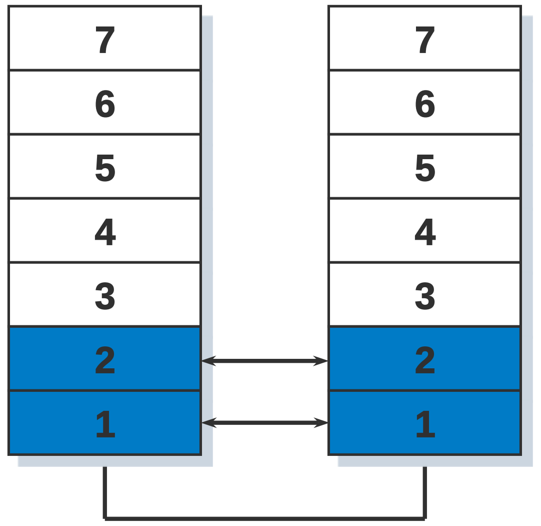
TI III: Operating Systems & Computer Networks
Host-to-Network

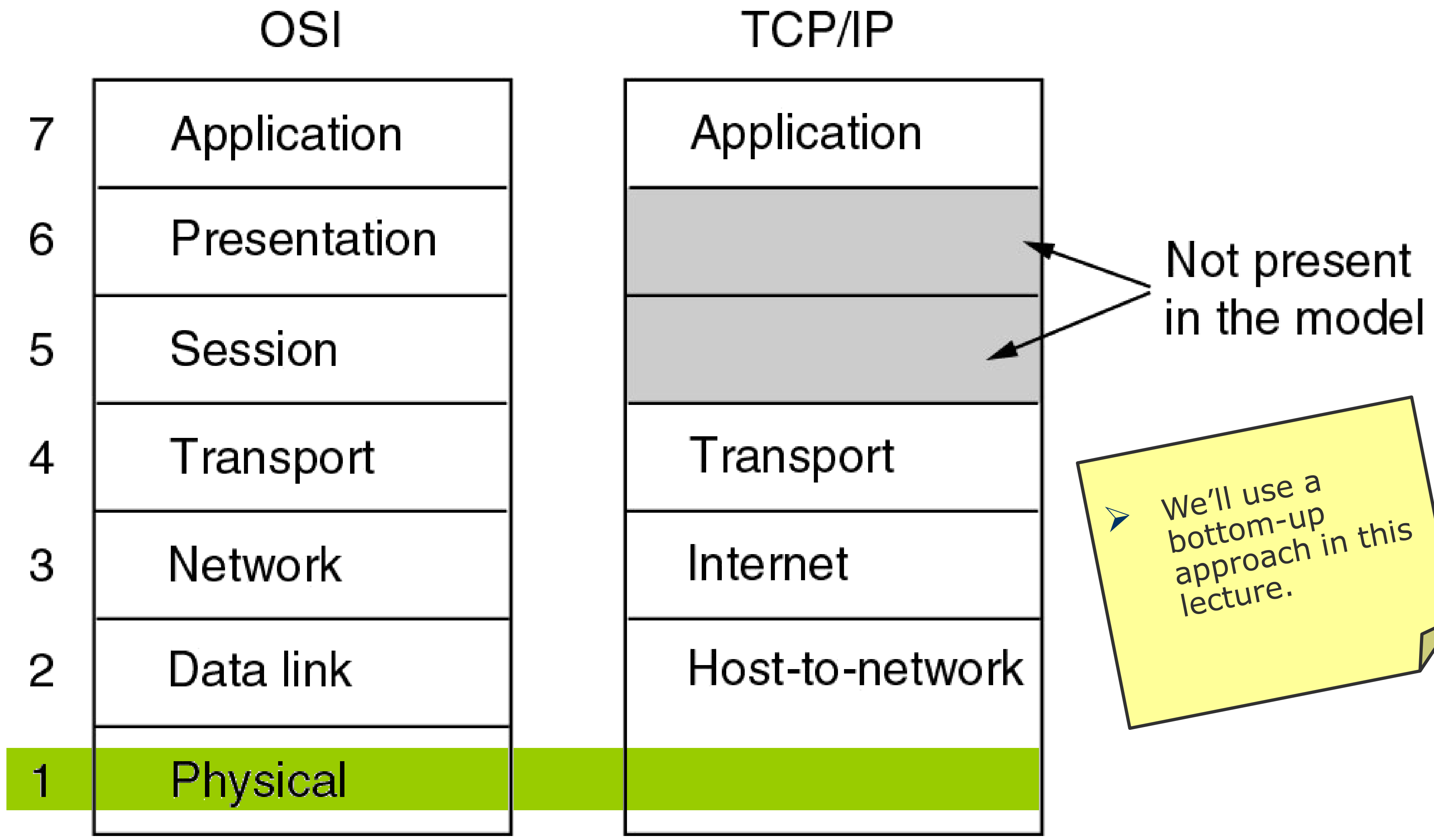
Prof. Dr.-Ing. Jochen Schiller
Computer Systems & Telematics
Freie Universität Berlin, Germany



