Computer Networks

Lecture 4: Physical layer + Data Link layer

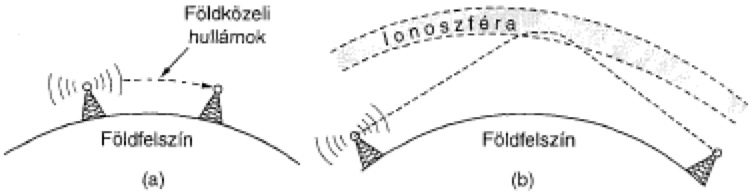
Based on slides from D. Choffnes Northeastern U. and P. Gill from StonyBrook University
Revised Autumn 2015 by S. Laki

Fundamentals – wireless transmission

- **Frequency:** the rate per second of a vibration constituting an electromagnetic wave.
  - Notation: f
  - Measured in: Hertz (Hz)

- **Wavelength:** the distance between successive crests of a wave
  - Notation: λ

- **Speed of light:** signal propagation speed of electric signals in a physical media
  - Notation c
  - In vacuum: kb. $3 * 10^8 \frac{m}{s}$
  - In copper or optical cable: 2/3 x c(vacuum)
- **Relationship: $\lambda f = c$**

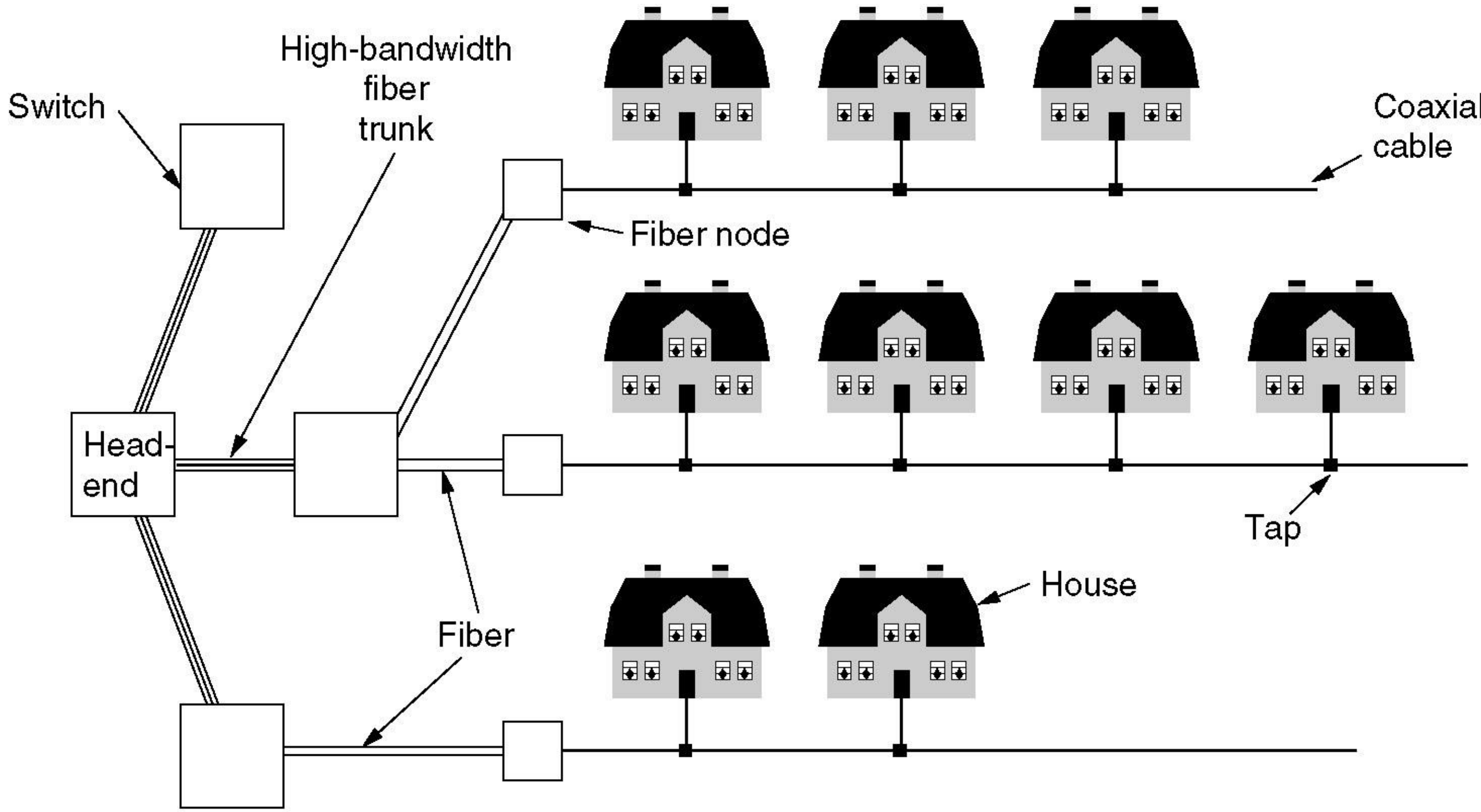


Wireless

- **Radio frequency transmission** – simple; large distances; indoor and outdoor; frequency-dependent propagation properties

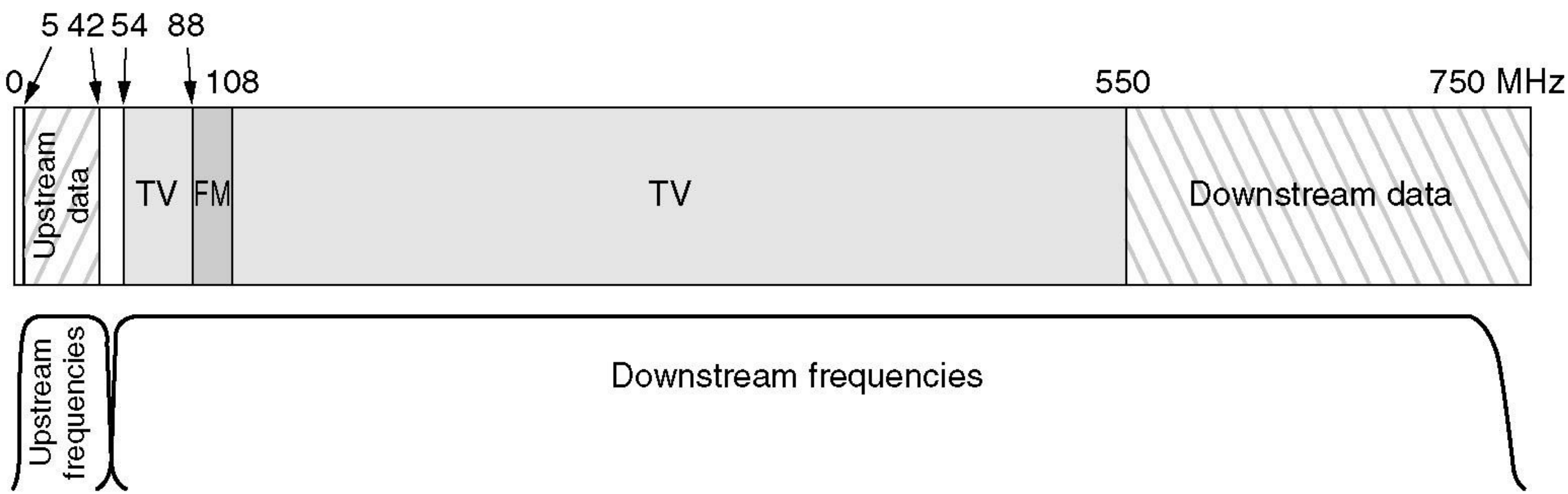
- **Microwave transmission** – propagation along a straight line; attenuation; cheap
- **Infrared and millimeter-wave** – small distances; cannot go through objects
- **Visible light** – laser; high speed, cheap; weather conditions;

Internet in a cable TV network



(a)

