# COMP 307 Fall 2018
# Professor Joseph Vybihal

Based on Professor Joseph Vybihal's lecture notes and slides, some additions by author.

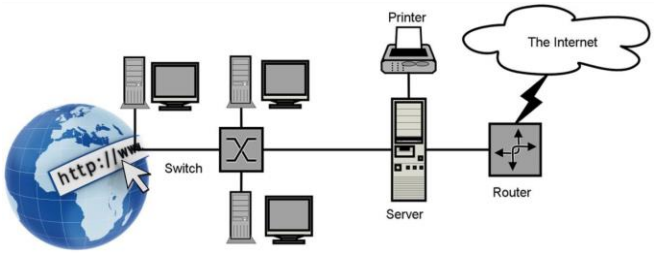
## Contents

# Lecture 1 – Wednesday, September 5, 2018

## History of Internet

The Internet is…

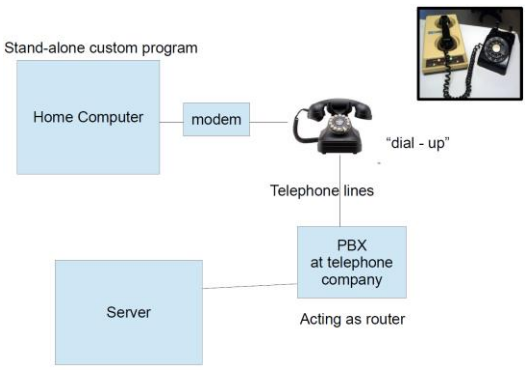



- Front end
- Back end
- Communication medium
- Security
- Technologies

## Engineered in Increments

- Developed over time (unfinished!)
    - Like any engineering project or software system the Internet was developed over time in small increments.
- Built in layers
    - Each new increment either built on top of a previous increment or replacing a previous increment.
- You will see...
    - Modern on top of archaic (a mixture of it! Best is kept and worst is removed…)
    - Speed and simplicity over fancy and pretty

## Before the Internet… (All of the cool kids had that!)



It was kind of like a bunch of servers connected via telephone. It was a lot of fun (at that time!) Had to know phone numbers (randomly)…

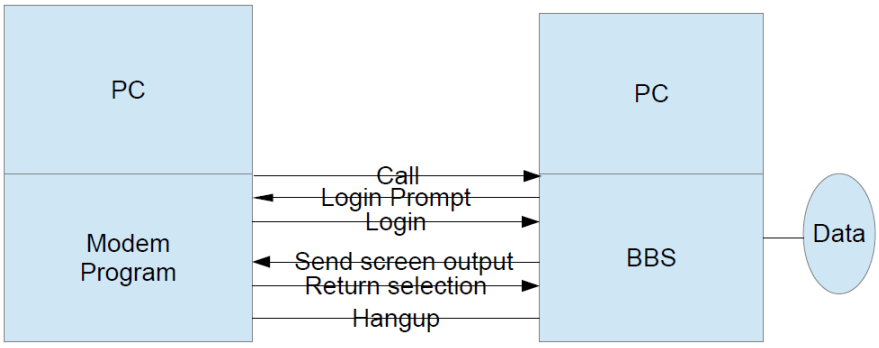
What came out of it are things we know very well. For example:

- Modem
  - Converted software data into sound (high for 1, low for 0) and transmitted over the regular telephone lines.
- Packets
  - Data was packaged together into a single “song” of sounds: source, dest, data, error bits.
- Phone numbers (Specific numbers for one client/server, now an IP address)
  - The telephone number was used for routing from source to destination computer. A dedicated connection.
- Strings
  - To keep it simple and easy to build, everything was ASCII strings.

## The BBS (Bulletin Board Service, 1980’s and 1990’s)


Remote connect with your modem to their server.
Just a window into their server.
Nothing executes on your end.

Behind it was very similar to what we experience now:




The Modem and BBS program interfaced with the PC hardware directly with little need for OS and server
The OS was used to read/write to files.

