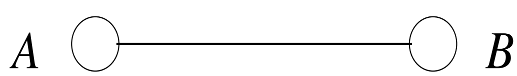
Fundamentals of information transmission

⟶ applies to both wired and wireless networks

⟶ special wireless features discussed later

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:

$\rightarrow$ electromagnetic (EM) waves

$\rightarrow$ why: it's fast (SOL) and other nice properties

$\rightarrow$ but has not-so-good-properties too

What is an electromagnetic wave?

$\rightarrow$ in principle: a complicated question involving quantum mechanics

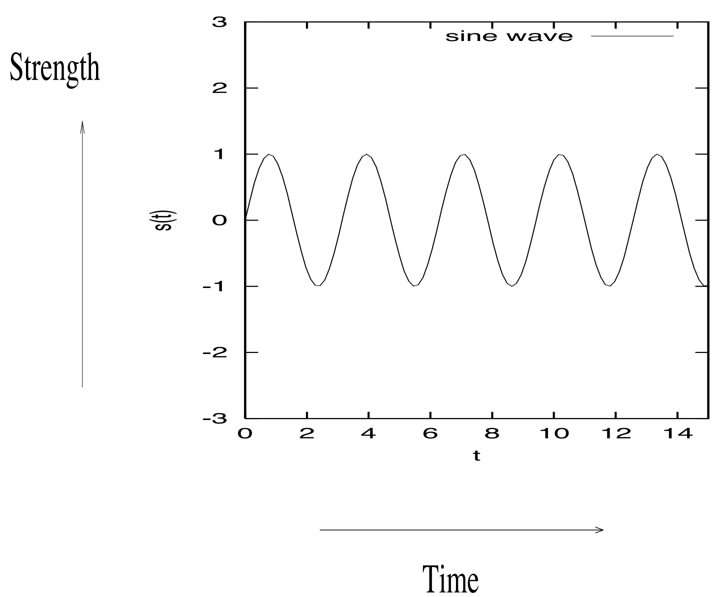
$\rightarrow$ still part of physics and engineering research

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

View EM as a "physical object" which has a strength (i.e., "loudness") that may vary over time.

In its purest form, the strength (or magnitude, amplitude, power, energy) varies in a regular fashion.

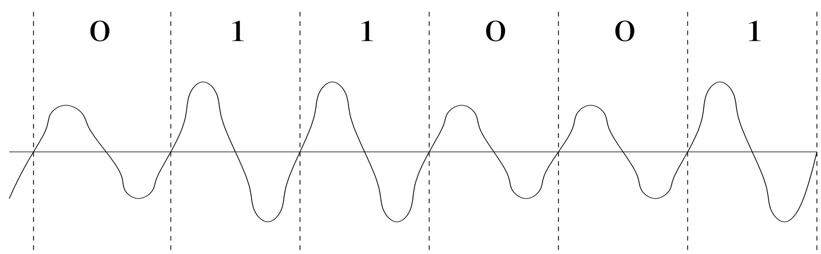
$\rightarrow$ i.e., oscillating sine curve



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

→ how do sine waves help?