Content

- Many hidden slides – pointers to more information, covered in the Computer Networking course (3rd year BSc/1st year MSc)!

Physical Layer



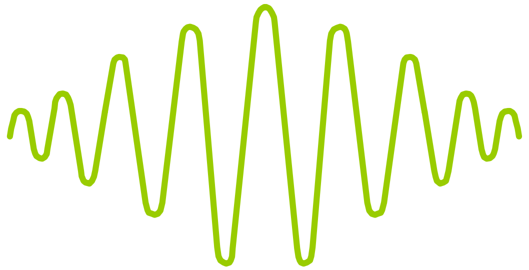
- We'll use a bottom-up approach in this lecture.

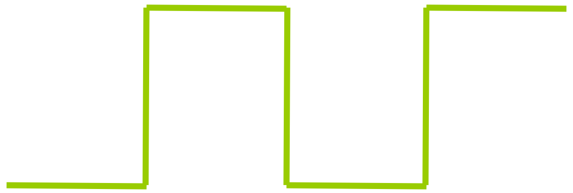
Transmission of data requires physical representation of data

A Signal is the physical representation of data

An analog signal is a sequence of continuous values

A digital signal is a sequence of discrete values





Computers deal with digital signals

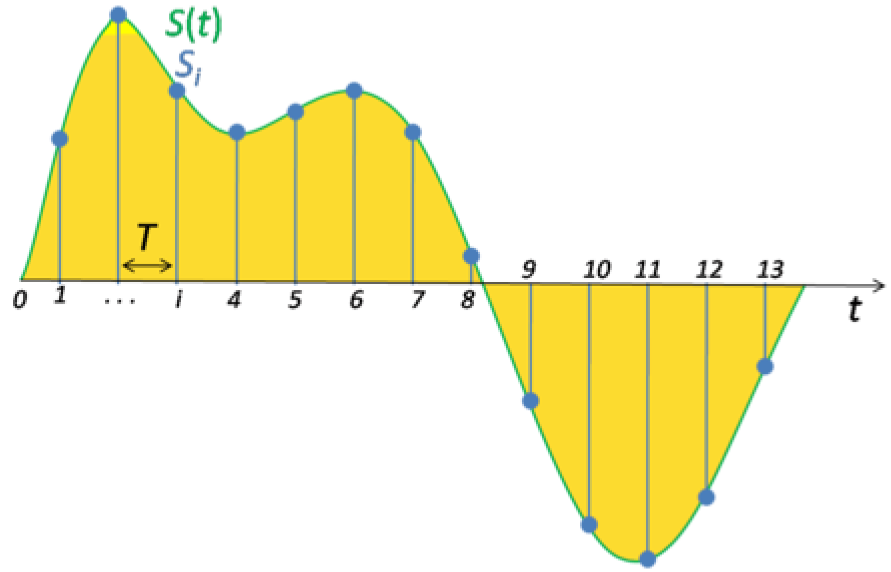
The physical layer transmits data based on signals through space

The need to convert - **Quantization**

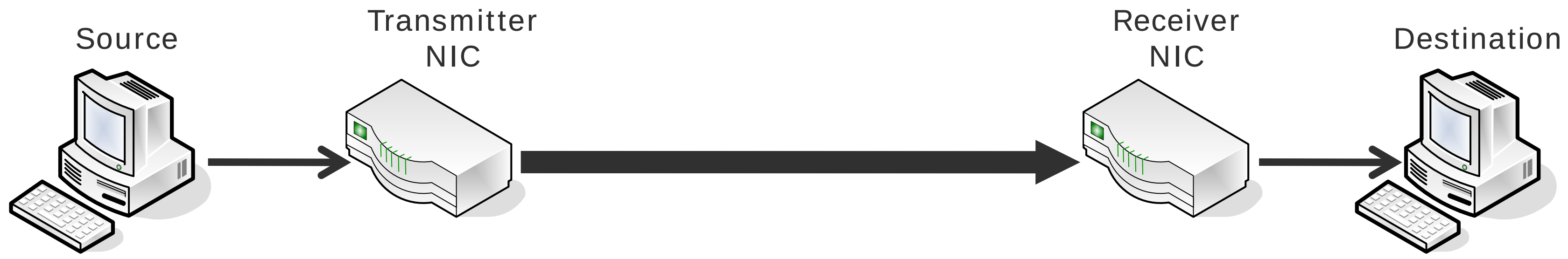
- Computers can only deal with digital data => discrete signal
- Physical mediums are by nature analog => continuous signal
- Must convert from digital signal to analog signal (and vice versa)

