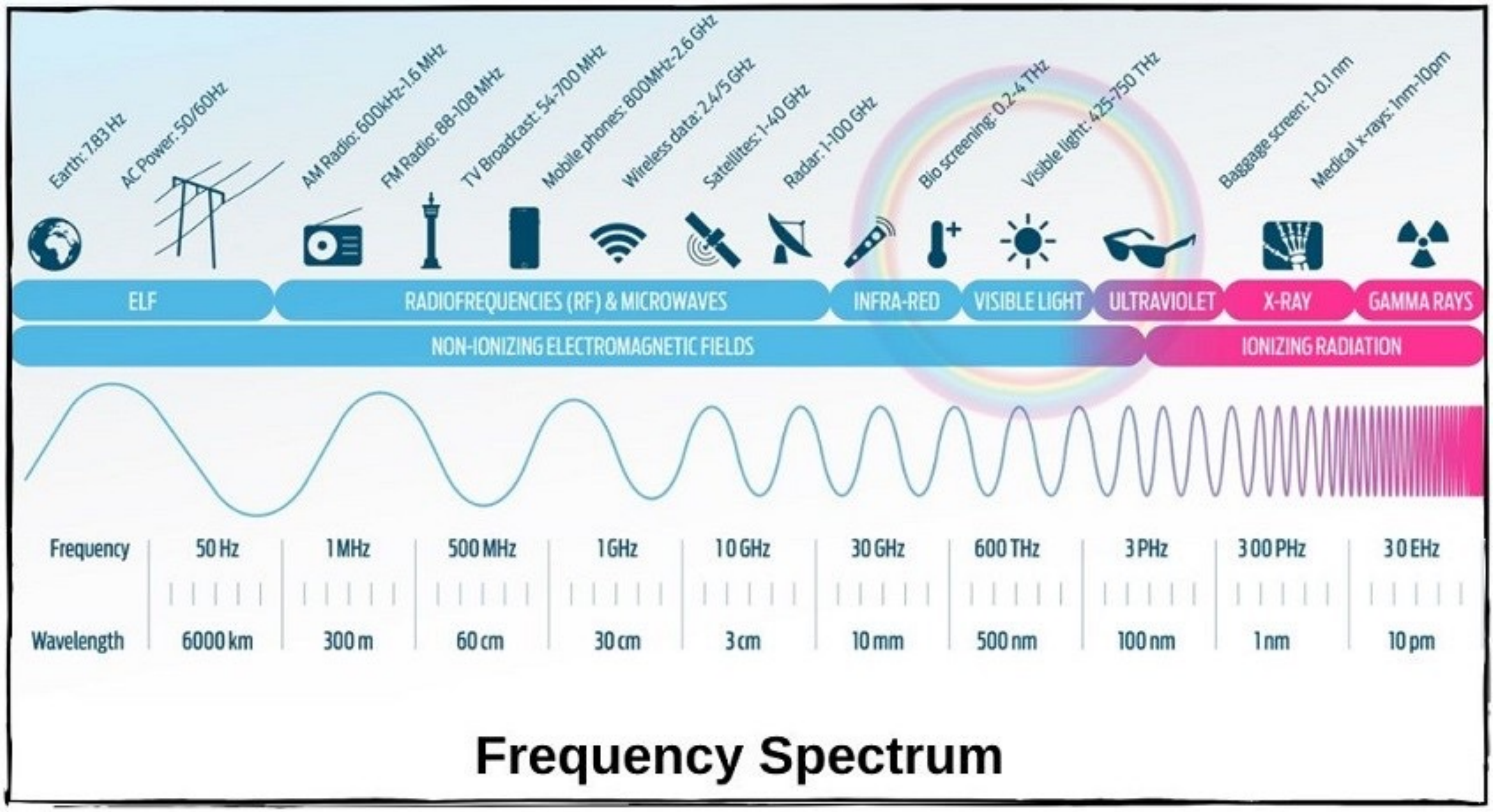
Fiber

- Multi-mode, e.g., 100Mbps, 2KM
- Single-mode, e.g., 10 Gbps for 80km

Wireless

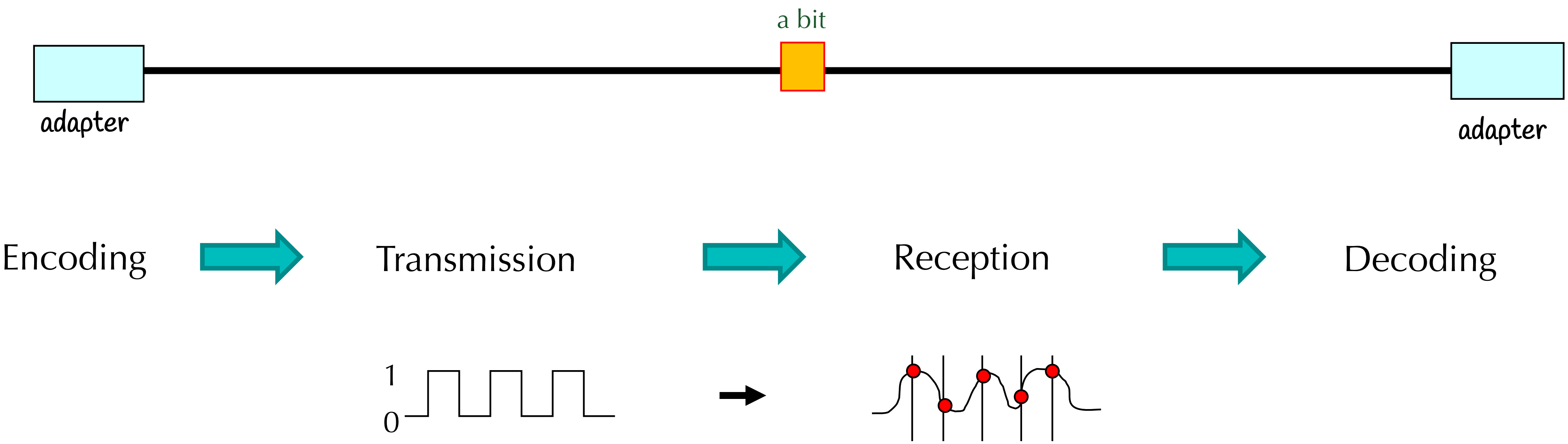
- Wireless LANs, e.g., WiFi (family of protocols based on IEEE 802.11 standards),
- Bluetooth, Microwave, satellite, cell phones, …