Internet in a cable TV network

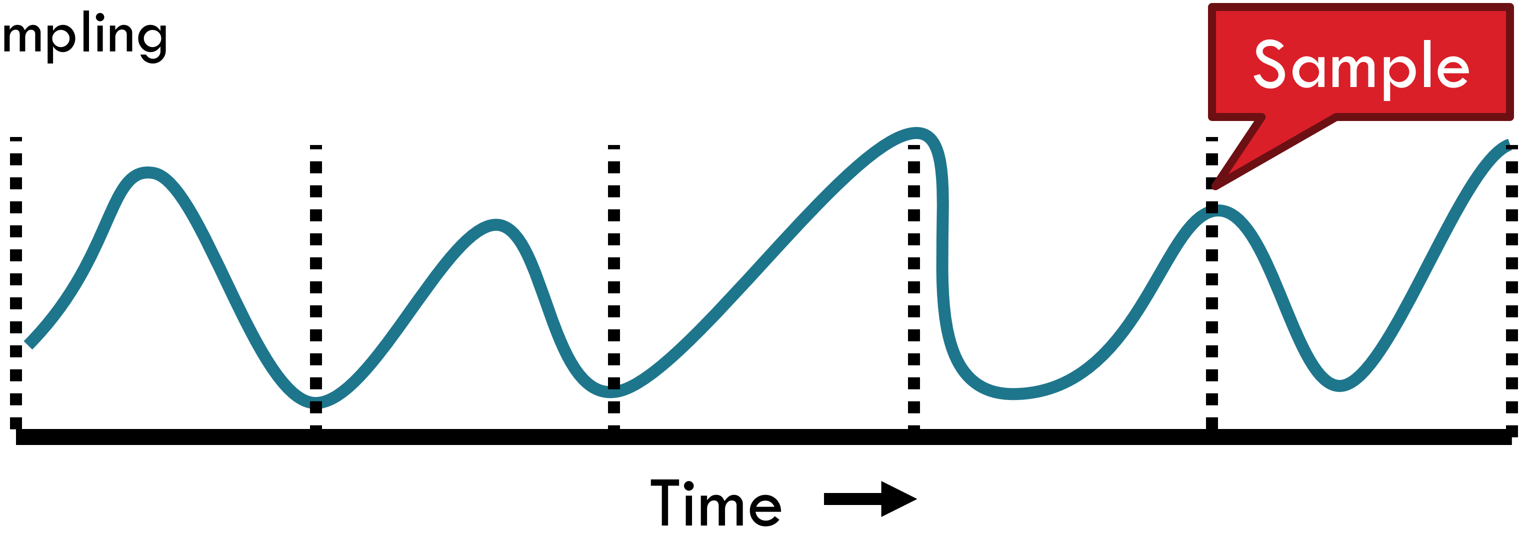


Already discussed…

Data transmission

Assumptions

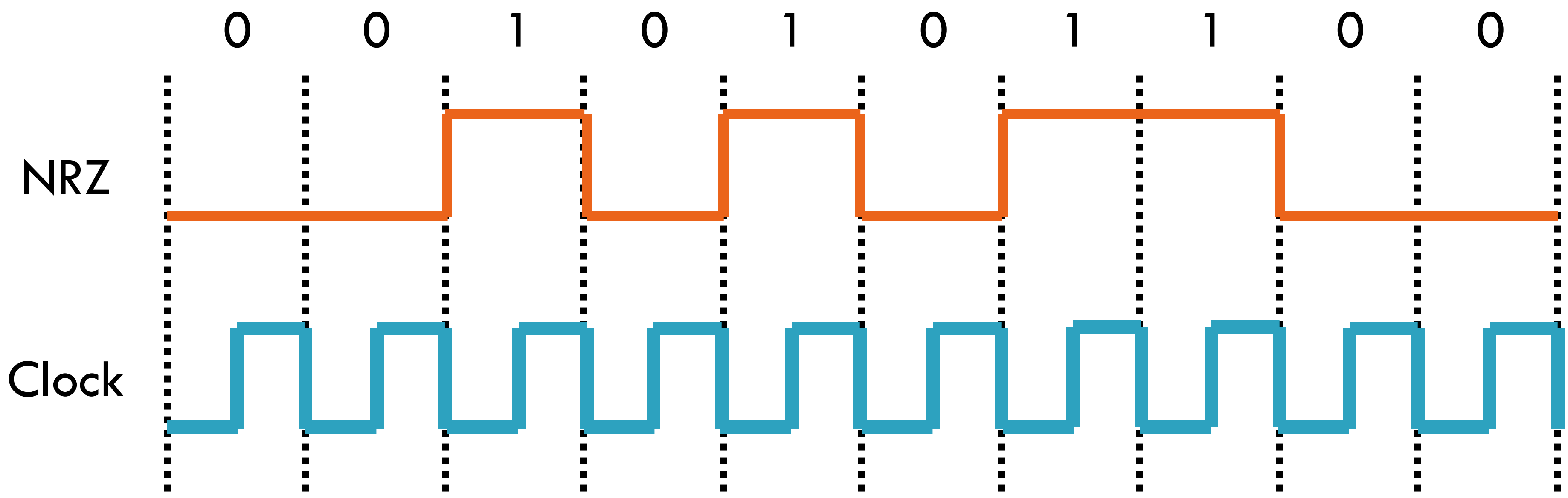
- We have two discrete signals, high and low, to encode 1 and 0
- Transmission is synchronous, i.e. there is a clock that controls signal sampling



- Amplitude and duration of signal must be significant

Non-Return to Zero (NRZ)

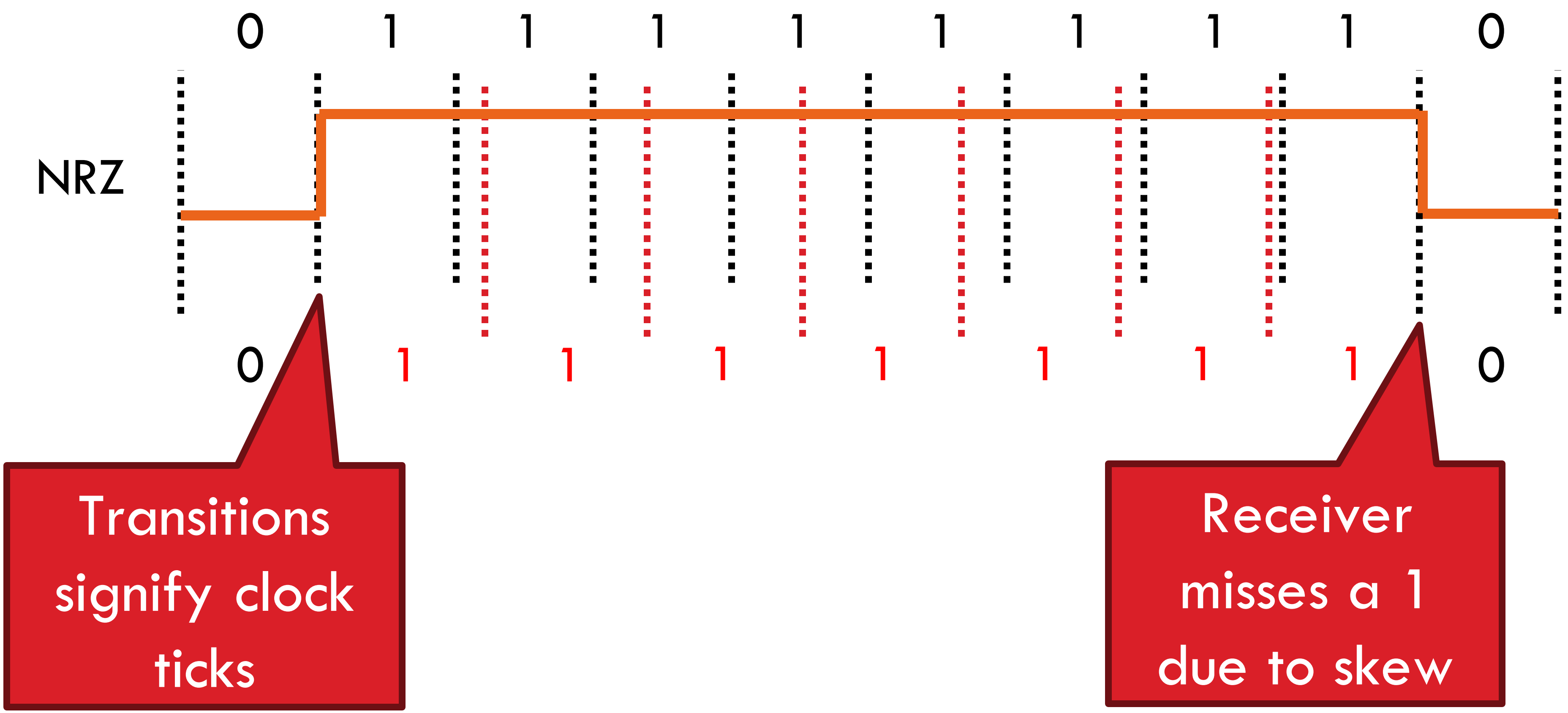
- 1 → high signal, 0 → low signal



- Problem: long strings of 0 or 1 cause desynchronization
  - How to distinguish lots of 0s from no signal?
  - How to recover the clock during lots of 1s?

Desynchronization

- Problem: how to recover the clock during sequences of 0's or 1's?



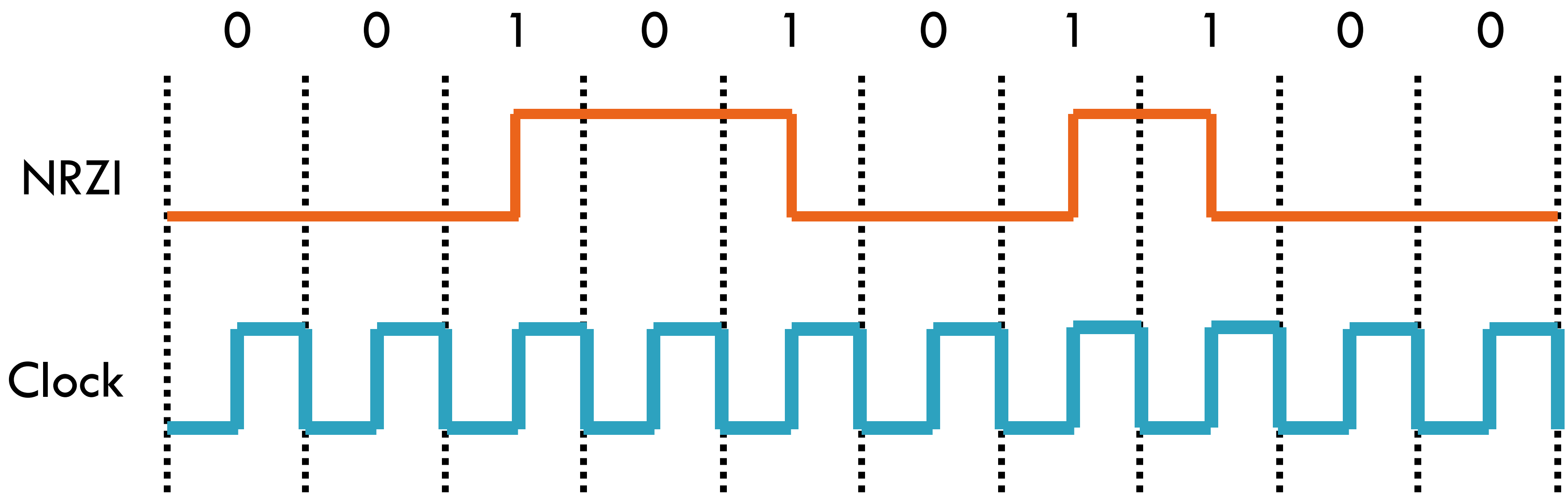
- Clock drift is major problem – two different clocks never stay in perfect synchrony

Options to tell the receiver when to sample

- Relying on permanently synchronized clocks does not work
  1. Explicit clock signal
     - Needs parallel transmission over some additional channel
     - Must be in synch with the actual data, otherwise pointless !
     - Useful only for short-range communication
  2. Synchronize the receiver at crucial points (e.g., start of a character or of a block)
     - Otherwise, let the receiver clock run freely
     - Relies on short-term stability of clock generators (do not diverge too quickly)
  3. Extract clock information from the received signal itself
     - *Self-clocked signals*
     - Put enough information into the data signal itself so that the receiver can know immediately when a bit starts/stop

Non-Return to Zero Inverted (NRZI)