The need to measure - **Sampling**

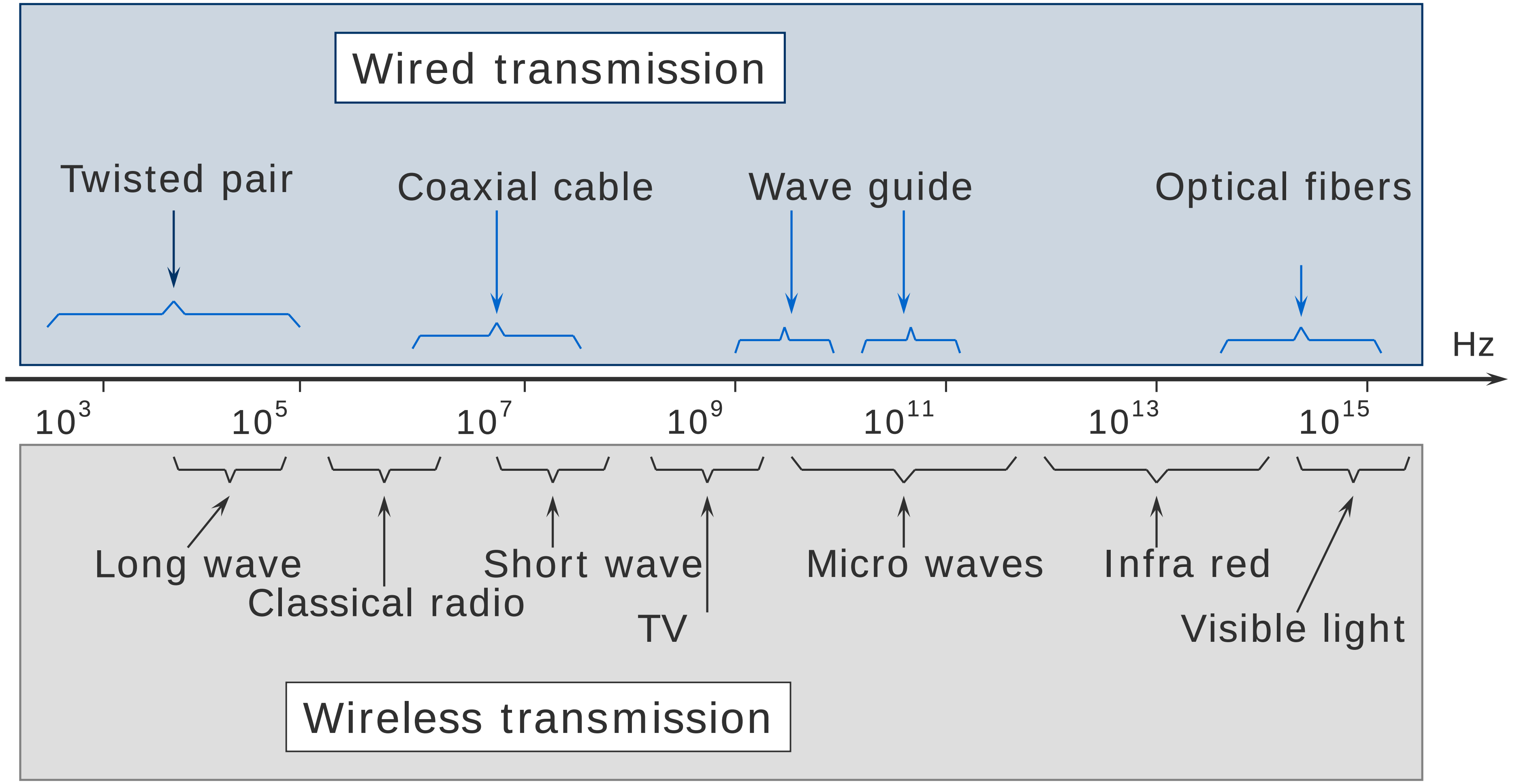
- Computers can only deal with discrete time
- Physical mediums' state vary continuously
- Must rely on periodical measurements of the physical medium (> 2 * bandwidth of the signal)