## Terminology

- Front-End
  - The part of the application that runs on the user's computer.
  - Old times:
    - The terminal: displays output from “server”
    - Read user input
    - Transmit user selection to back-end
- Back-End
  - The part of the application that runs on the provider's “server”.
    - Programs & data to process user's “requests”

## Search engines (1990s)

Google in text mode:


No mouse interface. Character data input at a prompt.
Still modem on client side…

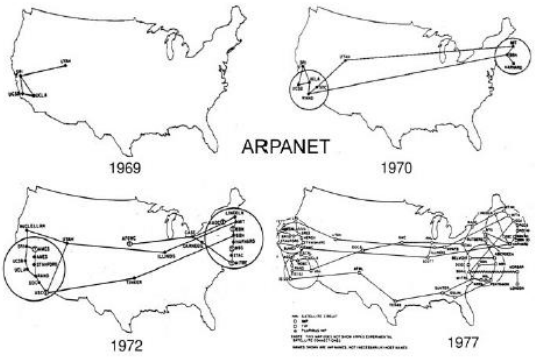
## The evolution of the Internet

### Beginnings of the Internet

- July 1961 – Leonard Kleinrock – MIT
  - Theory of packet-based networks
- August 1962 – J.C.R. Licklider – MIT
  - Theorized a “Galactic Network”
- During 1965 – Kleinrock, Roberts, Merrill
  - Low speed dial-up connection between MIT and California

- It was the first wide-area network
- In 1966 – Roberts went to DARPA
  (USA: Defense Advanced Research Projects Agency)

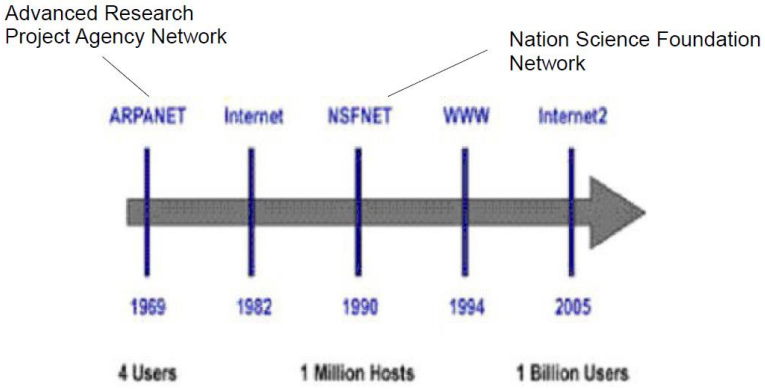
## ARPANET



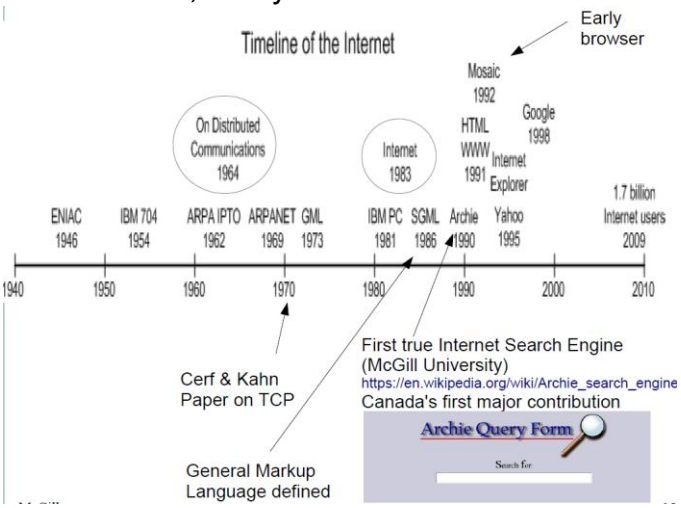

Based on a concept first published in 1967 (Roberts, Kleinrock) at DARPA (US Defense Advanced Research Project Agency), there ARPANET was developed under the direction of the U.S. Advanced Research Projects Agency (ARPA).

Built by: Truett Thatch, Bill Bartell (Honeywell), Dave Walden, Jim Geisman, Robert Kahn, Frank Heart, Ben Barker, Marty Thrope, Will Crowther, Severo Ornstein.

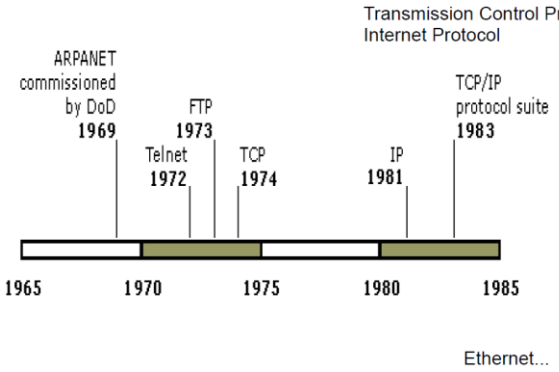
## Evolution




Another view, today:



## Network protocols




Ethernet...