utilize strength/amplitude to represent 1's and 0's



→ large amplitude: 1

→ small amplitude: 0

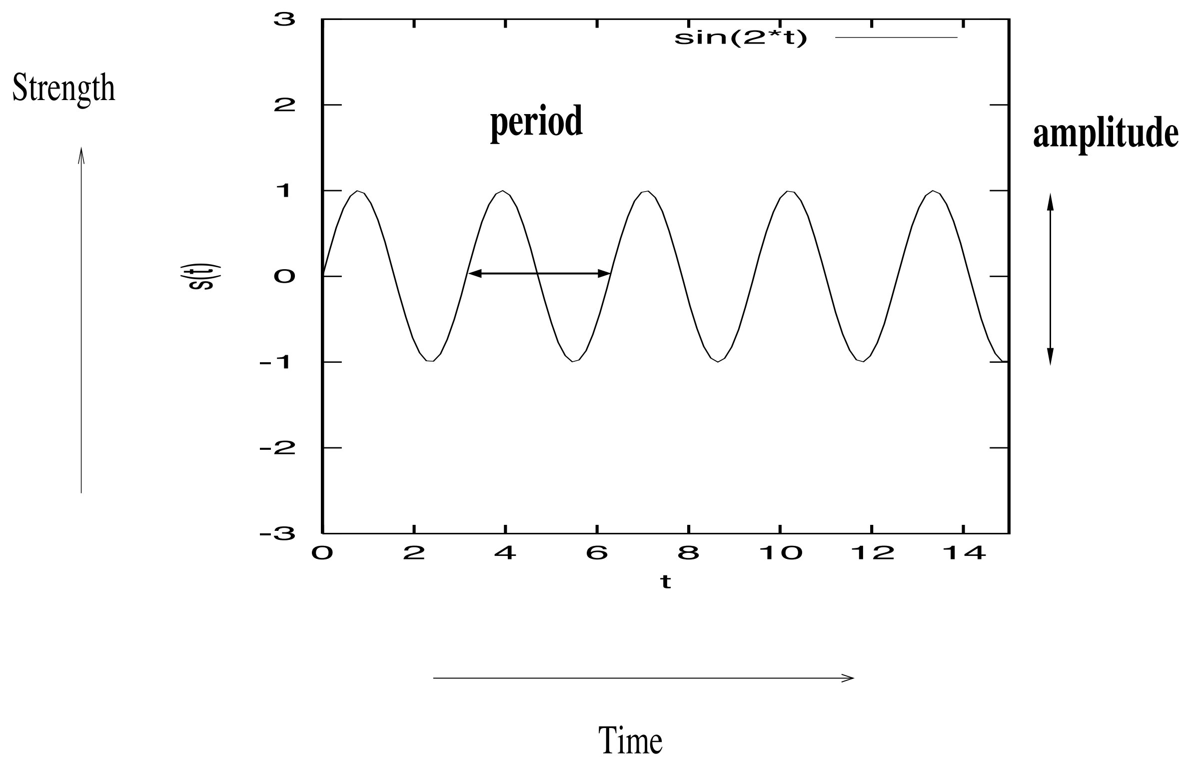
Method called amplitude modulation (AM)

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

→ same concept as AM radio

→ difference?

Three key features of EM:



→ period (also called cycle): T

→ amplitude

→ phase: i.e., shift in time

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)

In networks, often frequency is used to describe EM in place of period

→ one reason: allows easy translation to bandwidth (bps)

→ bps is of primary importance

Example: using AM to transmit bits from A to B

$\longrightarrow$ bandwidth (bps) of point-to-point link

$\rightarrow$ if frequency is 1 Hz then bandwidth 1 bps

$\rightarrow$ if 1 MHz then 1 Mbps

$\rightarrow$ if 1 GHz then 1 Gbps

$\rightarrow$ if 1 THz then 1 Tbps

Networking problem solved! (Not quite.)

Before discussing if networking problem is solved

$\rightarrow$ how to improve bps of AM system with tweaks

$\rightarrow$ can we get 2 bps from 1 Hz frequency?

Issues with just increasing frequency:

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

→ higher cost

Two: wireless propagation

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

→ complications due to multi-path propagation

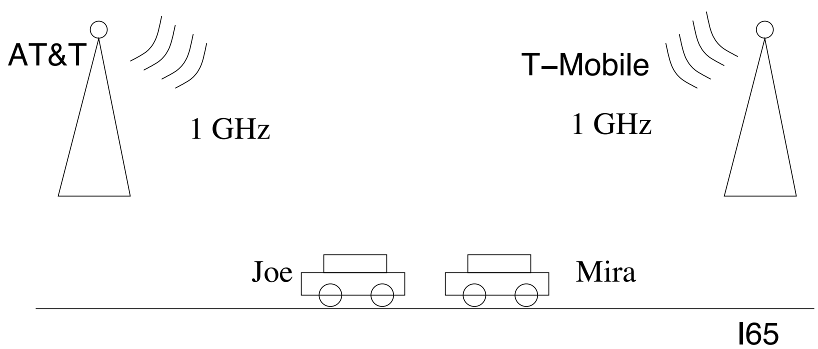
Three: multi-user communication

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

→ key networking problem

Problem of multi-user communication

Example one: interference



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

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

→ what specifically does Joe's smartphone hear?

Joe's device hears the sum of the two signals.