Source: https://en.wikipedia.org/wiki/Sampling_(signal_processing)



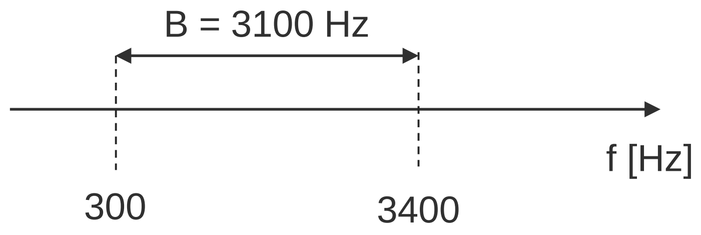
Electromagnetic Spectrum



Definition of Bandwidth & Throughput – Shannon brings this together!

Bandwidth

- Interval of the spectrum, difference between upper and lower frequencies, measured in Hertz
- Example: classical telephony over copper wires supports 300 – 3400Hz, thus bandwidth B = 3400Hz - 300Hz = 3100 Hz



Throughput

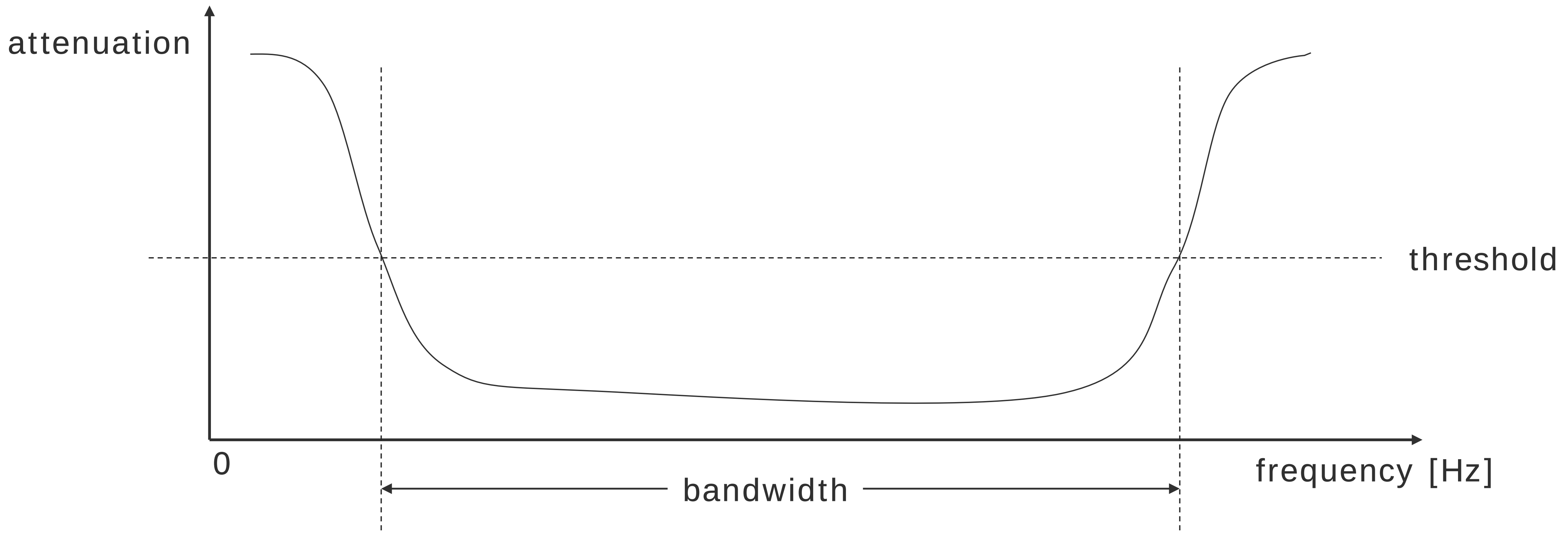
- Amount of bits per second transmitted over a system, measured in bit/s

