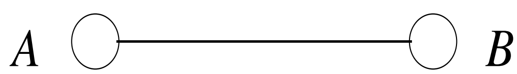
Fundamentals of information transmission

⟶ applies to both wired and wireless networks

⟶ wireless-specific features discussed separately

Sending bits using physical signals

Simplest case: hosts A and B are connected by point-to-point link



→ e.g., A wants to send bits 011001 to B

Choices for physical signals

- sound waves: air pressure changes
- underwater sonar: water pressure changes
- light: electromagnetic waves
- what else?

Preferred mode for data communication:

→ electromagnetic (EM) waves

→ low latency (SOL) and large bandwidth (bps)

→ some undesirable properties too

What is an electromagnetic wave?

→ in principle: a complicated question involving quantum mechanics

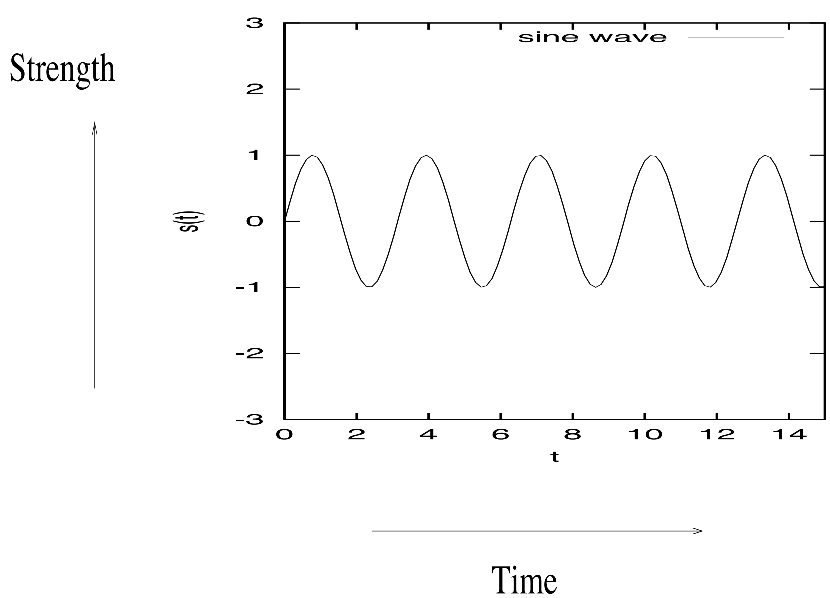
→ still part of physics and engineering research

In today's systems: only straightforward EM features are exploited

View EM as a physical phenomenon/object which has a strength (or magnitude) that may vary over time.

In simple form, a measurable quantity (or magnitude, amplitude, power, energy) varies in a regular fashion.

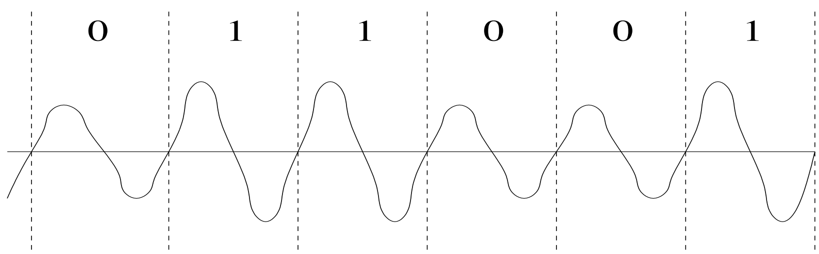
→ i.e., oscillating sine curve



Back to original problem: A wants to send B six bits 011001

→ use magnitude of sine waves

high amplitude represents 1, low amplitude 0 (or vice versa)