Range of Electromagnetic Waves



*https://ipcisco.com/lesson/wlan-frequency-bands/

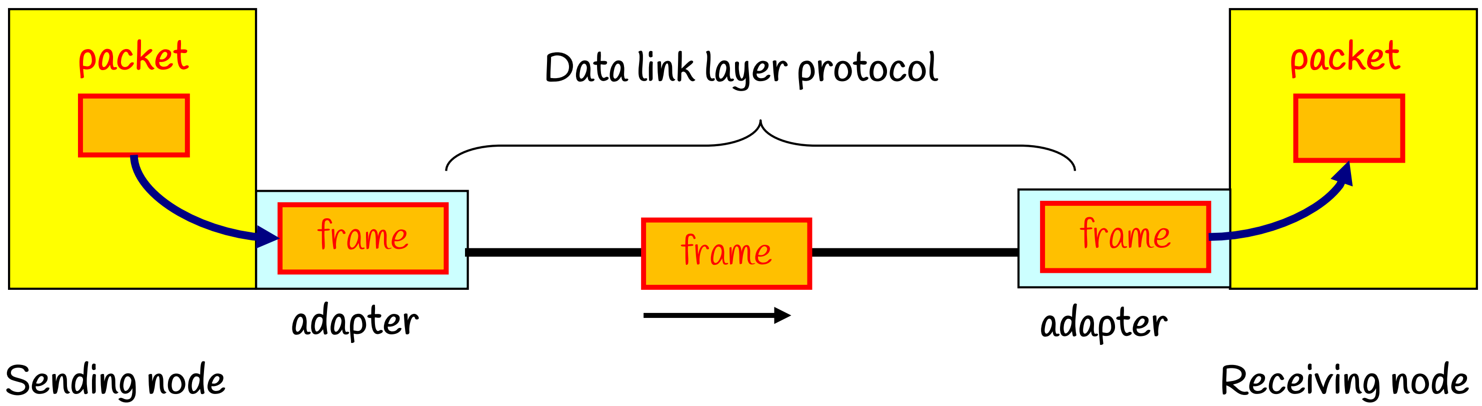
Physical Layer



- Encoding/Decoding:
  - Generate analog waveform (e.g., voltage) from digital data at transmitter and sample to recover at receiver
- We send/recover symbols that are mapped to bits
  - There are different ways of mapping bits to signals
    - E.g. NRZ, NRZI, MANCHESTER, 4B/5B

3. Framing

- Sending bits on links is good!
- But we need to send messages not bits.
  - Complete link layer messages are called frames.



Sending side
- Encapsulates packet in a frame
- Adds error checking bits, flow control, etc.

Receiving side
- Looks for errors, flow control, etc.
- Extracts datagram and passes to receiving node

Framing

- Main challenge:
  - When a receiver gets a sequence of bits, how should she know where the start and the end of a message/frame is?

- Common solution: Sentinels
  - Look for special *control codes* that mark start and end of frames
    - What if this control code appears in the main message?
    - How do you print a quotation mark in c?!
  - And "stuff" this control code within the data region
  - Example:
    - Sender always inserts a 0 after five 1s in the frame content
    - Receiver always removes a 0 appearing after five 1s

How do we identify link adapters?

Medium Access Control (MAC) Address

- Any adaptor is "built" with a Unique MAC address
  - Flat name space of 48 bits (e.g., 00-15-C5-49-04-A9 in HEX)
  - Works like a Social Insurance Number you get when you are born.

- Hierarchical Allocation
  - Blocks: assigned to vendors (e.g., Dell) by the IEEE
    - First 24 bits (e.g., 00-15-C5-**-**-**)
  - Adapter: assigned by the vendor from its block
    - Last 24 bits