Shannon brings this together

- Achievable data rate is limited by noise in real systems
- More precisely: by relationship of signal strength compared to noise, i.e., Signal-to-Noise Ratio (SNR, S/N)
- S: signal strength; N: noise level; B: bandwidth in Hz;
- S/N commonly expressed in dB: S/N [dB] = $10 \times \log_{10}(S/N)$

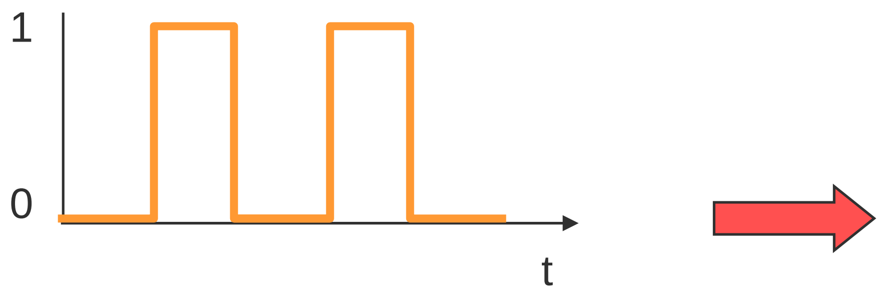
$$\boxed{\textbf{maximum throughput [bit/s]} = B \log_2 (1 + S/N)}$$

Real technical systems are always bandwidth-limited

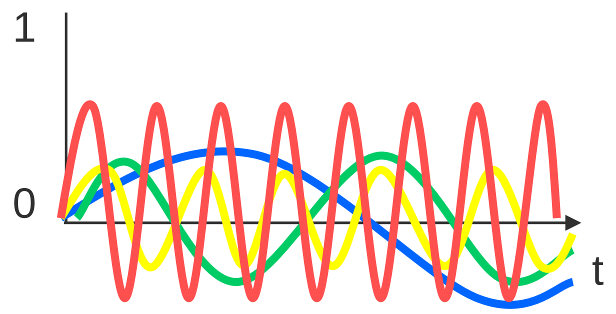


Fourier representation of periodic signals

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft)$$



ideal periodic signal



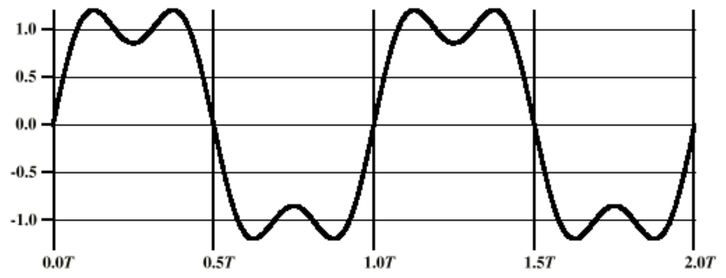
real composition
(based on harmonics)

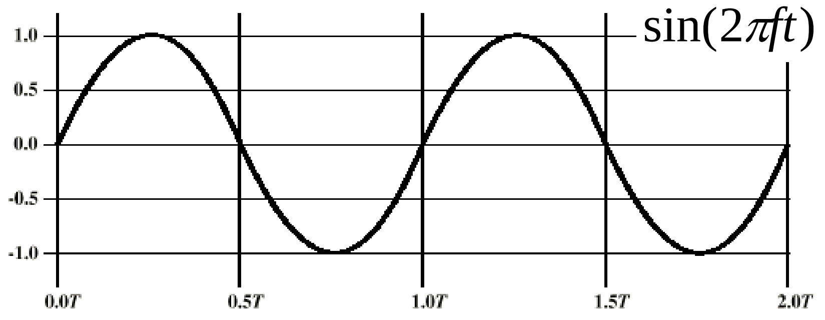
Composing periodic signals

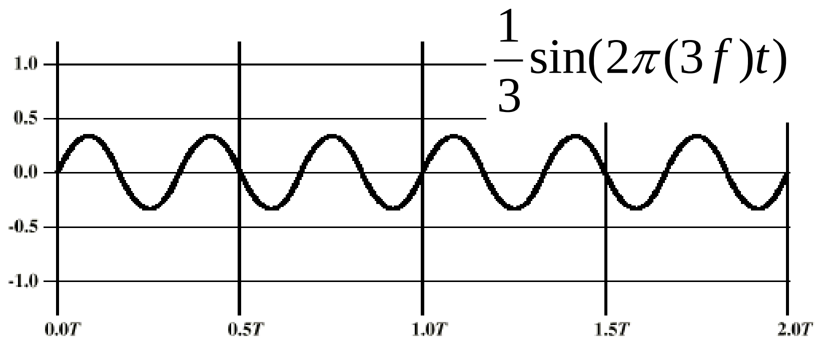
Composing multiple frequencies:

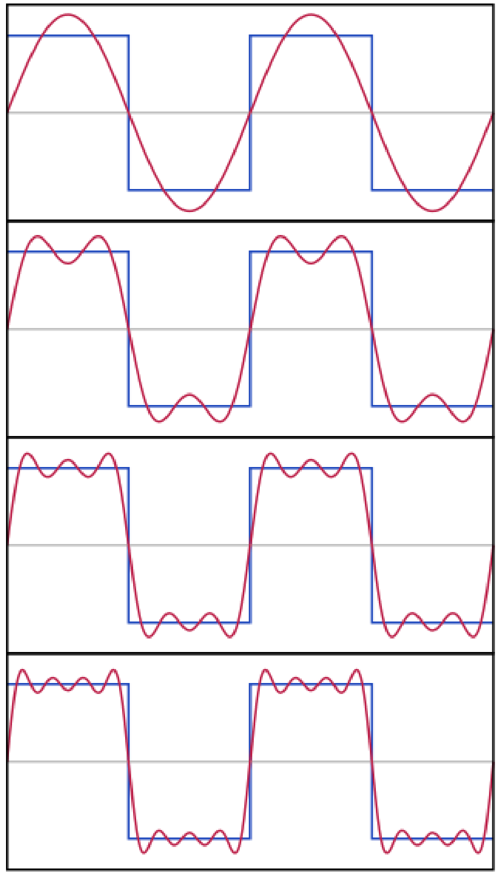
- Example:

$$s(t) = \sin(2\pi ft) + \frac{1}{3}\sin(2\pi(3f)t)$$





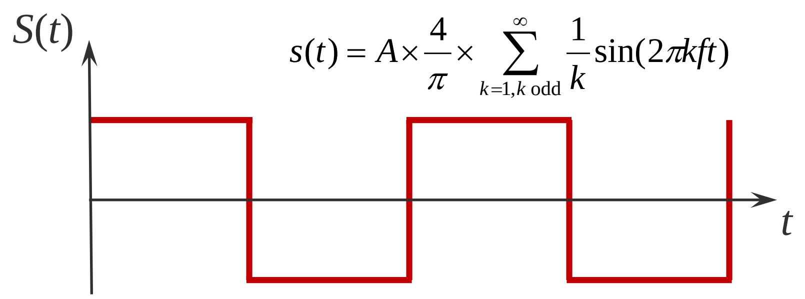




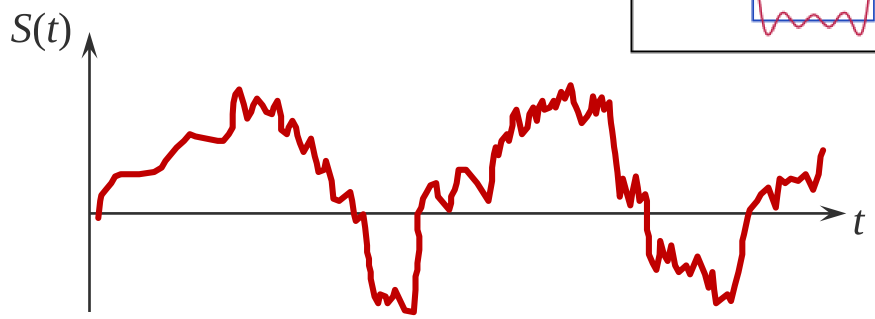
- Components of the signal are sine waves of frequencies *f* and 3*f*

- Composing a lot of different frequencies generates a variety of signals

$$s(t) = A \times \frac{4}{\pi} \times \sum_{k=1, k \text{ odd}}^{\infty} \frac{1}{k}\sin(2\pi kft)$$