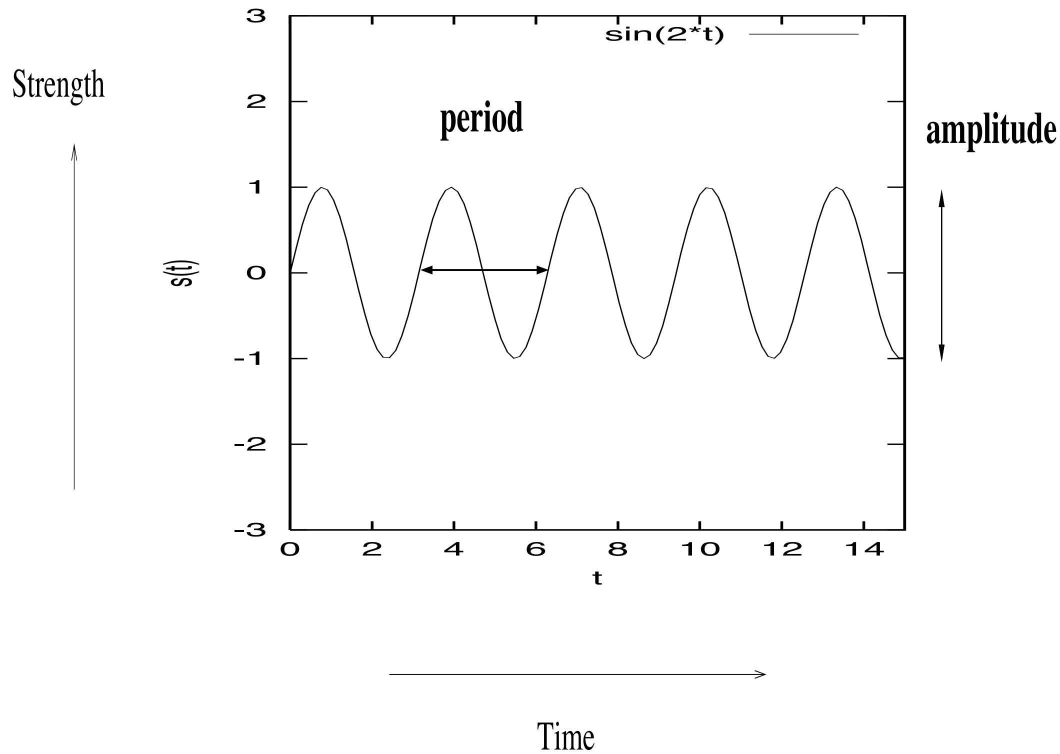
called amplitude modulation (AM)

→ i.e., modulate/manipulate amplitude to send bits

→ same concept as AM radio

→ difference?

Three features of EM as sinusoid:



→ period (also called cycle): T

→ strength: amplitude

→ phase: shift in time

How to utilize these features to communicate bits …

→ beyond binary AM

Basic properties of sinusoids:

How many periods can we squeeze in per second?

→ frequency: $1/T$

→ e.g., if period is 1 msec then frequency is 1000 cycles/sec

→ unit called Hertz (Hz)

Another unit: length (m)

→ distance

→ how long is a period

→ i.e., footprint in space

→ empty space: e.g., 1 GHz EM sinusoid about 11.8 inches long

→ fiber optic cable?

In computer networks, by default, frequency is used to specify EM

→ sometimes period is used (esp. high frequency, e.g., 100's of GHz plus)

Example: benefit of using frequency for AM to calculate bps

⟶ bandwidth (bps) of point-to-point link

→ if frequency is 1 Hz then bandwidth 1 bps

→ if 1 MHz then 1 Mbps

→ if 1 GHz then 1 Gbps

→ if 1 THz then 1 Tbps

Networking problem solved!