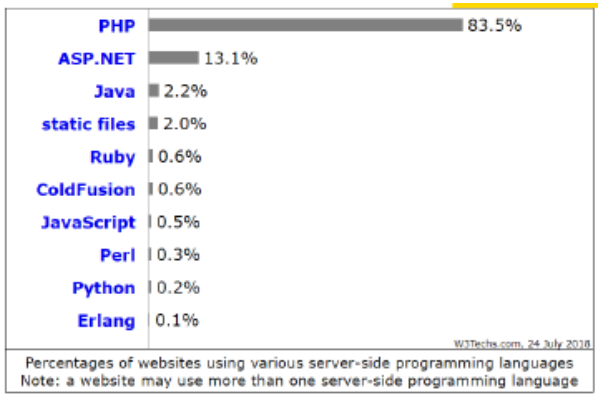
## Four Internet ground rules

- Each distinct network would have to stand on its own
- Communication would be on a best effort basis
    - If a packet did not get through it would be re-transmitted eventually
- Black boxes (gateways & routers) would be used to interconnect the networks
- No global control at the operational level

All these rules aim to make Internet stand alone and resistant to outside effort.

## The modern Internet (It's all about the "stack")

- A set of applications that work together as a software system to connect the front-end and back-end architectures.
- We will look at three in this course:
    - The XAMPP Stack (our main focus) (PHP)
    - The MEAN Stack (Java)
    - The Django Stack (Python)




Example stack:

- Common Stack
    - Client side
        - HTML
        - CSS
        - JS
    - Communication
        - CGI

        - JSON
    - Server side
        - REST server
        - PHP
        - SQL

A stack like this can create something like this (e.g. Facebook):


Facebook was initially a PHP framework a very, very long time.
If you become popular, it is a problem. A server can only handle a limited number of clients at a time.

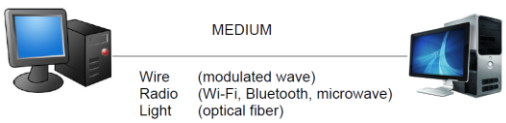
# Lecture 2 – Monday, September 10, 2018

## Basic network architecture

### Network

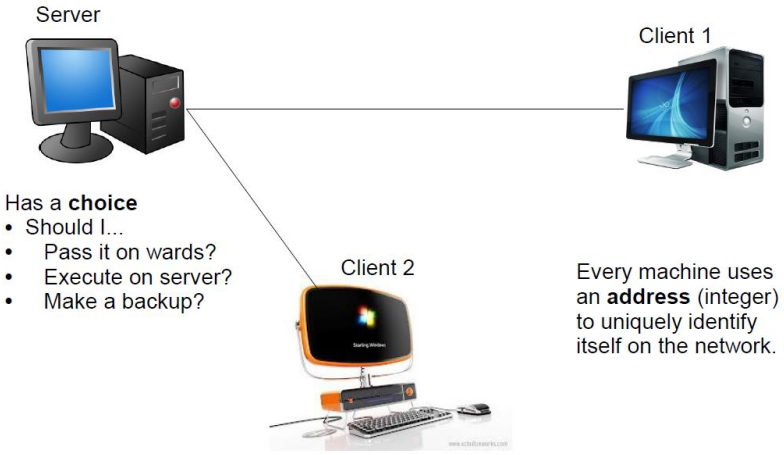
- A collection of machines interconnected via a **medium**.
- **Information** is passed between the machines using the medium.
- The information has a particular format and specific **protocols** are used to control the flow of information.

*A simple network (Peer-to-peer network, no server, just two PCs)*

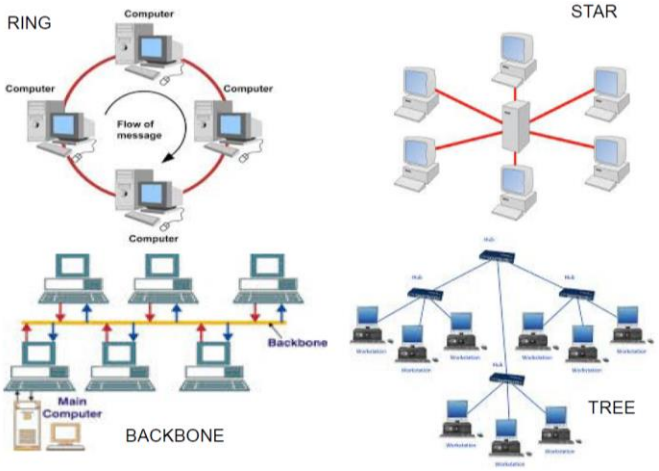


What would the **information** look like in each of these technologies?