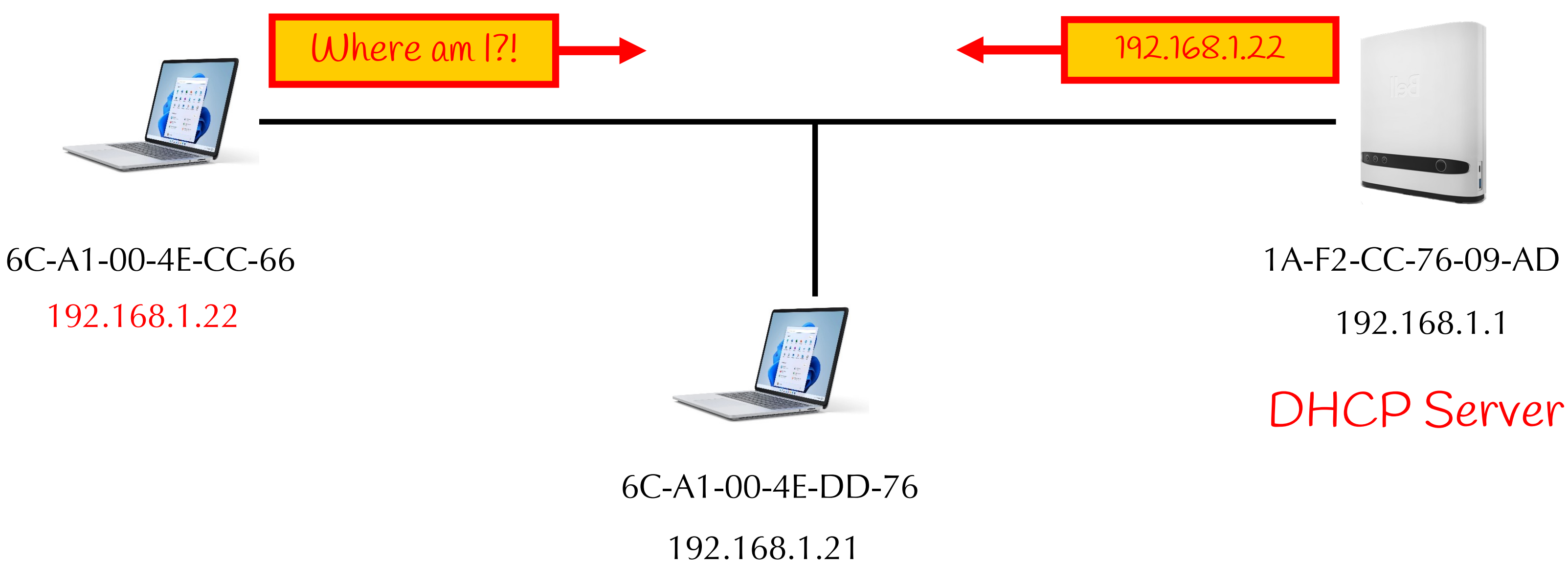
- Broadcast address (i.e., FF-FF-FF-FF-FF-FF)
  - Send the frame to all adapters

<u>Checkout your MAC address(es):</u>

- Linux, Mac: ifconfig
- Windows: ipconfig \all | findstr "Physical"

Bootstrapping …

- How do you get an IP address?
- Static: set the IP address manually (number got from your service provider); or
- Dynamic Host Configuration Protocol (DHCP)
  - Broadcast "I need an IP address, please!"
  - Response "You can have IP address 192.168.1.22"



Discovering the Receiver …

ARP Table

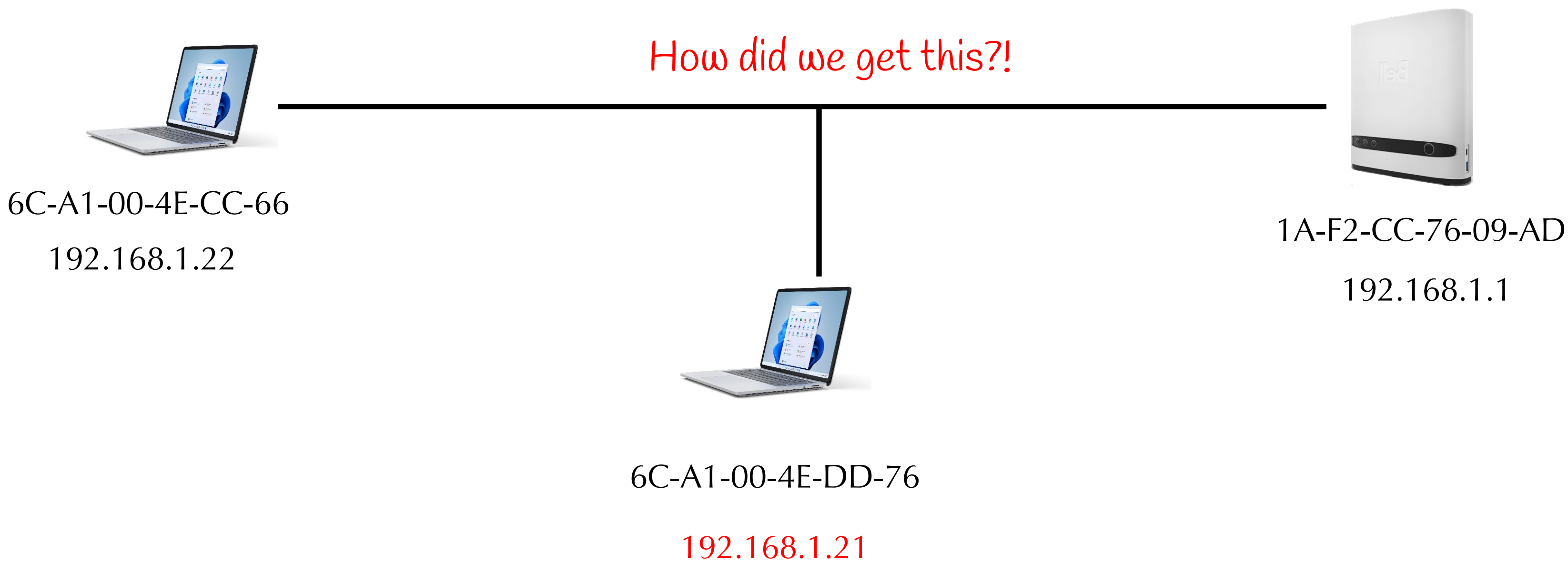
IP	MAC
192.168.1.1	1A-F2-CC-76-09-AD
192.168.1.21	6C-A1-00-4E-DD-76

I have an IP packet for 192.168.1.21
But I don't know her MAC address : (

Address Resolution Protocol (ARP)

- Broadcast "who has IP address 192.168.1.21 ?"
- Response "6C-A1-00-4E-DD-76 has 192.168.1.21 !"