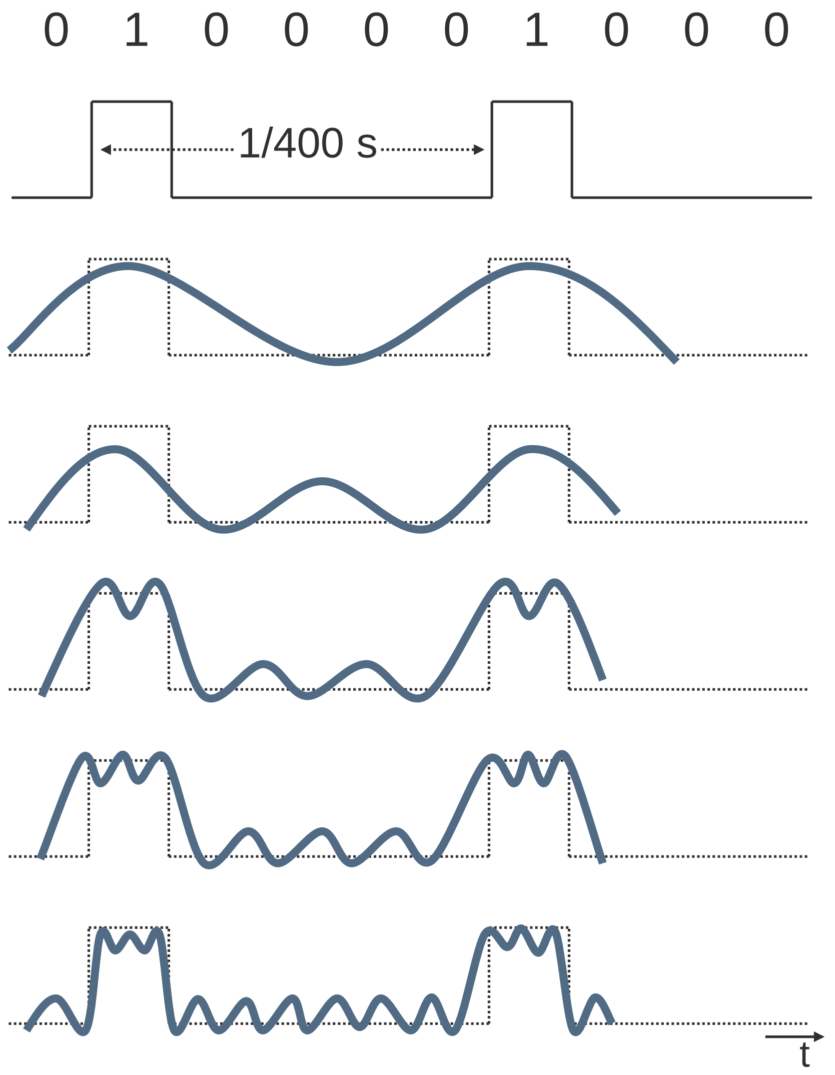


Digital signal (mixed frequencies & amplitudes)



Voice signal (mixed frequencies & amplitudes)

Bit rate *vs.* bandwidth: Medium limiting harmonics



Transmitted data (bit rate = 2kbps)

Ideal, requires infinite bandwidth!	
Bandwidth 500 Hz	1. Harmonic
Bandwidth 900 Hz	1.+2. Harmonics
Bandwidth 1300 Hz	1.-3. Harmonics
Bandwidth 1700 Hz	1.-4. Harmonics
Bandwidth 2500 Hz	1.-5. Harmonics

Questions & Tasks

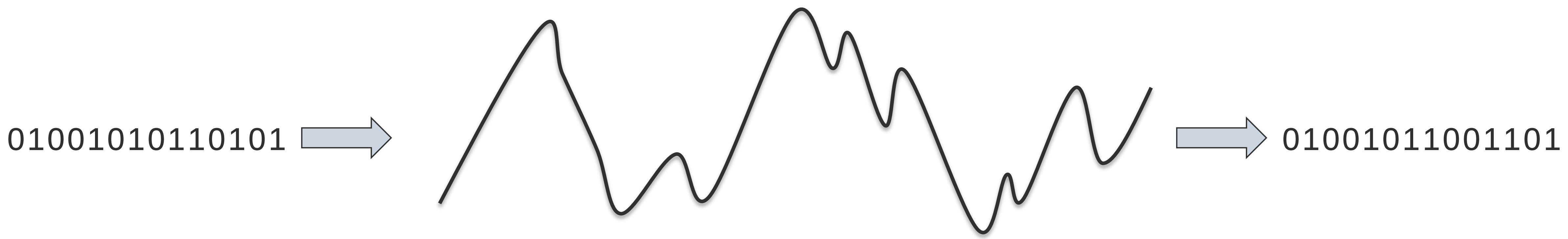
- Where/when do quantization and sampling create errors? Can we avoid these errors?
- Taking the classical telephony example: How often should the system sample the signal?
- What is the maximum data rate without noise? Is this realistic?
- Give simple examples of bandwidth limitations in everyday life!