- 1 → make transition, 0 → remain the same



- Solves the problem for sequences of 1s, but not 0s

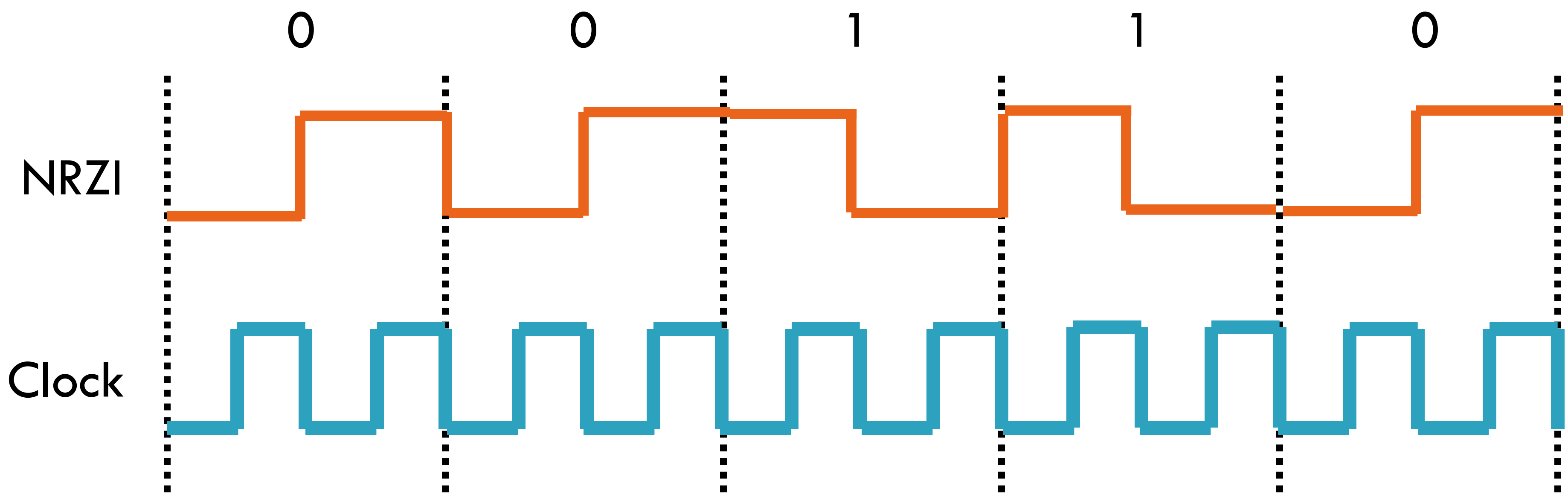
Ethernet examples:

10BASE-TX

100BASE-TX

Manchester – used by 10BASE-TX

- 1 → high-to-low, 0 → low-to-high



- Good: Solves clock skew (every bit is a transition)
- Bad: Halves throughput (two clock cycles per bit)

4-bit/5-bit (100 Mbps Ethernet)

- Observati: NRZI works as long as no sequences of 0
- Idea: e ll 4-bit sequences as 5-bit sequences with no

8-bit / 10-bit used in Gigabit Ethernet

4-bit	5-bit	4-bit	5-bit
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

- Tradeoff: efficiency drops to 80%

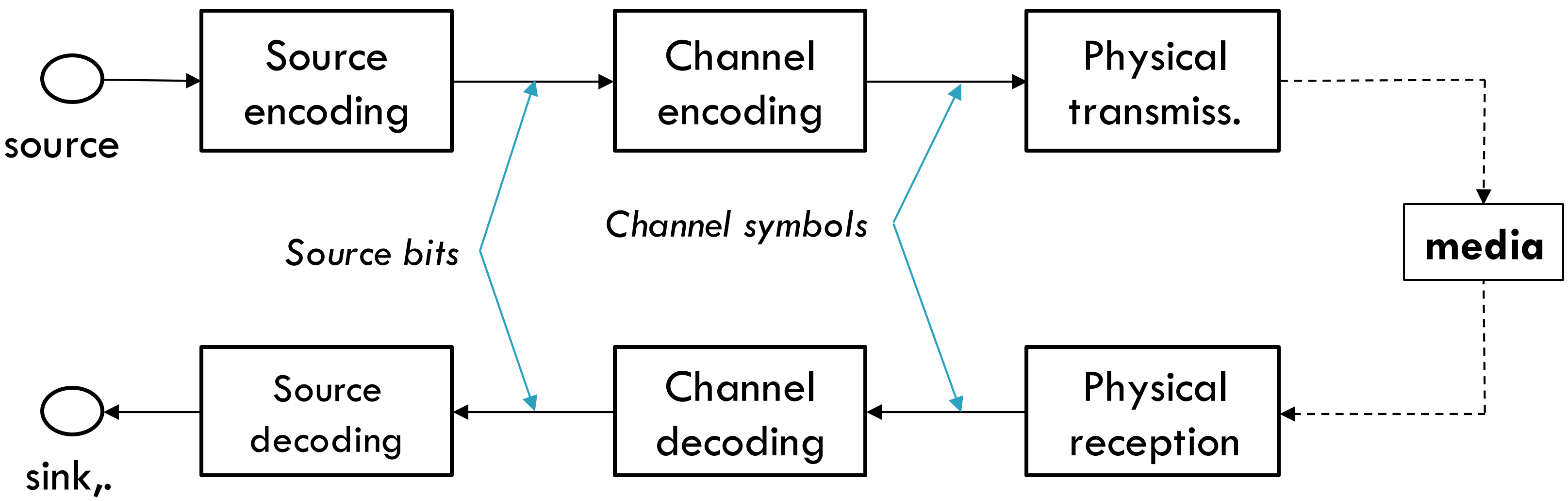
Signal transmission

Baseband VS broadband transmission

- *baseband*
  - Baseband transmission directly puts the digital symbol sequences onto the wire
  - At different levels of current, voltage, … essentially, direct current (DC) is used for signaling
  - Baseband transmission suffers from the problems discussed above
    - Limited bandwidth reshapes the signal at receiver
    - Attenuation and distortion depend on frequency and baseband transmissions have many different frequencies because of their wide Fourier spectrum

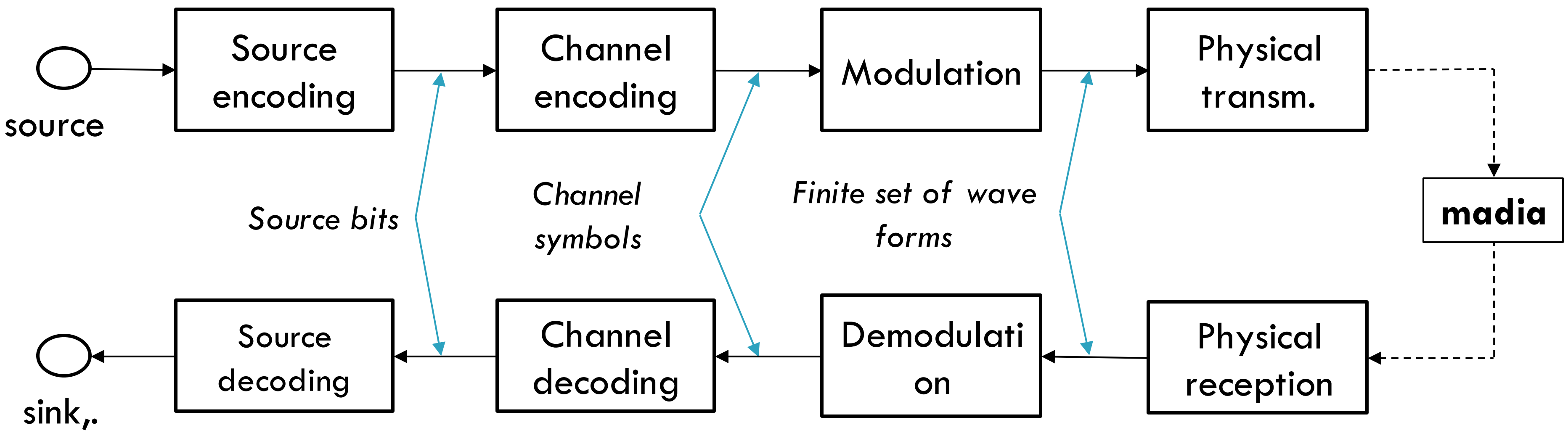
- *broadband*
  - Idea: get rid of the wide spectrum needed for DC transmission
  - Use a sine wave as a carrier for the symbols to be transmitted
    - Typically, the sine wave has high frequency
    - But only a single frequency!
    - Pure sine waves has no information, so its shape has to be influenced according to the symbols to be transmitted
  - The carrier has to be modulated by the symbols (widening the spectrum)
    - Three parameters that can be influenced Amplitude, Frequency, Phase

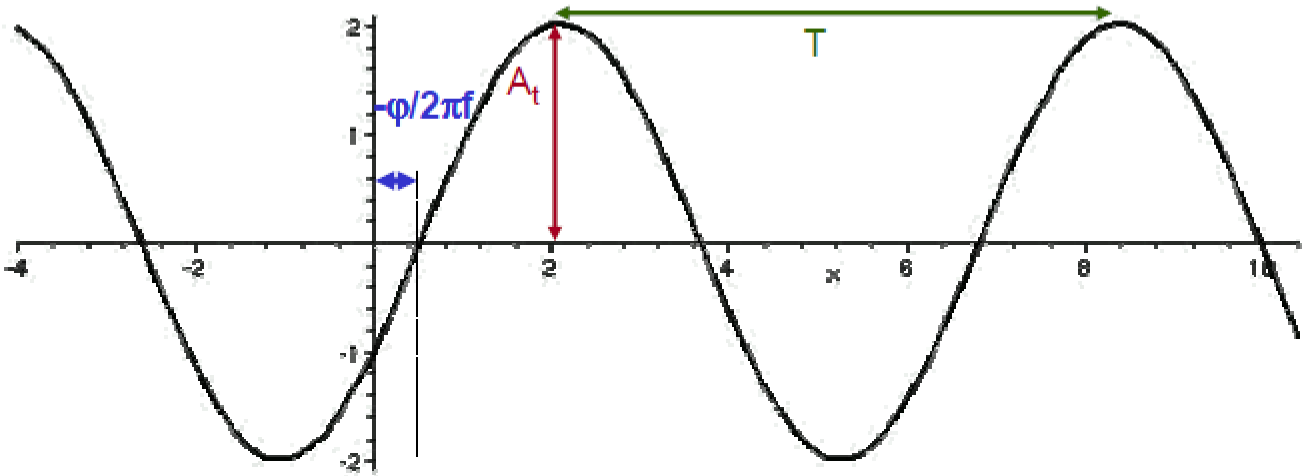
Digital baseband transmission



- Bring source information in digital form
  - E.g., sample and quantize an analog voice signal, represent text as ASCII
- Source encode: Remove redundant or irrelevant data
  - E.g., lossy compression (MP3, MPEG 4); lossless compression (Huffmann coding, runlength coding)
- Channel encode: Map source bits to channel symbols
  - Potentially several bits per symbol
  - May add redundancy bits to protect against errors
  - Tailored to channel characteristics
- Physical transmit: Turn the channel symbols into physical signals
- At receiver: Reverse all these steps