How did we get this?!

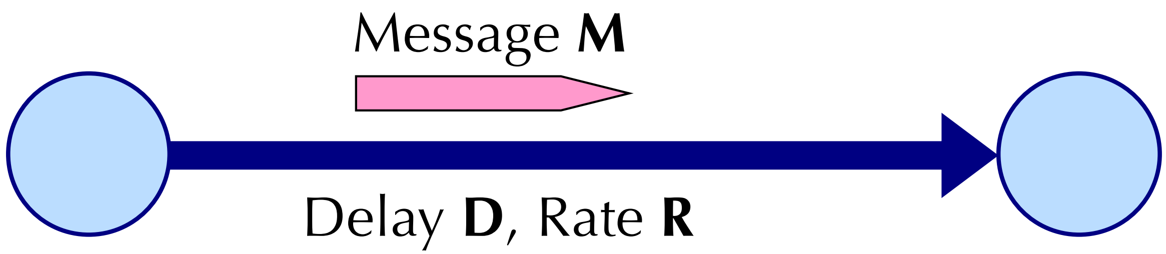


4. Model of a Link

- Abstract model is typically all we will need
  - What goes in comes out altered by the model
- Other parameters that are important:
  - The kind and frequency of errors
  - Whether the media is broadcast or not

Message Latency

- How long does it take to send a message?



- Two terms:
  - Propagation delay = distance / speed of light in media
    - How quickly a message travels over the wire
  - Transmission delay = message (bits) / rate (bps)
    - How quickly you can inject the message onto the wire
- Later we will see queuing delay …

Relationships

- Latency = Propagation + Transmit + Queue
- Propagation Delay = Distance/SpeedOfLight
- Transmit Time = MessageSize/Bandwidth

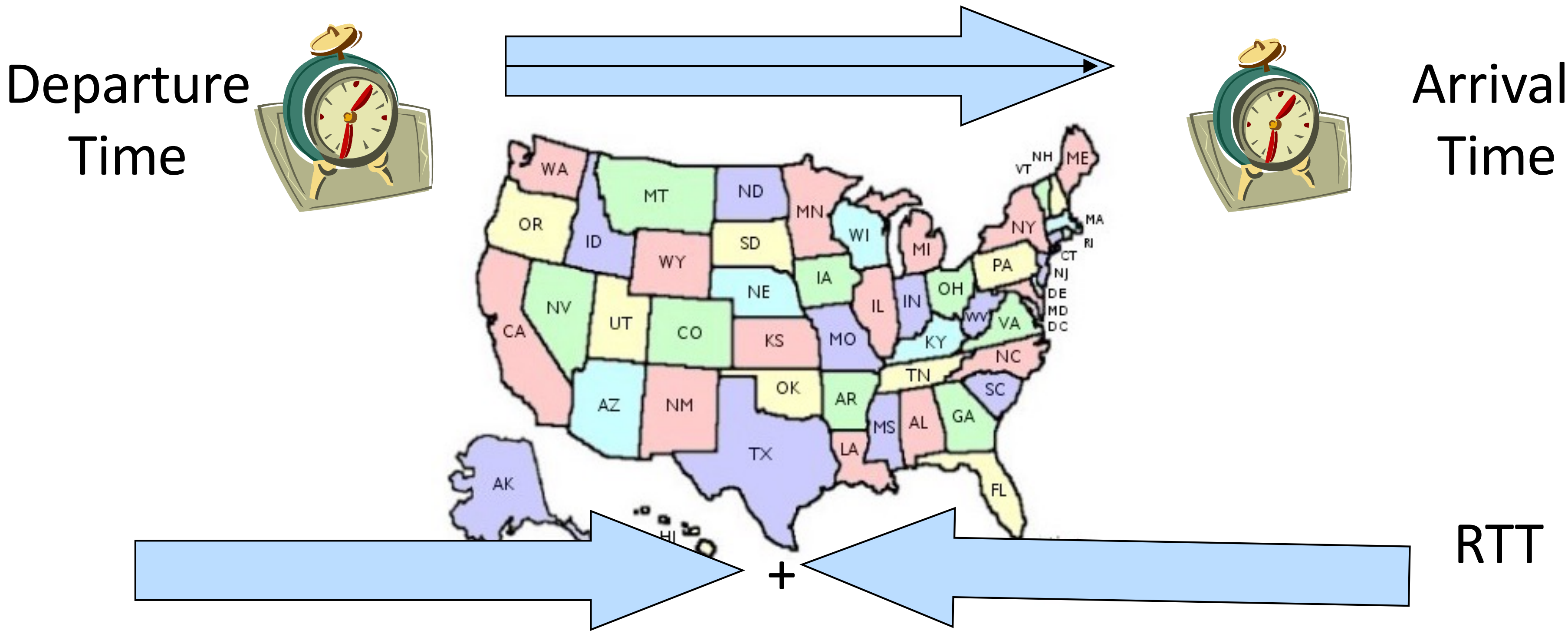
One-way Latency

- Dialup with a modem:
  - D = 10ms, R = 56Kbps, M = 1000 bytes
  - Latency = 10ms + (1000 x 8)/(56 x 1000) sec = 153ms!
- Cross-country with T3 (45Mbps) line:
  - D = 50ms, R = 45Mbps, M = 1000 bytes
  - Latency = 50ms + (1000 x 8) / (45 x 1000000) sec = 50ms!