If Client 1 sends a message to Client 2, the message must pass through the server. There are no other wires.

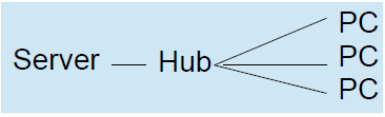



*Topology (Who sees what?)*

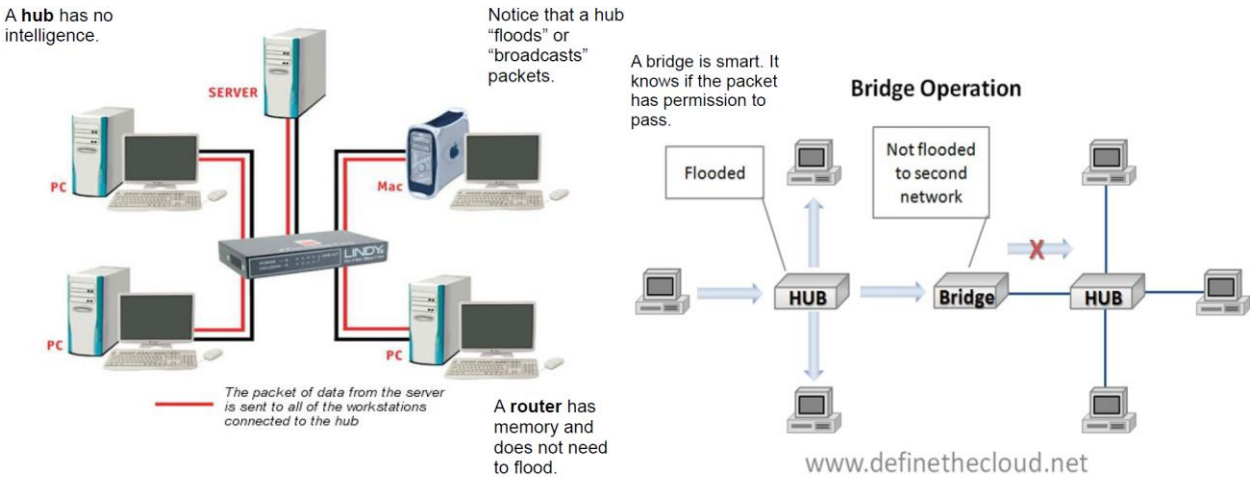



Network components

- Wire
  - The medium by which information travels
- Network card or modem
  - Converts strings to signals compatible with medium
- Hub or Router
  - A box that splits an input wire into multiple output wires. Hubs use broadcasting. Routers pick a path.



- Server
    - A master computer responsible for managing the communication and sharing between devices.
- ISP
    - A server responsible for providing Internet interconnection, routing, addressing and traffic control.
- Bridge
    - A server responsible for translating packets of one format into another format. Facilitates communication between different networks.
- Gateway
    - A server that provides a connection exiting your local area network (example, an ISP).

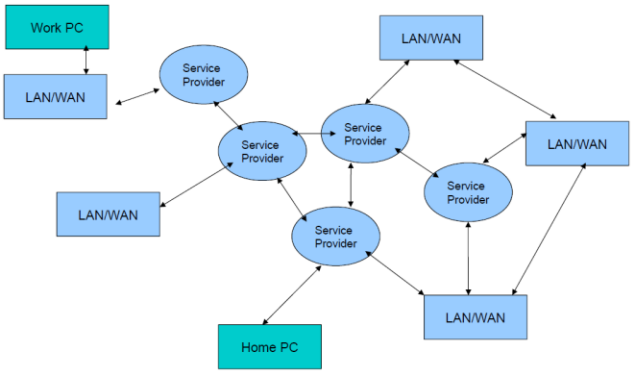



ISP (Internet service provider)

- A special server that
    - Has members
    - Provides URL resolution to IP address
    - Has routing tables
        - To calculate shortest path
        - Or at least, next best hop
- Can broadcast
    - By choice
    - Or when it does not know what to do
- Can do simple load balancing
- Can do simple traffic avoidance