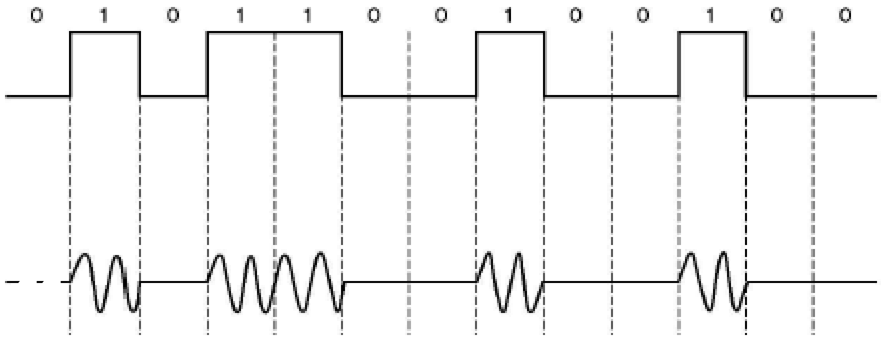
Digital broadband transmission





ry

uency and φ the

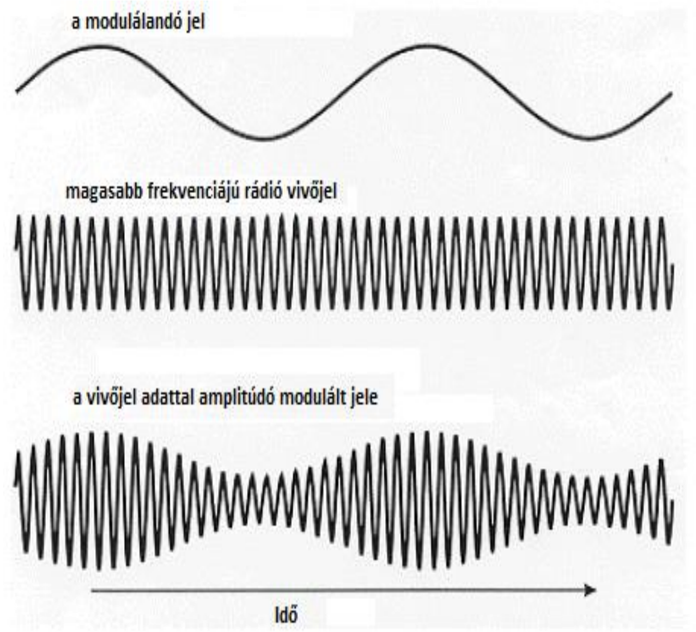


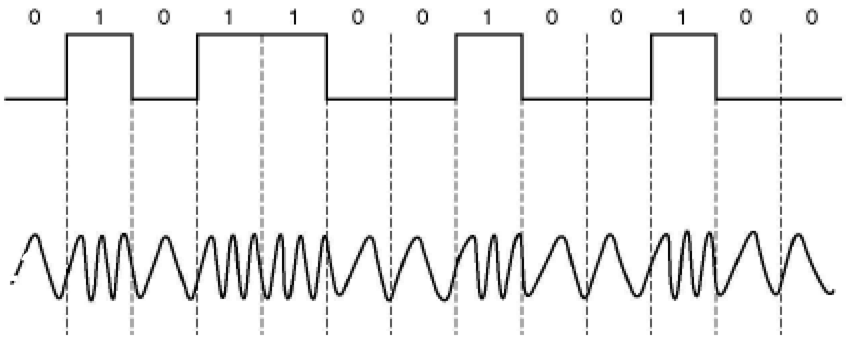
tion

- The time-varying *s(t)* signal is encoded into the amlitude of the sine wave (carrier):

  $$f_A(t) = s(t) * \sin(2\pi f t + \varphi)$$

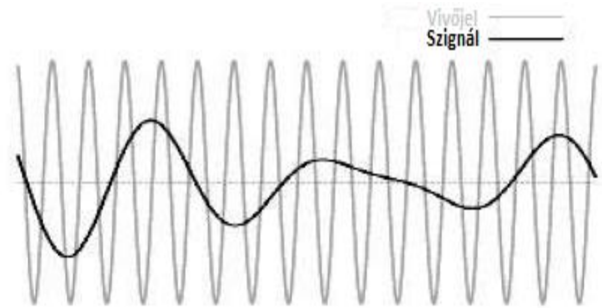
  - *Analog signal*: amplitude modulation
  - *Digital signal*: amplitude keying or on/off keying
    (s(t) takes discrete values)





ation

- The time-varying s(t) signal is encoded into the frequency of the sine wave:
  $$f_F(t) = a * \sin(2\pi s(t)t + \varphi)$$
  - *analog signal*: frequency modulation
  - *Digital signal*: frequency-shift keying



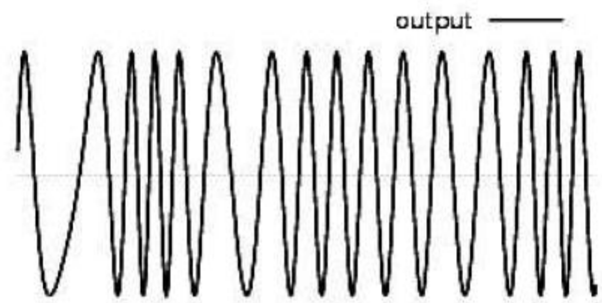
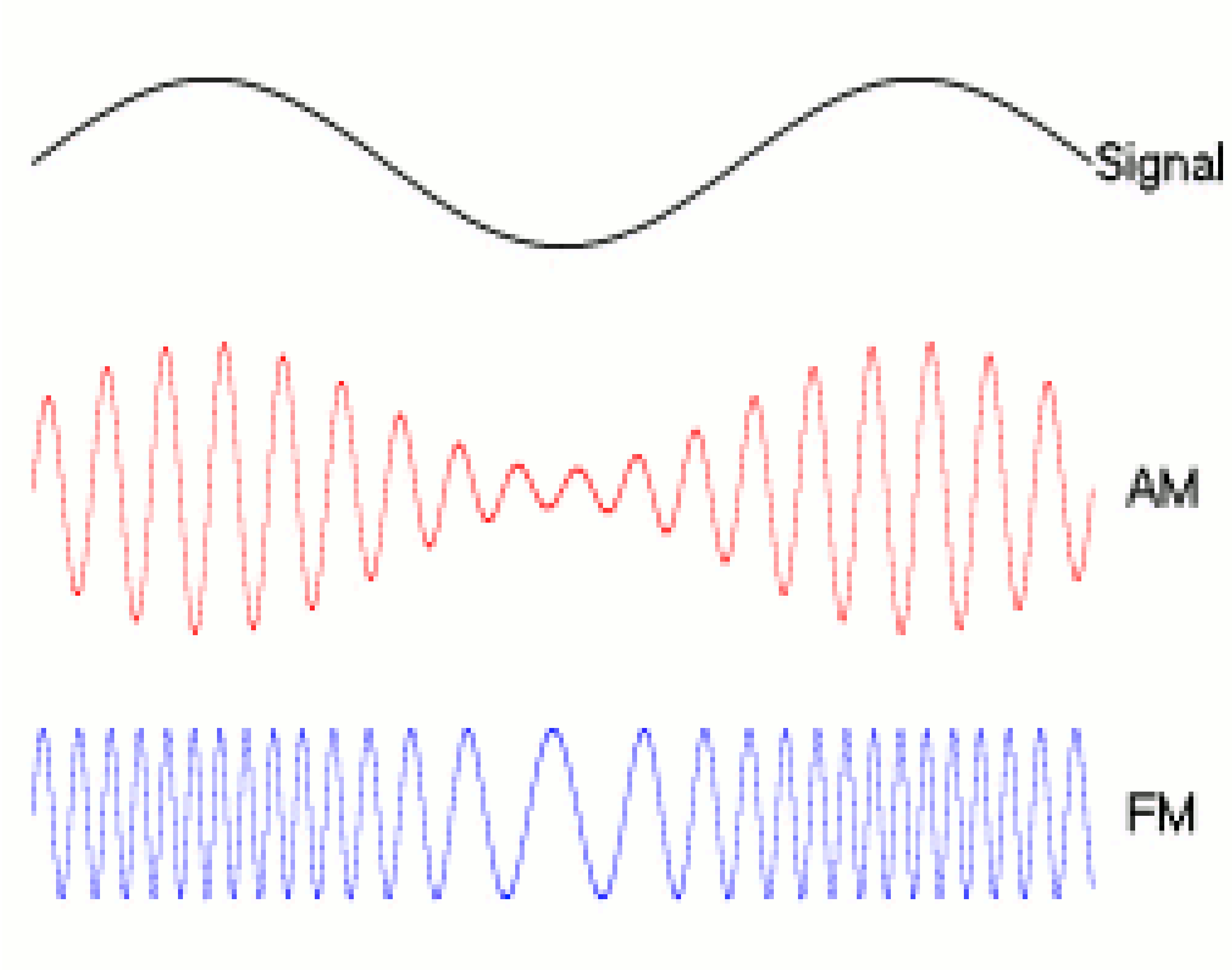
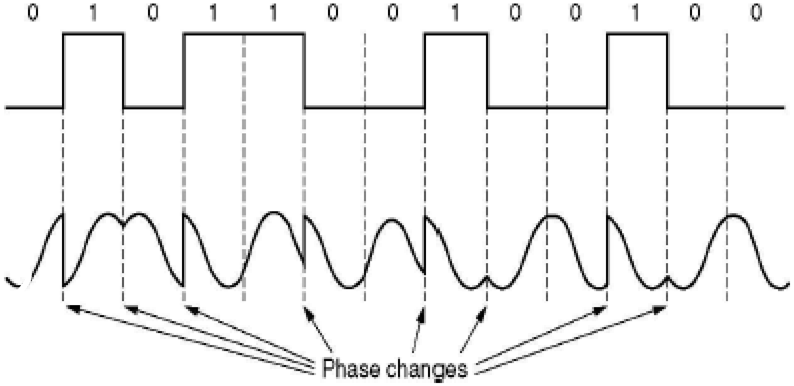


Illustration - AM & FM for analog signals





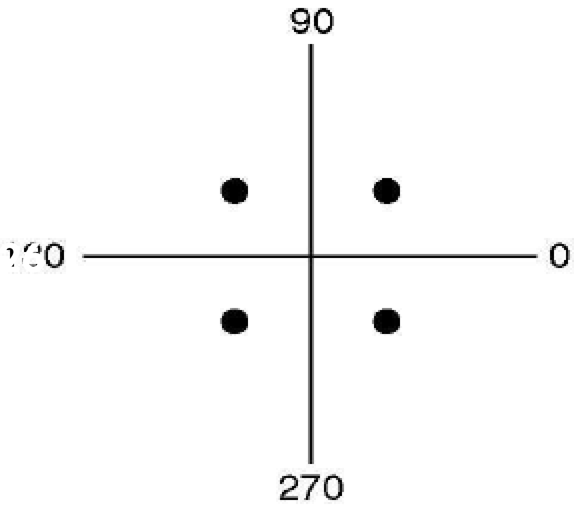
he phase of

$s(t))$

ulation (not

really used)