- Either a slow link or long wire makes for large latency

Latency and RTT

- Latency is typically the one way delay over a link
  - Arrival Time - Departure Time



- The round trip time (RTT) is twice the one way delay
  - Measure of how long to signal and get a response

Throughput

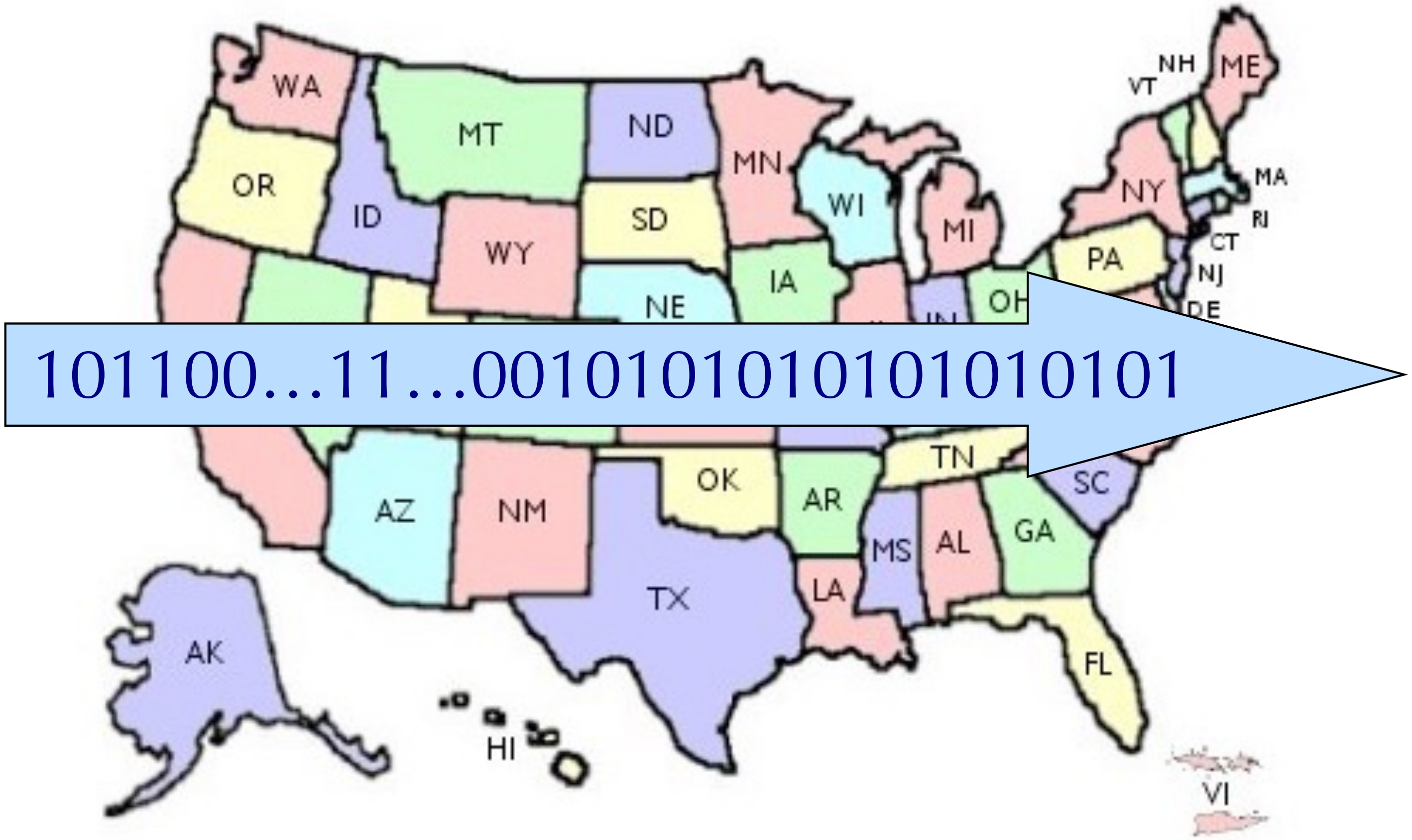
- Measure of system's ability to "pump out" data
  - NOT the same as bandwidth
- Throughput = Transfer Size / Transfer Time
  - Eg, "I transferred 1000 bytes in 1 second on a 100Mb/s link"
    - BW?
    - Throughput?
- Transfer Time = SUM OF
  - Time to get started shipping the bits
  - Time to ship the bits
  - Time to get stopped shipping the bits

Messages Occupy "Space" On the Wire

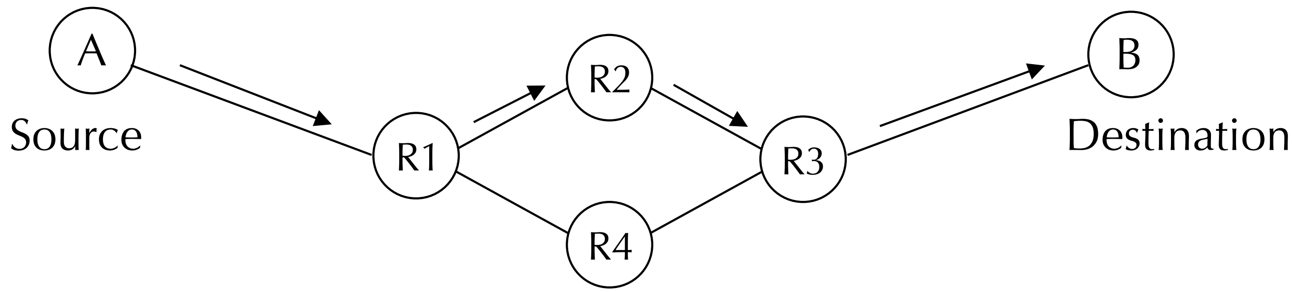
- Consider a 1b/s network.
  - How much space does 1 byte take?
- Suppose latency is 16 seconds.
  - How many bits can the network "store"
  - This is the BANDWIDTH-DELAY Product
  - Measure of "data in flight."
  - 1b/s * 16s = 16b
- Tells us how much data can be sent before a receiver sees any of it.
  - Twice B.D.P. tells us how much data we could send before hearing back from the receiver something related to the first bit sent.
  - Implications?