→ not quite

Issues with increasing frequency:

One: increasing frequency requires increase in clock rate and processing speed

→ high cost

→ computing systems that control hardware operate at lower speeds

→ heavy lifting: computation

Two: wireless propagation

→ above 10 GHz requires line-of-sight (LOS)

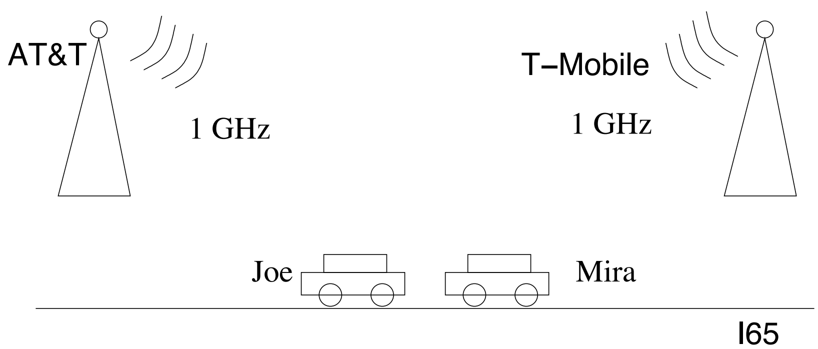
→ complications due to multi-path propagation

→ echos can be bad (and sometimes good)

Three: multi-user communication

→ not just point-to-point links connecting two parties

Example: wireless interference



Joe receives bits from AT&T's cell tower, Mira from T-Mobile.

→ Joe also hears T-Mobile's signal, Mira hears AT&T's signal

→ interference

→ What does Joe's smart phone actually hear?

Joe's device hears the sum of the two signals

→ property of electromagnetic waves

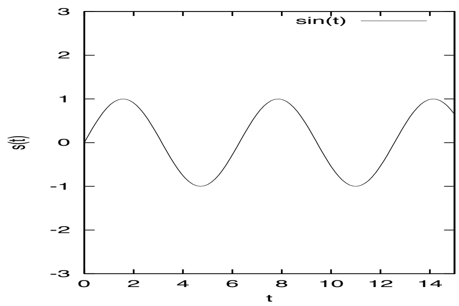
→ i.e., superposition

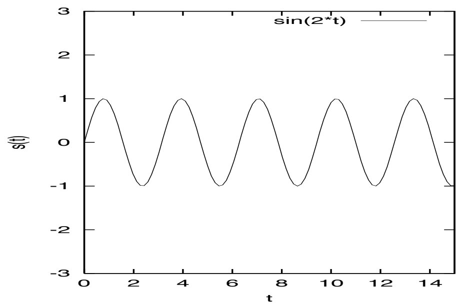
→ fundamental physics: linear

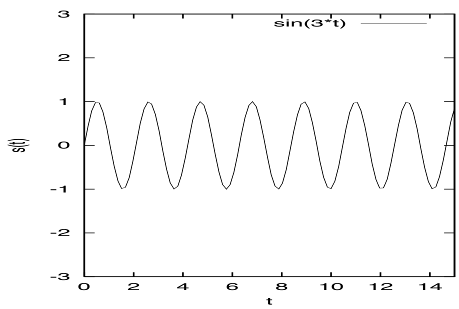
→ amenable to analysis and manipulation

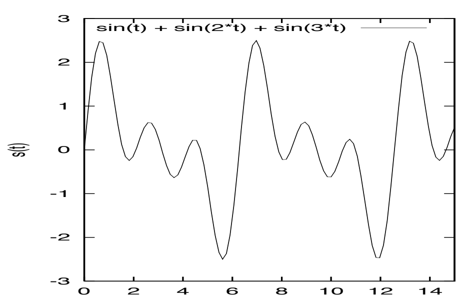
→ basis for modern computer networks

Superposition of three sine waves:

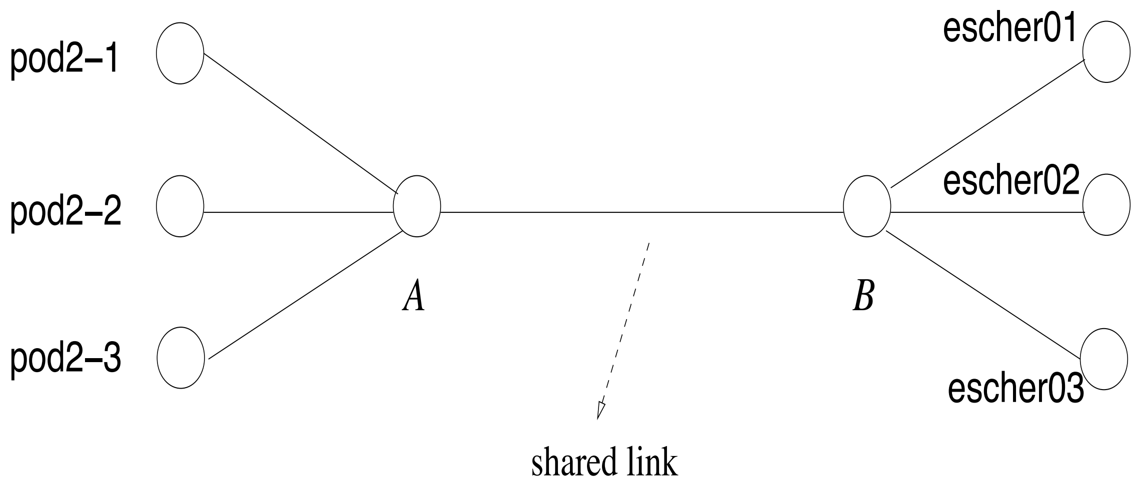








Example: multiplexing (i.e., intentional sharing of resources)



→ LWSN B148/HAAS G56 machines: A and B are Ethernet switches

→ A and B are routers/switches that forward multiple traffic streams

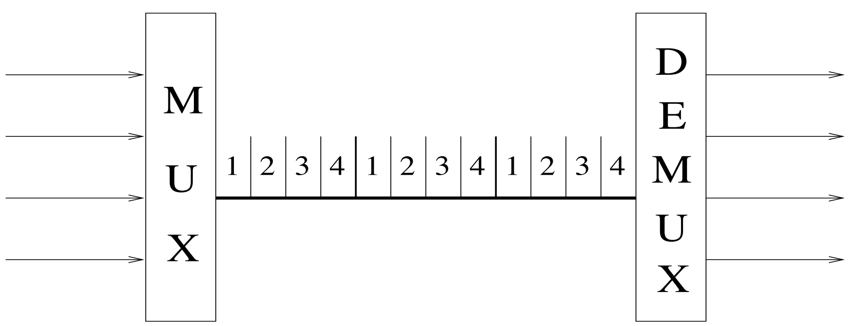
→ structured, orderly access

Splitting time based on AM method of sending bits using sine waves:

→ time-division multiplexing (TDM)

Ex.: four bit streams sharing same link

→ reserve time slots for each bit stream



→ user 1 gets slots 1, 5, 9, etc.

→ user 2 gets slots 2, 6, 10, etc.

→ router A: acts as multiplexer (MUX) or combiner

→ router B: acts as demultiplexer (DEMUX) or splitter

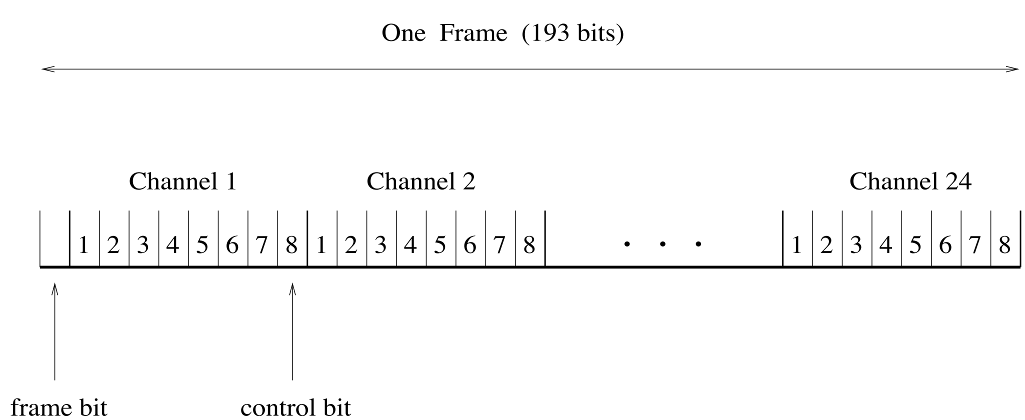
TDM, or TDMA (time-division multiple access) when slots belong to multiple users, is popular in cellular systems and traditional landline telephone systems.