## Internet topology

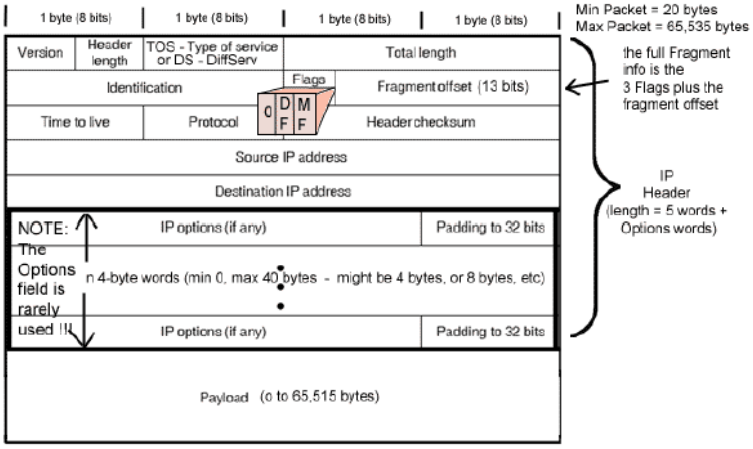
How does email work? (Storage nodes, readability, security…)



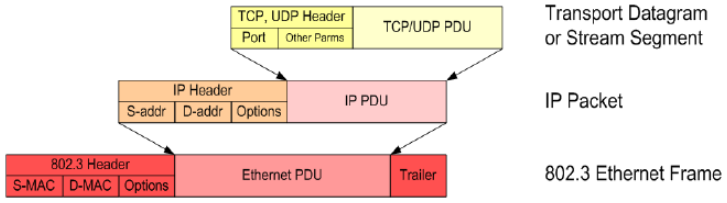

## Packets

- Data converted into “music” (e.g. Wi-Fi)
- Simple packet contains:
  - Source address
  - Destination address
  - Message
  - Error correction bits
- As a string: “address:address:message:correction”

- A data structure used to store data for transmission from point A to point B




## Nested structures

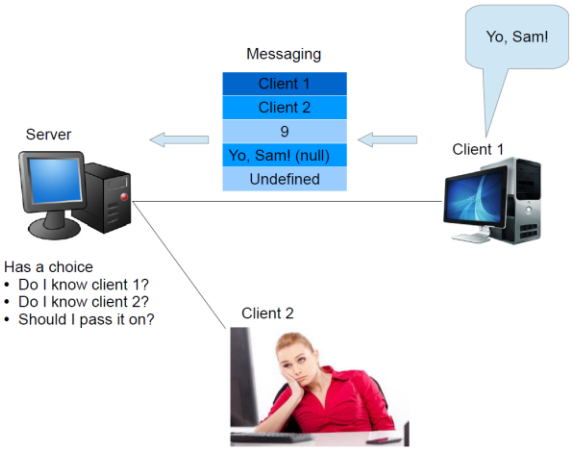



Important picture!

- Bottom: The frame with control information
- Middle: The packet with control information
- Top: The message with control information

Why so much control information?
Can you identify what the control information is about?

Example:

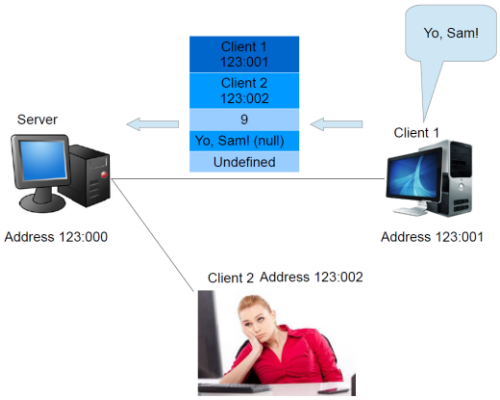



## Packet elements

### *Addresses*

- MAC Address
  - MAC = Media Access Control (physical)
- IP Address
  - IP = Internet Protocol (logical)
- Every device comes with a MAC. But you set the IP.

Example:




*Data*

ASCII table:

# ASCII TABLE

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [ENG OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

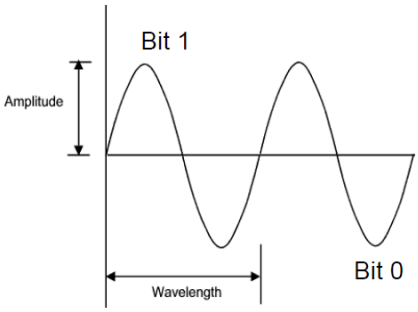
“hello” → 104, 101, 108, 108, 111, 0a

Data to digital

Binary: 104 → 01101000
“hello” → 01101000, 01100101, …, 00000000

Data to signals

[a] NULL, termination character




What will “hello” look like?