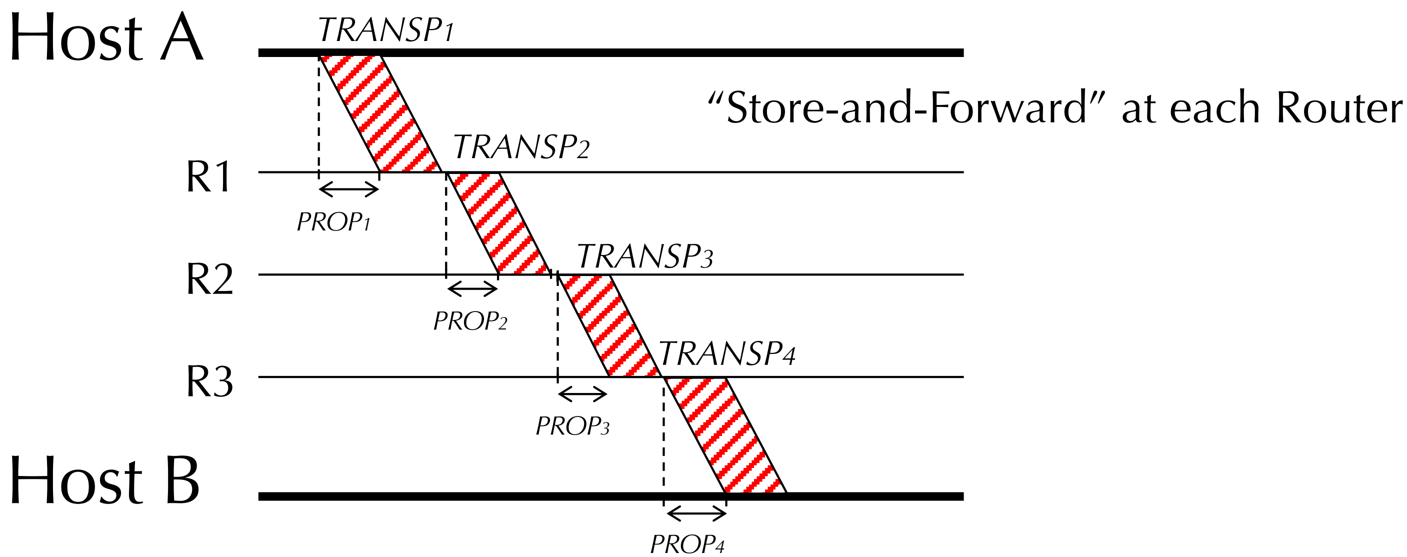
A More Realistic Example

BDP = 50ms * 100Mbps = 5Mb = 625KB



Packet Switching

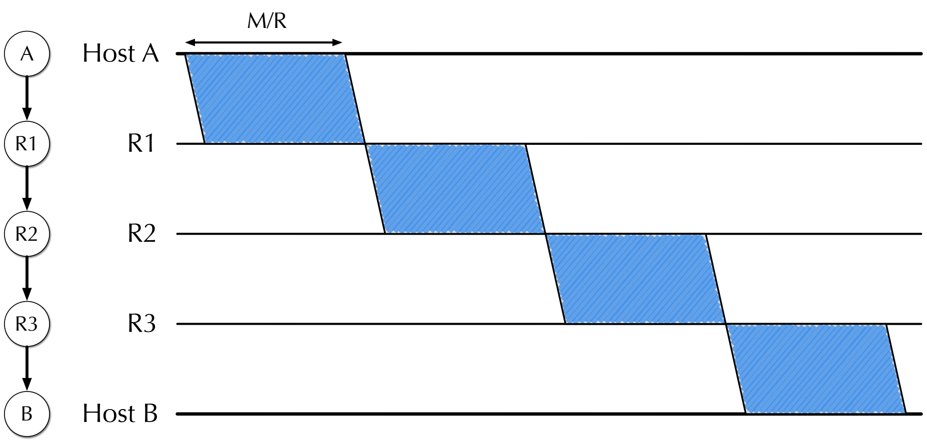




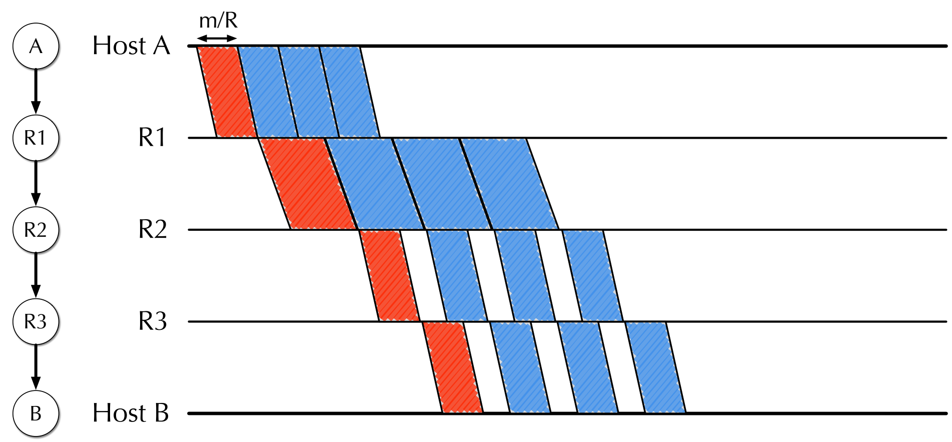
$$\text{Minimum end-to-end latency} = \sum_i (\text{TRANSP}_i + \text{PROP}_i)$$