- *Digital signal*: phase-shift keying (discrete set of phase changes)



ltiple symbols

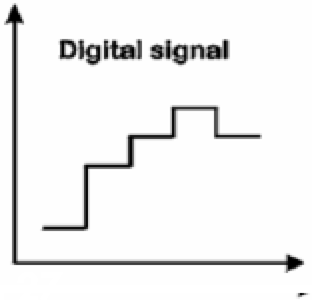
lues

ally quite well distinguish phase shifts

- 4 symbols/ values: $\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}$
- Result: Data rate is twice the symbol rate
- Technique is called Quadrature Phase Shift Keying (QPSK)

Amlitude + Phase modulation

- Methods can be combined
- Symbols are encoded by a discrete set of amlitude, phase values
  - E.g. 16 symbols
  - Four times higher data rate than the symbol rate
  - Called as **Q**uadrature **A**mplitude **M**odulation-16



VS analog signals

- A sender has two principal options what types of signals to generate
  - It can choose from a finite set of different signals – digital transmission
  - There is an infinite set of possible signals – analog transmission
- Simplest example: Signal corresponds to current/voltage level on the wire
  - In the digital case, there are finitely many voltage levels to choose from
  - In the analog case, any voltage is legal
- More complicated example: finite/infinitely many sinus functions
  - In both cases, the resulting wave forms in the medium can well be continuous functions of time!
- Advantage of digital signals: There is a principal chance that the receiver can precisely reconstruct the transmitted signal

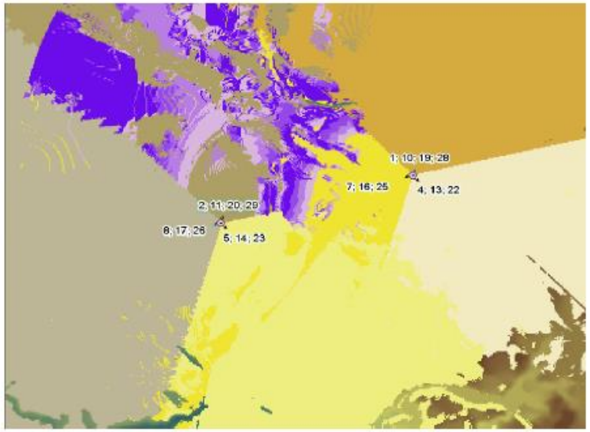
Static Channel Allocation

Multiplexing

- Enabling multiple signals to travel through the same media at the same time
- To this end, the channel is split into multiple smaller subchannels
- A special device (multiplexer) is needed at the sender, transmitting signals to the proper subchannel

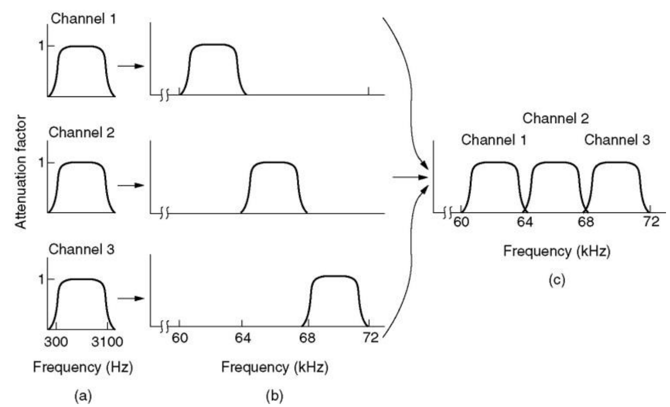
Space-Division Multiplexing

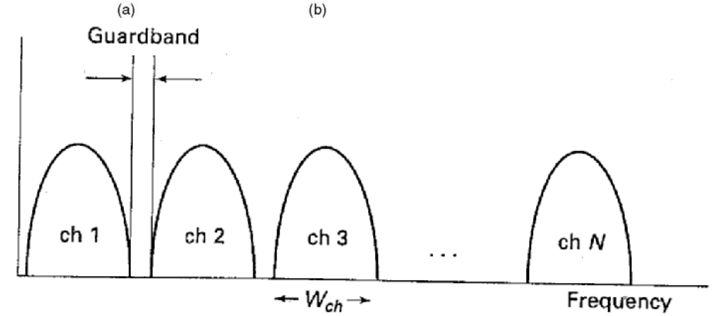
- Simplest way of multiplexing
- Wired example: point-to-point wire for each subchannel
- Wireless example: Different antennas for the subchannels



Frequency-Division Multiplexing

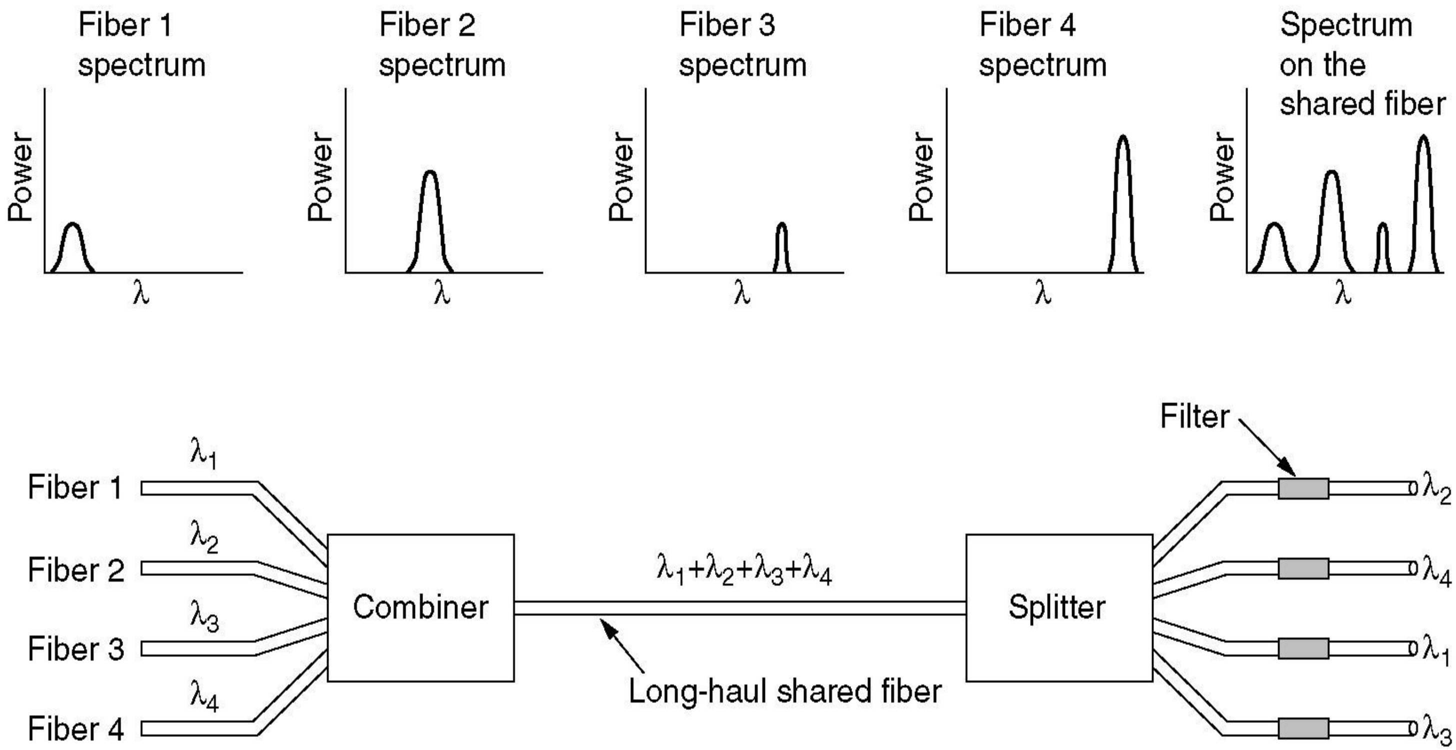
- Multiple signals are combined and transmitted over the channel
- Each signal is transmitted in different frequency ranges
- Typically used for analog transmission
- Multiple implementations…





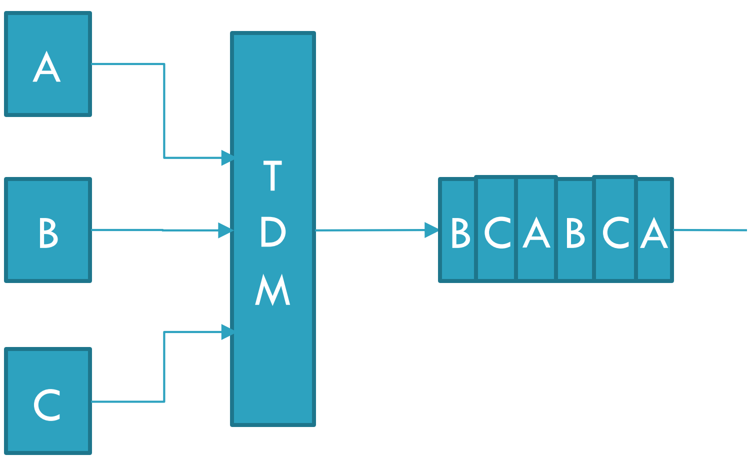
Wavelength-Division Multiplexing

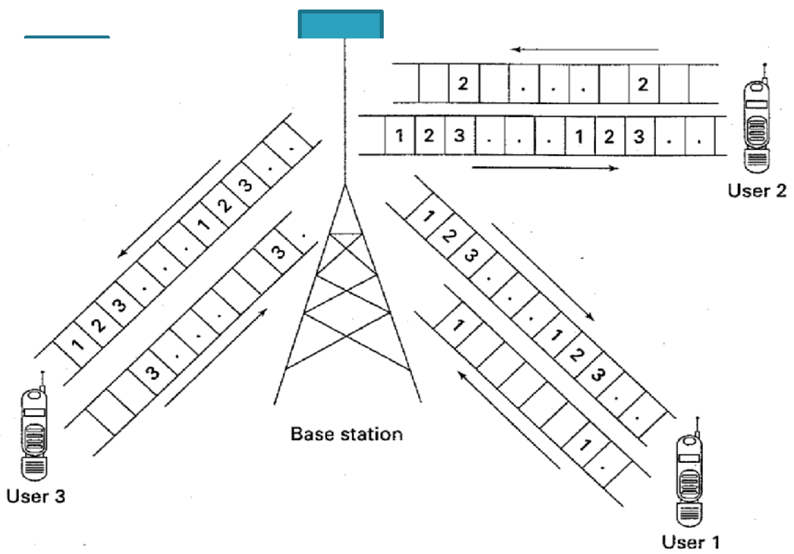
- Used for optical cables
- IR laser rays at different wavelengths



Time-Division Multiplexing

- Time is divided into not overlapping intervals
- Each time slot is assigned to a sender, exlusively.
- Empty slots may happen.