## Data modulation

- Instead of a simple sine wave to digitize data… Use multiple frequencies
    - Two frequencies: high=1, low=0
    - Four frequencies:
        - Very high: 11
        - High: 10
        - Low: 01
        - Very low: 00

      (Notice data travels twice as fast)
    - Eight frequencies, 16, etc.
        - Need very sensitive equipment
        - Eventually not cost effective, an upper limit exists…

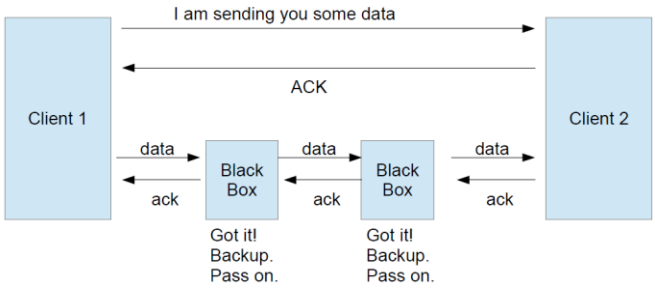
## Modems and network cards

- Step 1: Convert string to digital signal
- Step 2: Modulate for speed
- Step 3: Broadcast transmission
    - It transmits the signal down the wire (medium)

## Protocols

- An algorithm that describes how to transmit data.
- There are many protocols. In this course, we will only look at the most common.
- Today:
    - End-to-End protocol
    - Hop-to-Hop protocol

## End-to-End vs. Hop-to-Hop: a comparison




## End-to-End protocol

- A protocol that informs the source and destination computer of the status of a message.
- Message = one packet
- Big Messages = N packets
    - Each segment identified by a "segment" number, which is simply a sequence number from 0 to N–1.
- Source & destination want to know if all packets arrived.

*Algorithm:*

### Source

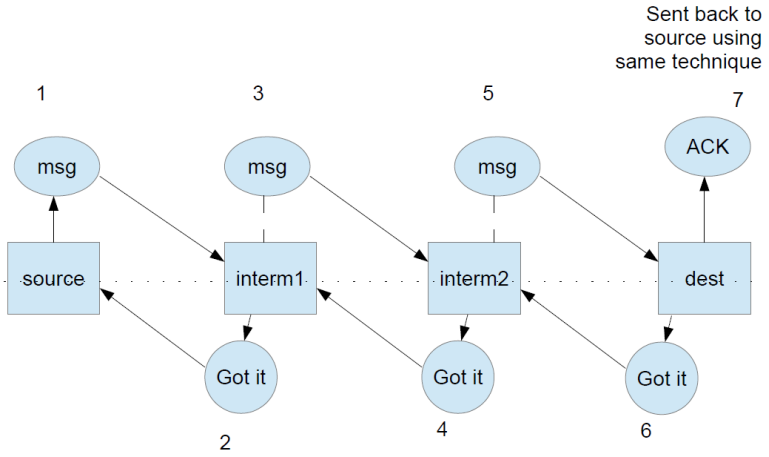
- Try 3 times:
    - Send(# of segments)
    - Wait for ACK or fail
- Try 3 times:
    - Send a segment
    - Wait for ACK
    - Timeout? Resend
- Terminate when all segments sent & ACK received

### Destination

- Wait infinite
- On receive initial:
    - Check # segments
    - Send ACK or ERR
    - Start wait timer
- Wait for segment:
    - Store & sort & ACK
    - Timeout? Prompt
- 3rd prompt, Fail.
    - Corrupt? Err

## Hop-to-Hop protocol

- The source and destination computers are normally not directly connected to each other. Instead there are often many intermediate computers/servers.
- A packet or segment must pass through these intermediate computers.
    - This game of "hot potato" needs to be managed.
    - Traffic? (is there a better path?)
    - Lost? (the packet never arrived!)
    - Damaged? (unable to understand the packet)




Timers are set on each confirmation packet to make sure it gets there, 3 tries.

## Protocol classes

- Application protocol
  - End-to-end
  - **Client** 1 app talks to Client 2 app
  - The application does not care about the route
- Network protocol
  - Hop-to-hop
  - The **black boxes** that manage the routing of the packet from source to destination (80% status, 20% data)
  - Hidden from the application
  - They send **status** packets to each other