→ simple but fundamental sharing technique

Real-world TDMA example from wired world:

→ T1 carrier (1.544 Mbps)

→ goal: support 24 simultaneous users ("channels")



Specs of T1 carrier:

- 24 channels (i.e., users)
- time slot: 8-bit block (each user sends 8 consecutive bits)
- $24 \times 8 = 192$ bits of payload
- plus 1 control bit: total 193 bits in a frame (unit of packaged data)
- squeeze 8000 frames into 1 second time interval

  $\rightarrow$ frame duration: 125 μsec
- total bandwidth (bps): $8000 \times 193 = 1.544$ Mbps
- per channel bandwidth (bps): $8000 \times 8 = 64$ Kbps

  $\rightarrow$ landline quality telephony service

At one time, popular service sold by ISPs (mainly) to companies

→ 20+ years back, Purdue leased about 6–7 T1 lines for the entire WL campus

→ next level T3 line: 44.736 Mbps

Today: residential subscriber can get 1 Gbps or faster download speed

→ uplink: significantly slower

→ bandwidth asymmetry

→ reflects client/server environment

TDMA: important multi-user link transmission technology

$\rightarrow$ works well if resource (e.g., frequency) is managed by central authority

$\rightarrow$ single provider

$\rightarrow$ otherwise: complications

What we want: parallel lanes where multiple bit streams are transmitted simultaneously

→ essence of modern high-speed networks

→ key technology: use multiple frequencies

→ e.g., 1 GHz and 2 GHz for two parallel lanes

How does using multiple frequencies for multiple lanes work?

→ classical method

→ improvements: our goal

→ modern broadband networks

Roadmap:

- start with CDMA

  → focus on coding: symbol processing

  → conceptual basis for analog methods

- move on to FDMA

  → use analog signals (sinusoid) to send parallel bit streams

  → classical method

  → limitations

- arrive at OFDM (orthogonal frequency division multiplexing)

  → extend FDMA to squeeze in more parallel lanes

  → increase bandwidth (bps)

CDMA motivation: linear algebra approach for sending multiple bit streams

Example: three users Alice, Bob, Mira

→ simplest case: cell tower wants to send each user 1 bit

→ but not TDMA

Assign each user a 3-D vector: called code

→ (1,0,0) for Alice

→ (0,1,0) for Bob

→ (0,0,1) for Mira

To send bit value 1 to Alice, 0 to Bob, 1 to Mira:

→ broadcast vector (1,0,1) to everyone

→ trivial: not much gained

Allow negative values:

→ send (1,-1,1): 1 means 1, -1 means 0

In general: let positive value means 1, negative value means 0

Example: assign Alice, Bob, Mira code vectors

→ Alice: (1,-2,1)

→ Bob: (3,5,7)

→ Mira: (19,4,-11)

The code vectors are stored in their smart phones.

Cell tower transmits via broadcast: (17,-3,-17)

→ ignore how the cell tower transmits (17, -3, -17) via electromagnetic waves

→ upon receiving (17, -3, -17), how does Alice know what bit was sent?

Solution: Alice calculates dot product of received vector (17, -3, -17) with her code vector (1, -2, 1).

Definition of dot product: Given two 3-D vectors $x = (x_1, x_2, x_3)$ and $y = (y_1, y_2, y_3)$, their dot (or inner) product is

$$x \circ y = x_1y_1 + x_2y_2 + x_3y_3$$

For Alice:

$(17, -3, -17) \circ (1, -2, 1) = 17 + 6 - 17 = 6 > 0$

$\rightarrow$ positive means bit 1

For Bob: $(17, -3, -17) \circ (3, 5, 7) = 51 - 15 - 119 = -83 < 0$

$\rightarrow$ negative means bit 0

For Mira: $(17, -3, -17) \circ (19, 4, -11) = 323 - 12 + 187 = 498 > 0$

$\rightarrow$ positive means bit 1

Why does this work?