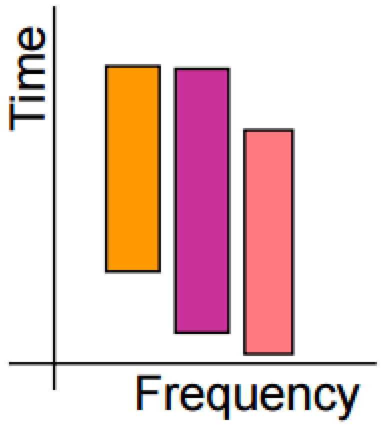


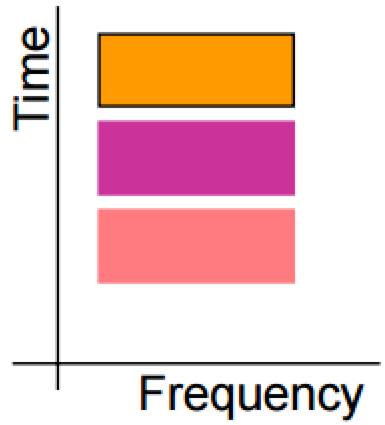
CDMA – Code Division Multiple Access

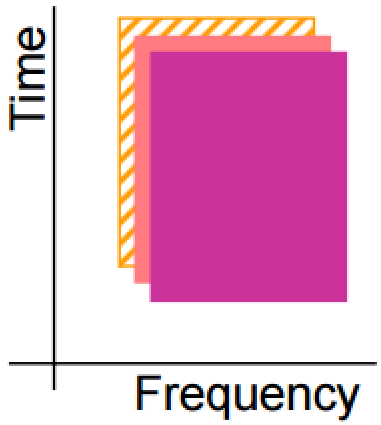
Frequency Division Multiple Access
FDMA

Time Division Multiple Access
TDMA

Code Division Multiple Access
CDMA







CDMA Analogy

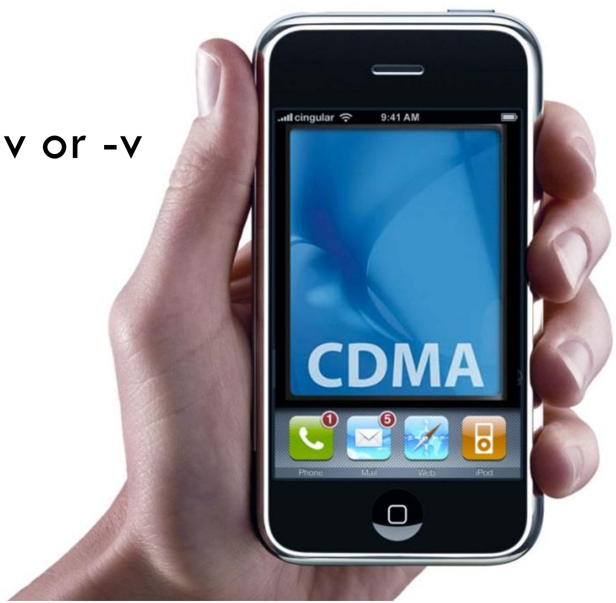
- 10 people in a room.
  - 5 speak English, 2 speak Spanish, 2 speak Chinese, and 1 speaks Russian.
- Everyone is talking at relatively the same time over the same medium – the air.
- Who can listen to whom and why?
- Who can't you understand?
- Who can't speak to anyone else?

CDMA – Code Division Multiple Access

- Used by 3G and 4G cellular networks
- Each station can broadcast at any time in the full frequency spectrum
- The signals may interfere
  - Resulting in a linear combination of individual signals

- Algorithm
  - We assign a vector of length m to each station: v
    - Pairwise orthogonal vectors!!!
  - Each bit is encoded
    by the chip vector of the sender or it's complement: v or -v
  - If it sends bit 1, it transmits v
  - If it sends bit 0, it transmits -v

- Result is a sequence of vectors of length m



CDMA – Code Division Multiple Access

- Interference
  - A sends a,-a,a,a
  - B sends b,b,-b,-b
  - After interference we receive: a+b,-a+b,a-b,a-b ???

- How to decode?

CDMA – Code Division Multiple Access

- Interference
  - A sends a,-a,a,a
  - B sends b,b,-b,-b
  - After interference we receive: a+b,-a+b,a-b,a-b ???

- Decoding the message of A
  - Take the dot product by the sender's chip code
    - $(a+b)a > 0 \Rightarrow 1$
    - $(-a+b)a < 0 \Rightarrow 0$
    - $(a-b)a > 0 \Rightarrow 1$
    - $(a-b)a > 0 \Rightarrow 1$

If the dot product is

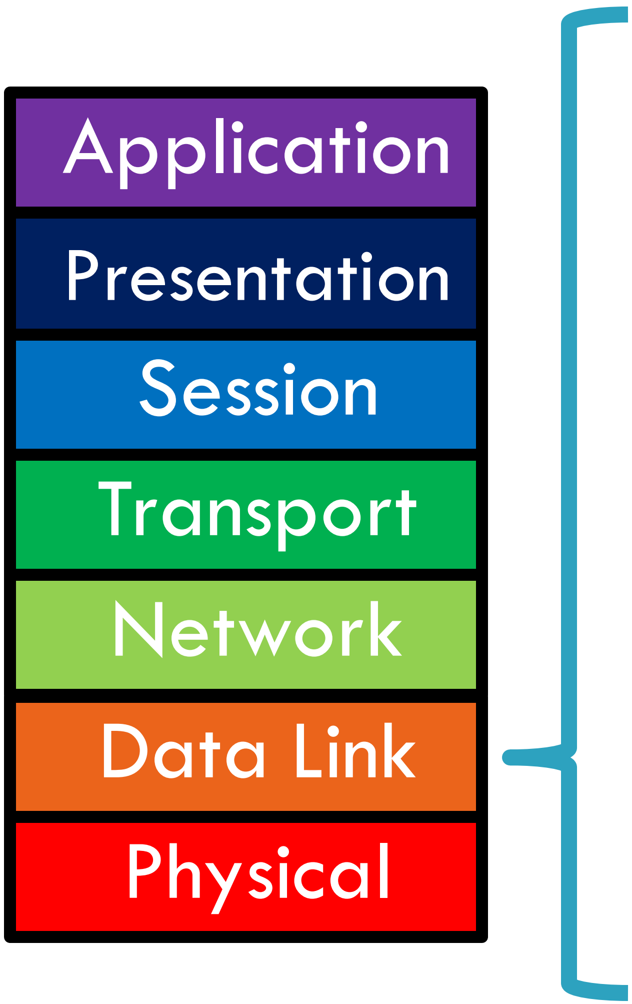
<0: bit 0 was sent by A

>0: bit 1 was sent by A

$=0$: nothing was sent by A

the channel is not used by A

Data Link Layer



- Function:
  - Send blocks of data (frames) between physical devices
  - Regulate access to the physical media
- Key challenge:
  - How to delineate frames?
  - How to detect errors?
  - How to perform media access control (MAC)?
  - How to recover from and avoid collisions?

Outline

- Framing
- Error Checking and Reliability
- Media Access Control
  - 802.3 Ethernet
  - 802.11 Wifi

Framing

- Physical layer determines how bits are encoded
- Next step, how to encode blocks of data
  - Packet switched networks
  - Each packet includes routing information
  - Data boundaries must be known so headers can be read
- Types of framing
  - Byte oriented protocols
  - Bit oriented protocols
  - Clock based protocols

Byte Oriented: Byte Stuffing

- Add **FLAG** bytes as sentinel to the beginning and end of the data
- Problem: what if **FLAG** appears in the data?
  - Add a special **DLE** (Data Link Escape) character before **FLAG**
  - What if **DLE** appears in the data? Add **DLE** before it.
  - Similar to escape sequences in C
    - printf("You must \"escape\" quotes in strings");
    - printf("You must \\escape\\ forward slashes as well");
- Used by Point-to-Point protocol, e.g. modem, DSL, cellular

Byte Oriented: Byte Counting

- Sender: insert length of the data in bytes at the beginning of each frame
- Receiver: extract the length and read that many bytes
- What happens if there is an error transmitting the count field?

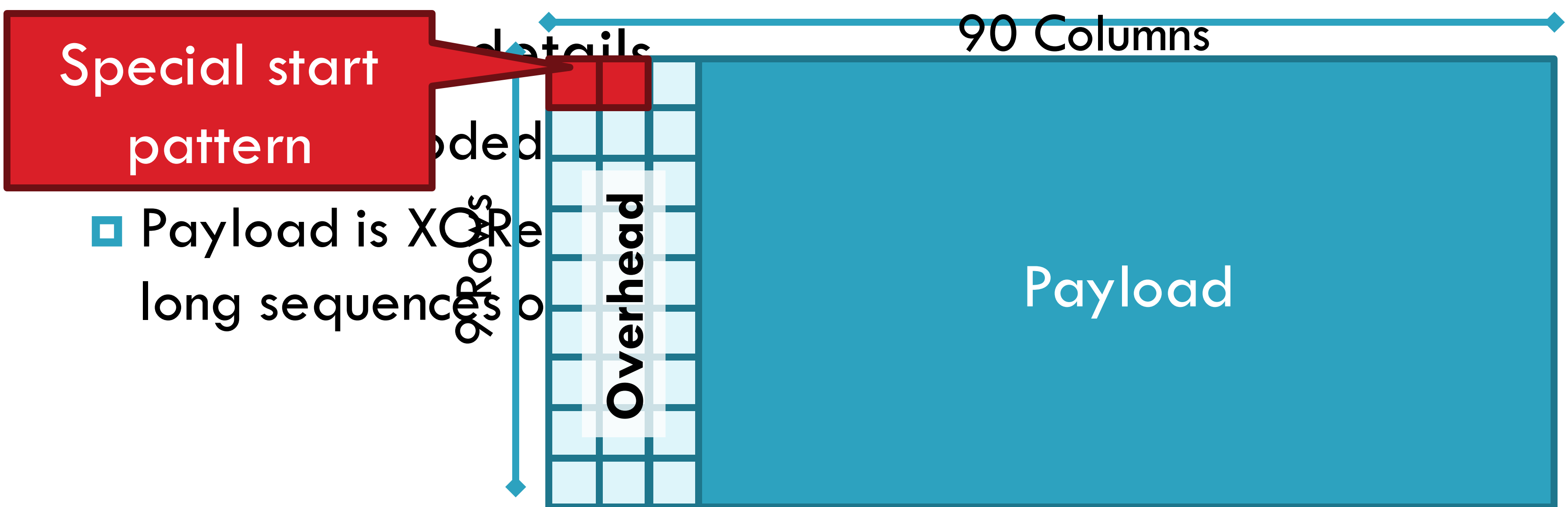
Bit Oriented: Bit Stuffing

01111110	Data	01111110

- Add sentinels to the start and end of data (similarly to byte stuffing)
  - Both sentinels are the same
  - Example: 01111110 in High-level Data Link Protocol (HDLC)
- Sender: insert a 0 after each 11111 in data
  - Known as "bit stuffing"
- Receiver: after seeing 11111 in the data…
  - 11111**0** → remove the 0 (it was stuffed)
  - 11111**1** → look at one more bit
    - 11111**10** → end of frame
    - 11111**11** → error! Discard the frame
- Disadvantage: 20% overhead at worst
- What happens if error in sentinel transmission?