→ property of electromagnetic waves

→ i.e., superposition

Since what Joe's smartphone hears is not what AT&T cell tower sent

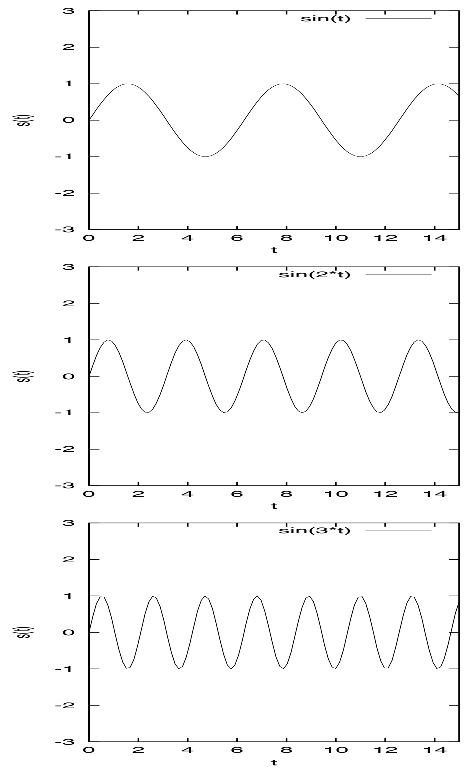
→ distorted signal may cause confusion

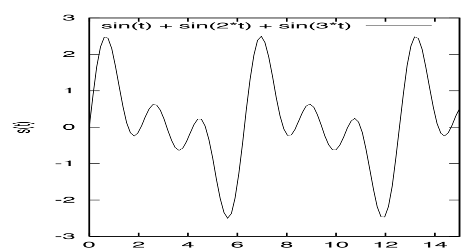
→ called interference

→ figuring out what bits were sent may fail

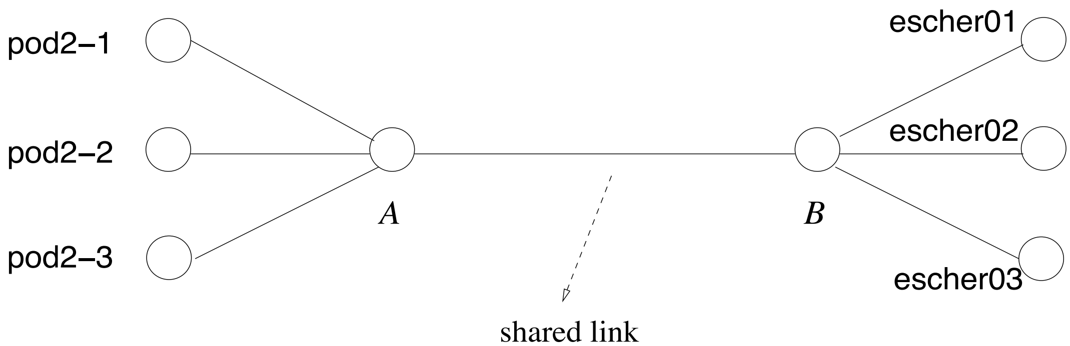
→ not good

Superposition of three sine waves:





Example two: multiplexing



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

→ bits from pod2-1 to escher01 share the point-to-point link with bits from pod2-2 to escher02

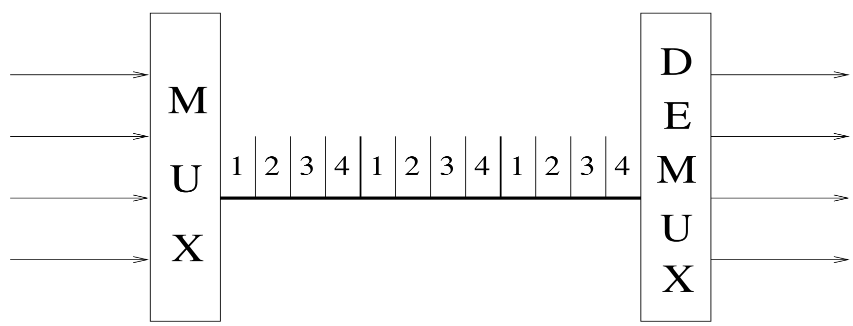
→ A and B are routers/switches that forward multiple traffic streams

Approach based on AM method of sending bits using sine waves:

$\rightarrow$ time-division multiplexing (TDM)

Ex.: four bit streams sharing same link

$\rightarrow$ reserve time slots for each bit stream



$\rightarrow$ user 1 gets slots 1, 5, 9, etc.