Packet Switching

- *Why not send the entire message in one packet?*
  - *M is the message size*
  - *m is the packet size (M divided by number of packets)*
  - *K is the number of packets in the message.*



$$\text{Latency} = \sum_i (\text{PROP}_i + M/R_i)$$

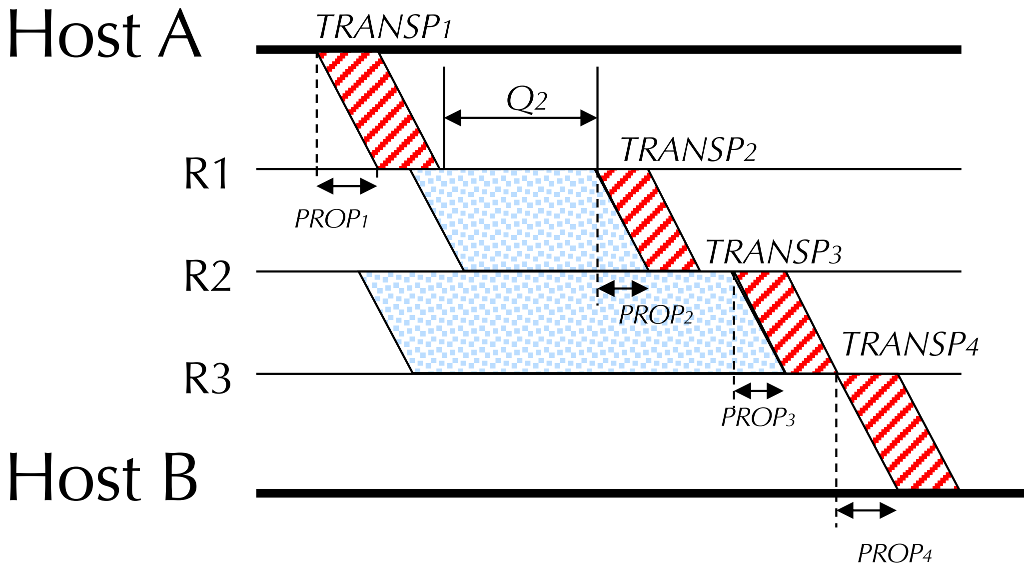


$$\text{Latency} = \sum_i (m/R_i + \text{PROP}_i) + m(K-1)/R_{\min}$$

Breaking message into packets allows parallel transmission across all links, reducing end to end latency. It also prevents a link from being "hogged" for a long time by one message.

Packet Switching – *Queueing Delay*

Because the egress link is not necessarily free when a packet arrives, it may be queued in a buffer. If the network is busy, packets might have to wait a long time.



How can we determine the queueing delay?

$$\text{End-to-end latency} = \sum_i (\text{TRANSP}_i + \text{PROP}_i + Q_i)$$

Increasing Link Bandwidth

- Typical approaches
  1. Increase number of wires (fibers, etc.)
  2. Increase bits per second

- Example:
  - NVLink 1.0:
    - 20 GB/s on each link in each direction
    - 8 lanes (links)
    - Total: 160 GB/s capacity
  - NVLink 4.0:
    - 25 GB/s in each direction
    - 18 links
    - Total: 1.8 TB/s

- This is a simplified model. The reality is a bit more complicated, but we'll get to it later.

Part 1: Key Concepts

- We typically model links in terms of bandwidth and delay, from which we can calculate message latency.
- Different media have different properties that affect their performance as links.
- We need to encode bits into signals so that we can recover them at the other end of the channel.
- Framing allows complete messages to be recovered at the far end of the link.

Outline

Part 1. Physical/link layer

- Different types of media
- Encoding bits with signals
- Framing
- Model of a link

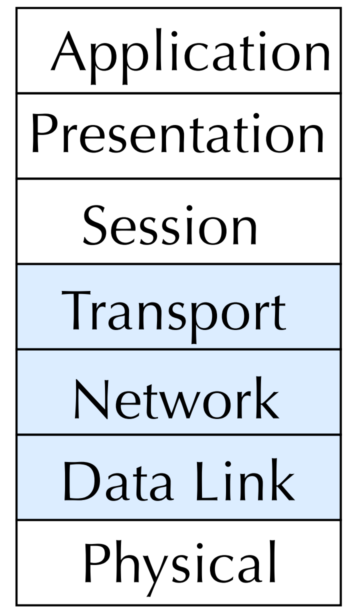
Part 2. Error detection and correction

- Hamming distance
- Parity, checksums, CRC, …

Part 2 – Error Detection and Correction

- Focus: How do we detect and correct messages that are garbled during transmission?

- The responsibility for doing this cuts across the different layers



Errors and Redundancy

- Noise can flip some of the bits we receive
  - We must be able to detect when this occurs!
  - Why?
  - Who needs to detect it? (links, routers, OSs, or apps?)
- Basic approach: add redundant data
  - Error detection codes allow errors to be *recognized*
  - Error correction codes allow errors to be *repaired* too

Motivating Example

- A simple error detection scheme:
  - Just send two copies. Differences imply errors.