Clock-based Framing: SONET

- **S**ynchronous **O**ptical **Net**work
  - Transmission over very fast optical links
  - STS-*n*, e.g. STS-1: 51.84 Mbps, STS-768: 36.7 Gbps
- STS-1 frames based on fixed sized frames
  - 9*90 = 810 bytes → after 810 bytes look for start pattern
  - details
  - oded
  - Payload is XORe
    long sequences o



Outline

- Framing
- Error Checking
- Media Access Control
  - 802.3 Ethernet
  - 802.11 Wifi

Dealing with Noise

- The physical world is inherently noisy
  - Interference from electrical cables
  - Cross-talk from radio transmissions, microwave ovens
  - Solar storms
- How to detect bit-errors in transmissions?
- How to recover from errors?

Naïve Error Detection

- Idea: send two copies of each frame
  - if (memcmp(frame1, frame2) != 0) { OH NOES, AN ERROR! }
- Why is this a bad idea?
  - Extremely high overhead
  - Poor protection against errors
    - Twice the data means twice the chance for bit errors

Parity Bits

- Idea: add extra bits to keep the number of 1s even
  - Example: 7-bit ASCII characters + 1 parity bit

0101001 1 1101001 0 1011110 1 0001110 1 0110100 1

10

- Detects 1-bit errors and some 2-bit errors
- Not reliable against bursty errors

Error control

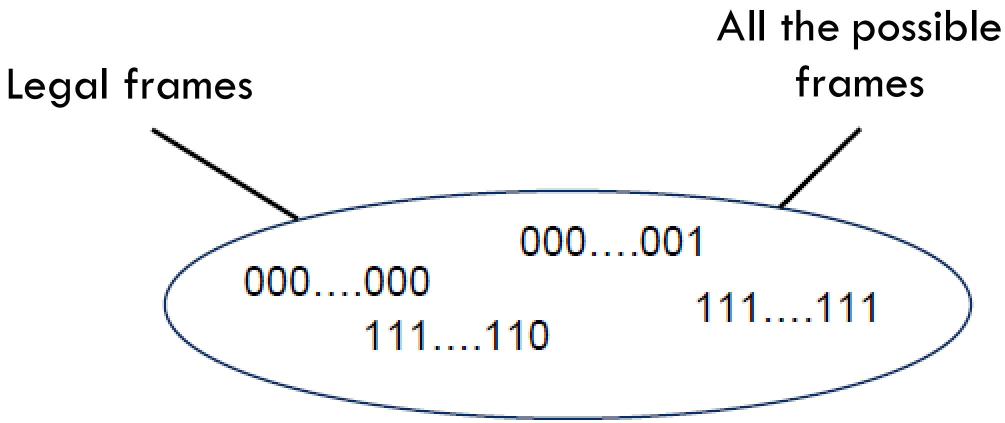
- Error Control Strategies
  - Error Correcting codes (Forward Error Correction (FEC))
  - Error detection and retransmission Automatic Repeat Request (ARQ)

Error control

- Objectives
  - Error detection
    - with correction
      - Forward error correction
    - without correction -> e.g. drop a frame
      - Backward error correction
      - The erroneous frame needs to be retransmitted
  - Error correction
    - without error detection
      - e.g. in voice transmission

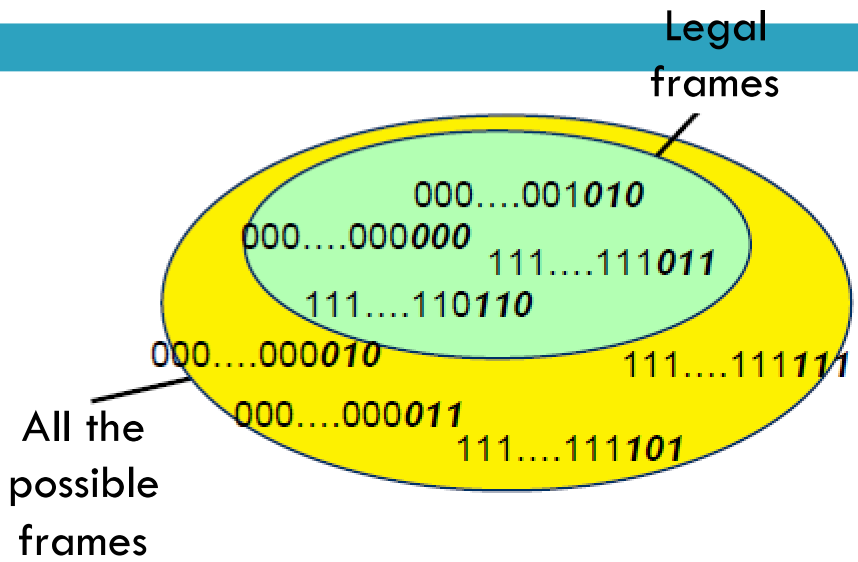
Redundancy

- Redundancy is required for error control
- Without redundancy
  - 2^m possible data messages can be represented as data on m bits
  - They all are legal!!!
  - Each error results a new legal data message
- How to detect errors???



Error-correcting codes
Redundancy

- A frame consists of
  - m data bits (message)
  - r redundant/check bits
  - The total length n = m + r

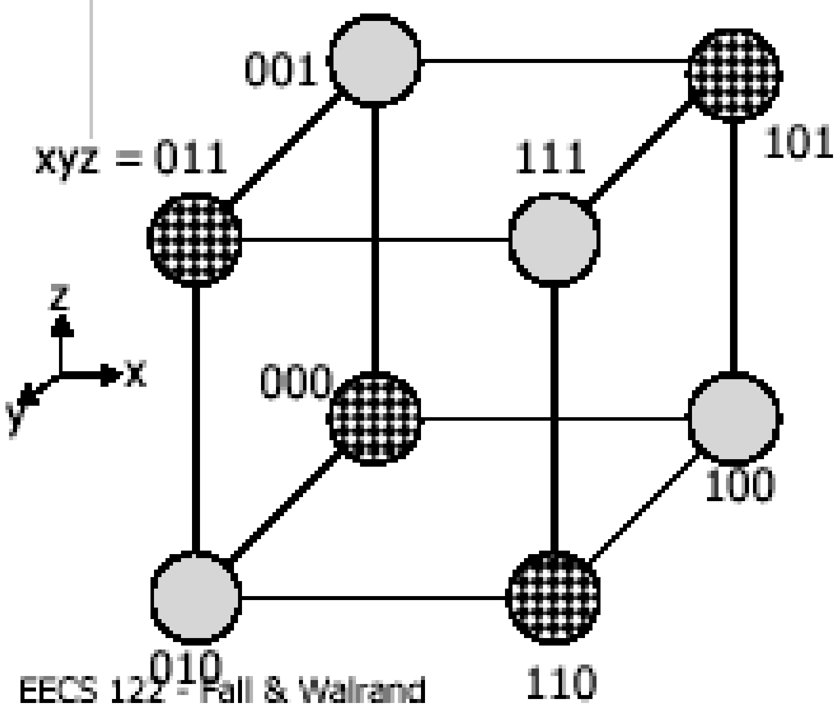


- This n-bit unit is referred to as an n-bit codeword!

Error Control Codes

How Codes Work: Words and Codewords

- Code = subset of possible words: Codewords
- Example:
  - 3 bits => 8 words; codewords: subset



Words:
000, 001, 010, 011
100, 101, 110, 111

Code:
000, 011, 101, 110

Send only codewords

Hamming distance

- The Hamming distance between two codewords is the number of differences between corresponding bits.

1. The Hamming distance d(000, 011) is 2 because

$\boxed{000 \oplus 011 \text{ is } 011 \text{ (two 1s)}}$

2. The Hamming distance d(10101, 11110) is 3 because

$\boxed{10101 \oplus 11110 \text{ is } 01011 \text{ (three 1s)}}$

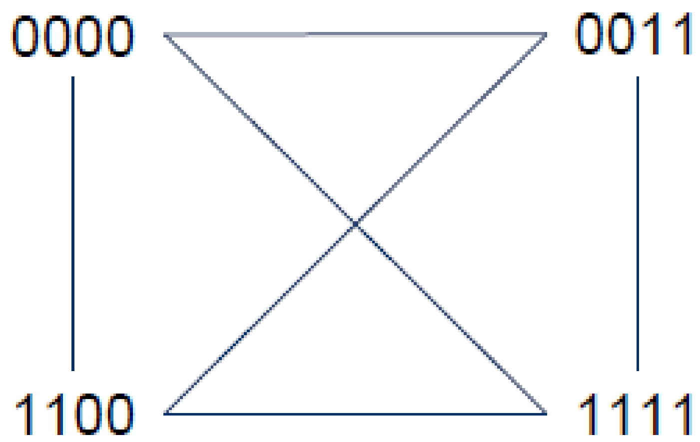
Hamming distance

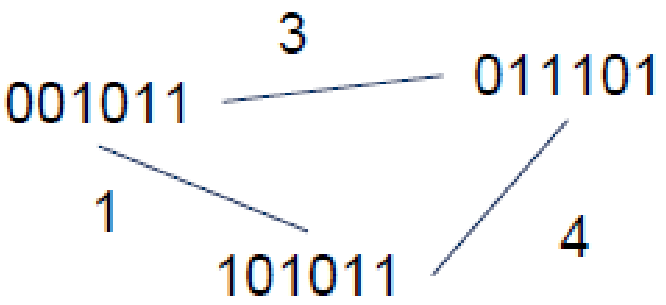
- If not all the 2^n possible codewords are used
  - Set of legal codewords =: S
- Hamming distance of the complete code
  - The smallest Hamming distance of between all the possible pairs in the set of legal codewords (S)

$$d(S) = \min_{x,y \in S, x \neq y} d(x, y)$$

What is the Hamming distance?