$\rightarrow$ user 2 gets slots 2, 6, 10, etc.

$\rightarrow$ router A: acts as multiplexer (MUX) or combiner

$\rightarrow$ router B: acts as demultiplexer (DEMUX) or splitter

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

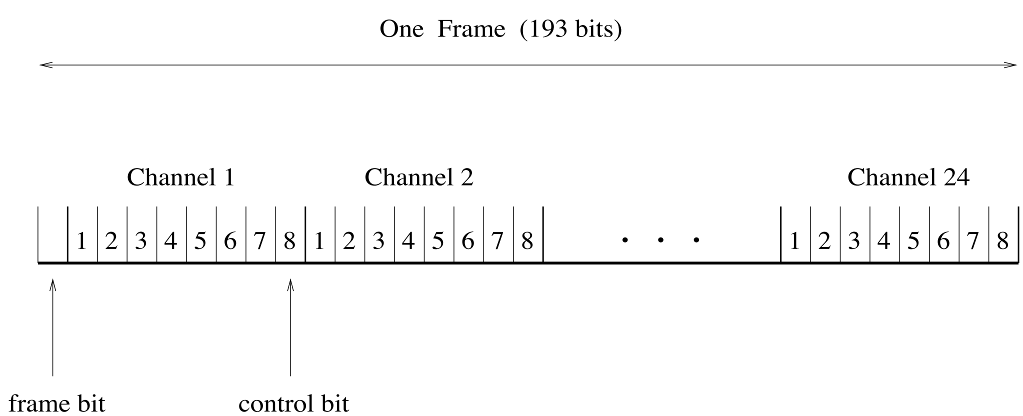
→ e.g., both AT&T and T-Mobile use TDMA

→ 4G/LTE and 5G: OFDMA

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
- bandwidth (bps): $8000 \times 193 = 1.544$ Mbps

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

→ T1 line is still in use today ...

→ today: a single subscriber can get 1000 Mbps nomimal download speed

→ "true" 4G (aka 5G) cellular: 1 Gbps download speed

TDMA is an important multi-user link transmission technology

→ works well if frequency is managed by central authority: e.g., single provider

→ complications if frequency is shared by multiple providers: interference

What we want: multiple information lanes where multiple bit streams can be transmitted simultaneously

→ what modern high-speed networks do

→ 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
  - → coding methods to send parallel bit streams
  - → conceptual basis for analog methods
- move on to FDMA
  - → use analog signals (sinusoid) to send parallel bit streams
- OFDM based multiple access
  - → extend FDMA to squeeze in many parallel bit streams

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

→ not use 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: positive value means 1, negative value means 0

Consider assigning Alice, Bob, Mira code vectors:

→ Alice: (1,-2,1)

→ Bob: (3,5,7)

→ Mira: (19,4,-11)

The code vectors are stored on their smart phones.

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

→ ignore how the cell tower transmits (17, -3, -17) using 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 3D vectors $x = (x_1, x_2, x_3)$ and $y = (y_1, y_2, y_3)$, their dot (or inner) product is

$$x \circ y = x_1 y_1 + x_2 y_2 + x_3 y_3$$

Hence, for Alice:

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

$\rightarrow$ positive means bit 1

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

$\rightarrow$ negative means bit 0

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 orthogonal: $x \circ y = 0$

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

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

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

The cell tower wants to send 1 to Alice, 0 to Bob, 1 to Mira.

The cell tower computed (17, -3, 17) to broadcast via:

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

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 can we do?

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

Thus in CDMA (code division multiple access) using algebra, we hide the bits 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.

→ e.g., Verizon, Sprint use CDMA

→ if code vectors are chosen to be random, then additional feature of security

In general: to communicate n bits belonging to n users

- Assign n orthogonal code vectors in n-dimensional vector space

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

- 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

- To decode user i'th bit a_i, receiver computes dot product

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

  $\rightarrow$ by orthogonality