- **Question**: Can we do any better?
  - With less overhead
  - Catch more kinds of errors
- **Answer**: Yes – stronger protection with fewer bits
  - But we can't catch all inadvertent errors, nor malicious ones

- We will look at basic block codes
  - K bits in, N bits out is a (N, K) code
  - Simple, memoryless mapping

Detection vs. Correction

- Two strategies to correct errors:
  - Detect and retransmit, or Automatic Repeat reQuest. (ARQ)
  - Error correcting codes, or Forward Error Correction (FEC)

- Satellites, real-time media tend to use error correction
- Retransmissions typically at higher levels (Network+)

- **Question**: Which should we choose?

Detect or Correct?

- Advantages of Error Detection
  - Requires smaller number of bits → low overhead.
  - Requires less/simpler processing.

- Advantages of Error Correction
  - Reduces number of retransmissions.

- Most data networks today use error detection, not error correction.

Retransmissions vs. FEC

- The better option depends on the kind of errors and the cost of recovery
- Example: Message with 1000 bits, Prob(bit error) 0.001
  - Case 1: random errors
  - Case 2: bursts of 1000 errors
  - Case 3: real-time application (teleconference)

Encoding to Detect Errors

- We use codes to help us detect errors.
- The set of possible messages is mapped by a function onto the set of codes.
- We pick the mapping function so that it is easy to detect errors among the resulting codes.
- Example: Consider the function that duplicates each bit in the message. E.g. the message 1011001 would be mapped to the code 11001111000011, and then transmitted by the sender. The receiver knows that bits always come in pairs. If the two bits in a pair are different, it declares that there was a bit error.
- Of course, this code is quite inefficient…

The Hamming Distance

- Errors must not turn one valid codeword into another valid codeword, or we cannot detect/correct them.
- Hamming distance of a code is the smallest number of bit differences that turn any one codeword into another
  - e.g, code 000 for 0, 111 for 1, Hamming distance is 3
- For code with distance d+1:
  - d errors can be detected, e.g, 001, 010, 110, 101, 011
- For code with distance 2d+1:
  - d errors can be corrected, e.g., 001 → 000

Hamming Distance

Number of bits that differ between two codes

e.g. 1 0 0 1 0 1 0 1

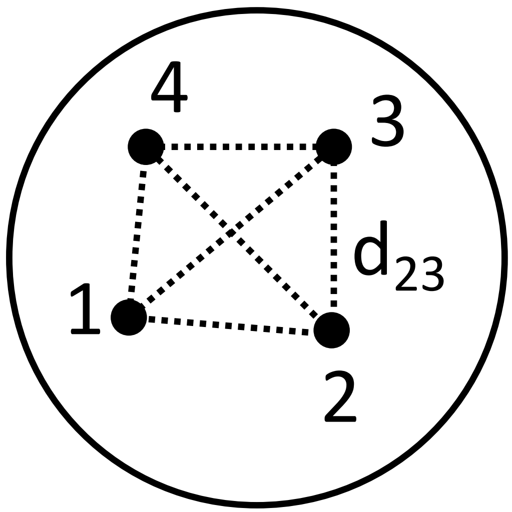
1 0 1 1 1 0 0 1

0 0 1 0 1 1 0 0 → HD=3

In our example code (replicated bits), all codes have at least two bits different from every other code. Therefore, it has a Hamming distance of 2.

Hamming Distance

Set of codes



HD = min (d_{ij})

To reliably detect a d-bit error: HD ≥ d+1

To reliably correct a d-bit error: HD ≥ 2d+1

Parity

- Start with n bits and add another so that the total number of 1s is even (even parity)
  - e.g. 0110010 → 01100101
  - Easy to compute as XOR of all input bits

- Will detect an odd number of bit errors
  - But not an even number
- Does not correct any errors

2D Parity

- Add parity row/column to array of bits
- Detects all 1, 2, 3 bit errors, and many errors with >3 bits.
- Corrects all 1 bit errors

0101001	1
1101001	0
1011110	1
0001110	1
0110100	1
1011111	0
1111011	0

Checksums

- Used in Internet protocols (IP, ICMP, TCP, UDP)
- **Basic Idea**: Add up the data and send it along with sum

- **Algorithm**:
  - *checksum* is the 1s complement of the 1s complement sum of the data interpreted 16 bits at a time (for 16-bit TCP/UDP checksum)

- **1s complement:** flip all bits to make number negative
- **1s complement sum**: if there is a carry from the most significant bit (MSB), wrap it around and add it back to the least significant bit (LSB).

1s Complement Sum Examples

- Example 1
  - Number 1: 0110
  - Number 2: 0101

```
    0110
+   0101
--------
    1011
```

- No carry in this case, so no wrapping is needed.
- The sum is 1011.
- The 1s complement sum is 0100.

- Example 2
  - Number 1: 1110
  - Number 2: 0101

```
    1110
+   0101
--------
   10011
```

- The result is 5 bits (10011). The carry (1) is added to the least significant 4 bits:

```
    0011
+      1
--------
    0100
```

- The final sum is 0100.
- The 1s complement sum is 1011.

CRCs (Cyclic Redundancy Check)

- Stronger protection than checksums
  - Used widely in practice, e.g., Ethernet CRC-32
  - Implemented in hardware (XORs and shifts)

- Mathematics more involved …
  - Optional: see textbook if you are interested

Part 2: Key Concepts

- Redundant bits are added to messages to protect against transmission errors.
- Two recovery strategies are retransmissions (ARQ) and error correcting codes (FEC)
- The Hamming distance tells us how much error can safely be tolerated.