$\rightarrow$ what is special about (1,-2,1), (3,5,7), (19,4,-11)

$\rightarrow$ where did (17,-3,-17) come from

The three code vectors are mutually orthogonal: $x \circ y = 0$

$\rightarrow (1, -2, 1) \circ (3, 5, 7) = 3 - 10 + 7 = 0$

$\rightarrow (1, -2, 1) \circ (19, 4, -11) = 19 - 8 - 11 = 0$

$\rightarrow (3, 5, 7) \circ (19, 4, -11) = 57 + 20 - 77 = 0$

Cell tower's job: send 1 to Alice, 0 to Bob, 1 to Mira

Cell tower computes (17, -3, 17) to broadcast where

$$(+1) \cdot (1, -2, 1) + (-1) \cdot (3, 5, 7) + (+1) \cdot (19, 4, -11) \\ = (17, -3, 17)$$

When Alice performs dot product of received vector (17,-3,-17) with her code vector (1,-2,1), it is equivalent to

$$\Big\{(+1)\cdot(1,-2,1)+(-1)\cdot(3,5,7)+(+1)\cdot(19,4,-11)\Big\} \circ(1,-2,1)$$

By orthogonality, the second and third terms vanish and what is left is

$\rightarrow (+1)(1,-2,1)\circ(1,-2,1) = 1+4+1 = 6 > 0$

$\rightarrow$ taking the dot product with oneself is always positive

For Bob:

$\rightarrow (17, -3, -17) \circ (3, 5, 7) = 51 - 15 - 119 = -83 < 0$

$\rightarrow$ negative means bit 0

For Mira:

$\rightarrow (17, -3, -17) \circ (19, 4, -11) = 323 - 12 + 187 = 498 > 0$

If we wanted the dot product for Alice to yield +1, Bob -1, Mira, +1, what to do?

Why might we not want the result to be 1, -1, 1, but 6, -83, 498?

CDMA (code division multiple access): using linear algebra, hide the bits to send in the coefficients of the code vectors.

→ in TDMA we divide time to transmit multiple bits in time slots

→ in CDMA, we "divide" code to transmit multiple bits as coefficients

→ coefficients are called spectrum

In CDMA, coding is used to encode multiple bits before transmission using electromagnetic waves occurs.

→ separate (analog) stage

→ omit here: will cover FDMA and OFDMA

→ driver in 90s-20s: QUALCOMM

→ e.g., Verizon, Sprint used CDMA in 3G cellular networks

→ retired in '22 and '23

Origin: military context (long history)

→ if code vectors are chosen to be random, additional feature of security (confidentiality)

Generalize:

To communicate n bits belonging to n users

- Set-up: assign n orthogonal code vectors in n-dimensional vector space

  $\rightarrow \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$

- Sender: to encode n data bits $a_1, a_2, \ldots, a_n$ (+1 for 1, -1 for 0), compute

  $\rightarrow \mathbf{z} = a_1\mathbf{x}^1 + a_2\mathbf{x}^2 + \cdots + a_n\mathbf{x}^n$

  $\rightarrow$ $\mathbf{z}$ is an n-dimensional vector that hides n bits in its coefficients (spectra)

  $\rightarrow$ convert $\mathbf{z}$ into analog signal and transmit to all receivers

- Receiver: to decode user i'th bit a_i, receiver computes dot product

  $\rightarrow$ $\mathbf{z} \circ \mathbf{x}^i = a_i(\mathbf{x}^i \circ \mathbf{x}^i) = a_i \times$ positive constant

  $\rightarrow$ by orthogonality

Next: borrow the conceptual framework from linear algebra for hiding bits in electromagnetic waves

$\rightarrow$ FDMA and OFDMA

$\rightarrow$ replace n-dimensional vectors with continuous complex sinusoids

$\rightarrow$ good news: much of the conceptual framework carries over