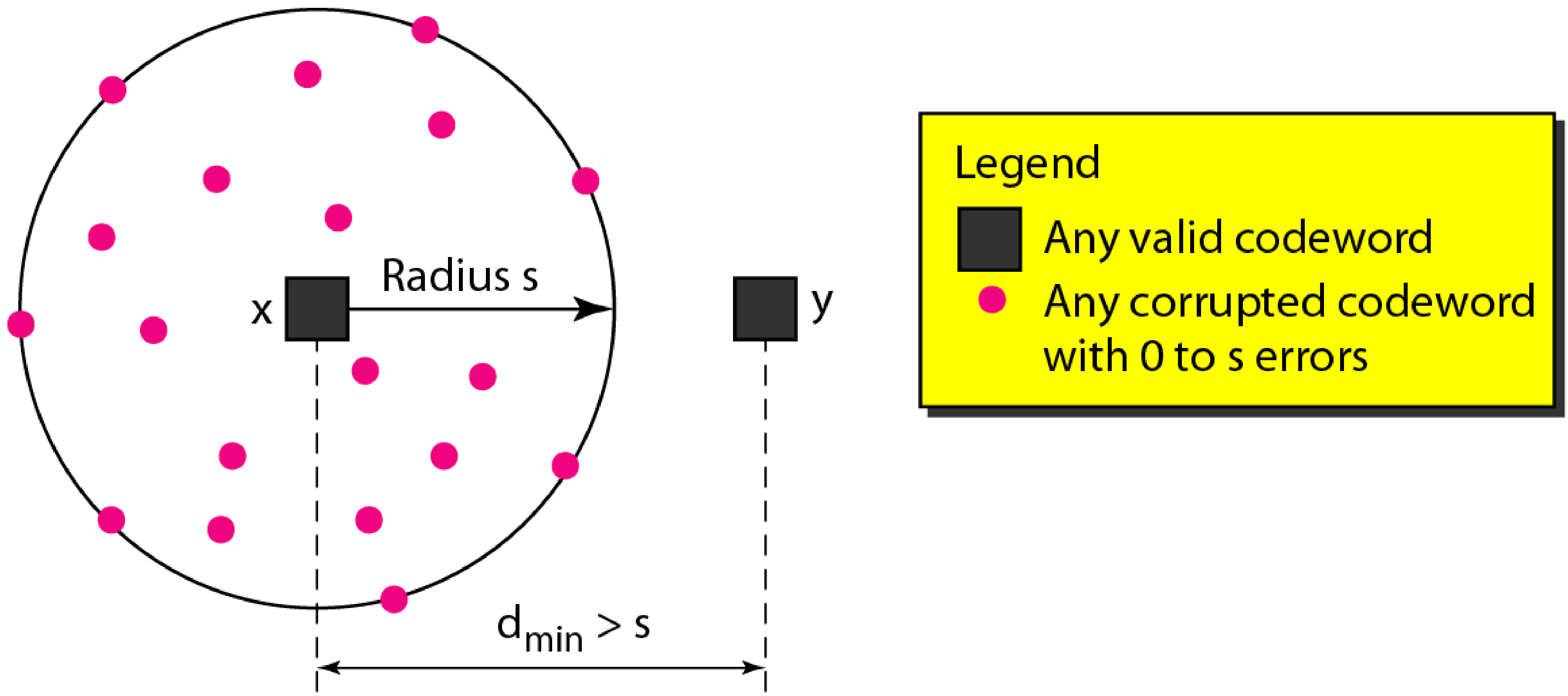
- Two examples:





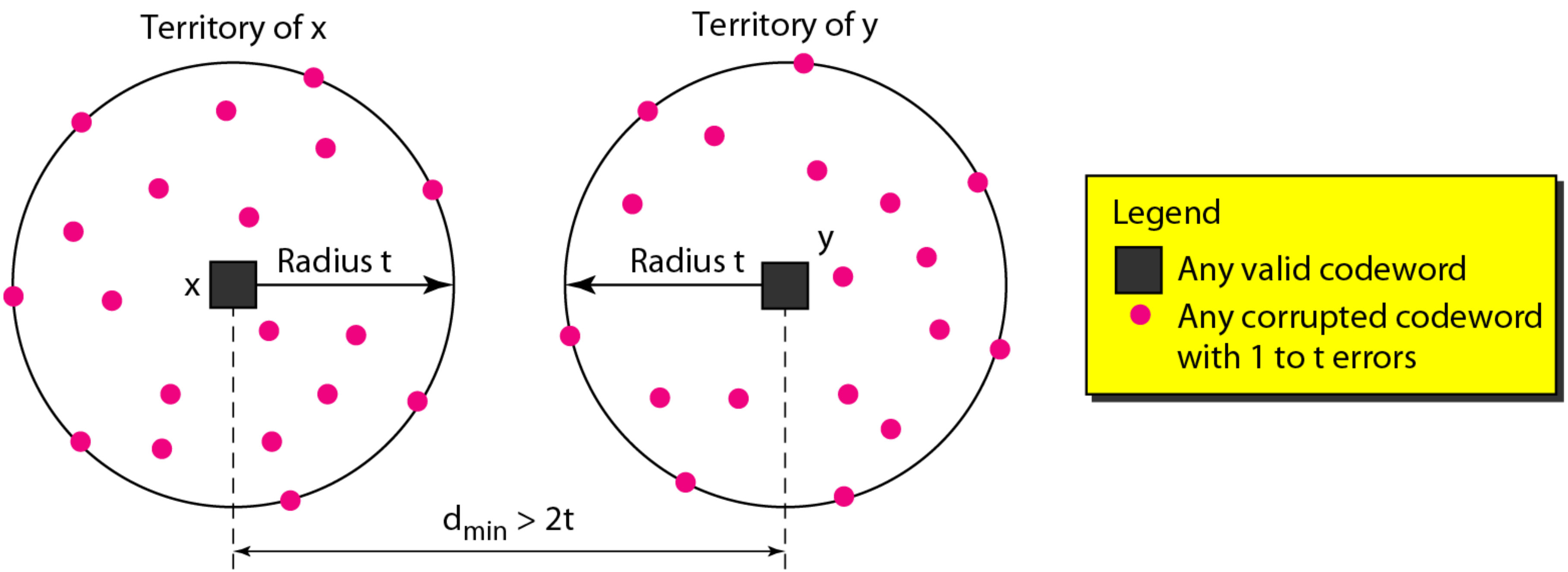
Error detection

To detect d errors, you need a distance d+1 code.



Error correction

To correct d errors, you need a distance 2d+1 code.



Example

$$S = \{\ 00000000,\ 00001111,\ 11110000,\ 11111111\ \}$$

Parity bit – already discussed

- A single parity bit is appended to the data
  - Choosen according to the number of 1 bits in the message
    - odd or even

- An example using even parity
  - Original message: 1011010
  - A 0 bit is added to the end: 10110100
  - m=8 and r=1 in this case

- The distance of this code is 2, since any single-bit error produces a codeword with the wrong parity.

Checksums

- Idea:
  - Add up the bytes in the data
  - Include the sum in the frame

- Use ones-complement arithmetic
- Lower overhead than parity: 16 bits per frame
- But, not resilient to errors
  - Why? $\boxed{1}$101001 + $\boxed{0}$101001= 10010010
- Used in UDP, TCP, and IP

Cyclic Redundancy Check (CRC)

- Uses field theory to compute a semi-unique value for a given message

- Much better performance than previous approaches
  - Fixed size overhead per frame (usually 32-bits)
  - Quick to implement in hardware
  - Only 1 in 2^{32} chance of missing an error with 32-bit CRC

CRC (Cyclic Redundancy Check)

- Polynomial code
  - Treating bit strings as representations of polynomials with coefficients of 0 and 1.
- CRC
  - Add k bits of redundant data to an n-bit message.
  - Represent n-bit message as an n-1 degree polynomial;
    - e.g., MSG=10011010 corresponds to $M(x) = x^7 + x^4 + x^3 + x^1$.
  - Let k be the degree of some divisor polynomial G(x);
    - e.g., $G(x) = x^3 + x^2 + 1$.
    - Generator polynomial
      - Agreed upon it in advance

CRC

- Transmit polynomial P(x) that is evenly divisible by G(x), and receive polynomial P(x) + E(x);
  - E(x)=0 implies no errors.

- Recipient divides (P(x) + E(x)) by G(x);
  - the remainder will be zero in only two cases:
    - E(x) was zero (i.e. there was no error),
    - or E(x) is exactly divisible by C(x).

- Choose G(x) to make second case extremely rare.

A basic example with numbers

- Make all legal messages divisible by 3
- If you want to send 10
  - First multiply by 4 to get 40
  - Now add 2 to make it divisible by 3 = 42
- When the data is received ..
  - Divide by 3, if there is no remainder there is no error
  - If no error, divide by 4 to get sent message
- If we receive 43, 44, 41, 40, then error
- 45 would not be recognized as an error

Mod 2 arithmetic

- Operations are done modulo 2

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A - B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

```
  0110111011
+ 1101010110
= 1011101101
```

```
 1111
+1010
=====
 0101
```

```
    11001
    x 101
    =====
    11001
+ 11001
=========
  1111101
```

A basic example with polynomials

- Sender:
  - multiply $M(x) = x^7 + x^4 + x^3 + x^1$ by x^k; for our example, we get
    - $x^{10} + x^7 + x^6 + x^4$ (10011010000);
  - divide result by $C(x)$ (1101);

```
                        11111001
Generator   1101 | 10011010000      Message
                   1101
                    1001
                    1101
                     1000
                     1101
                      1011
                      1101
                       1100
                       1101
                          1000
                          1101
                           101      Remainder
```

Send 10011010000 + 101 = 10011010101,
since this must be exactly divisible by $C(x)$;

Further properties

- Want to ensure that $G(x)$ does not divide evenly into polynomial $E(x)$.
- All single-bit errors, as long as the x^k and x^0 terms have non-zero coefficients.
- All double-bit errors, as long as $G(x)$ has a factor with at least three terms.
- Any odd number of errors, as long as $G(x)$ contains the factor $(x + 1)$.
- Any "burst" error (i.e sequence of consecutive errored bits) for which the length of the burst is less than k bits.
- Most burst errors of larger than k bits can also be detected.