Tasks of Physical Layer

Responsible for turning a logical sequence of bits into a physical signal that can propagate through a medium
- Many forms of physical signals
- Signals are limited by their propagation in a physical medium
- Limited bandwidth, attenuation, dispersion, and by noise

Includes connectors, media types, voltages, …

No error correction!



Tasks of Physical Layer

Bits can be combined into multi-valued symbols for transmission

- ➢ Gives rise to the difference in **data rate** (bits per sec, bit/sec) and **baud rate** (symbols per sec)

Two types

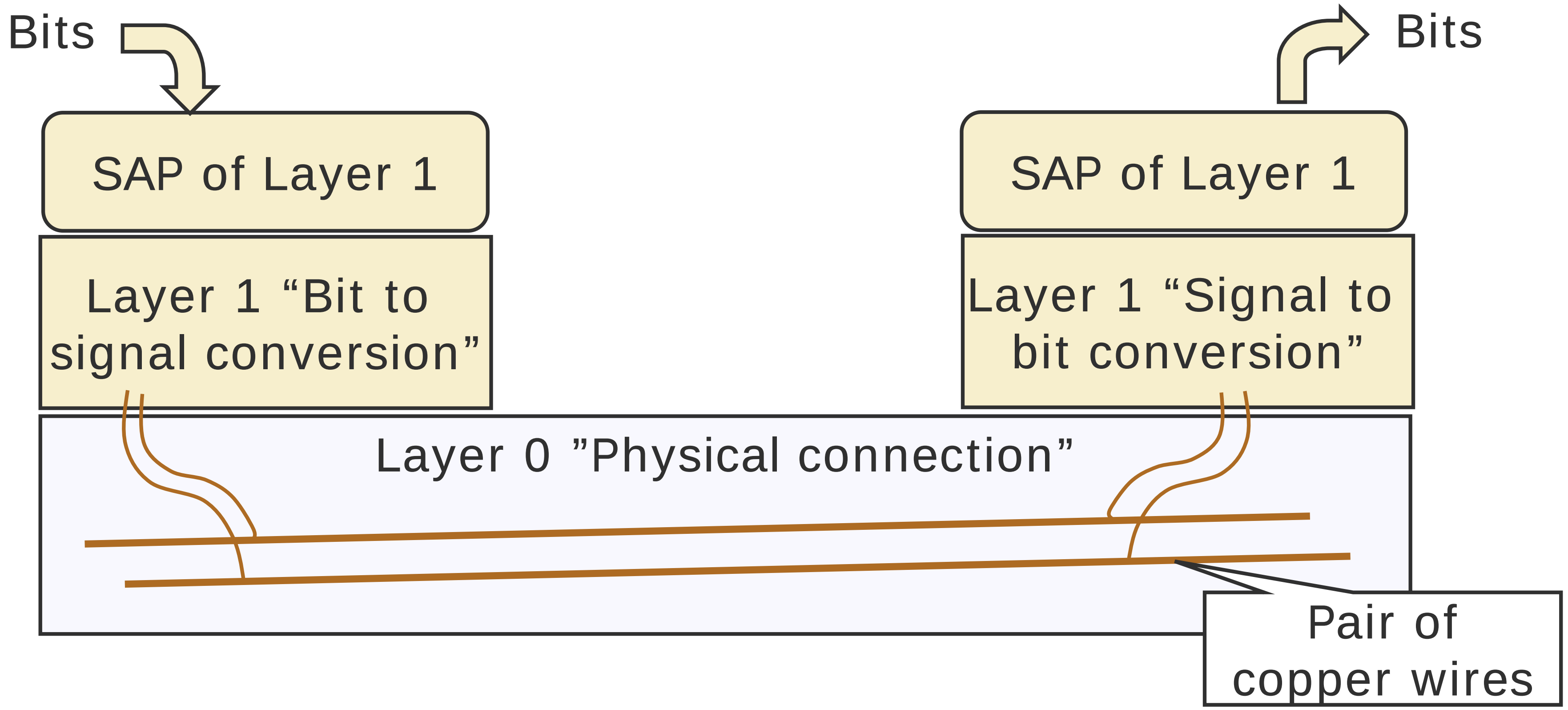
- Baseband transmission
- Broadband transmission

Basic Service of Physical Layer: Transport Bits

Physical layer should enable transport of bits between two locations A and B

Abstraction: Bit sequence (in order delivery)

- But no guarantee on correct transmission of bits

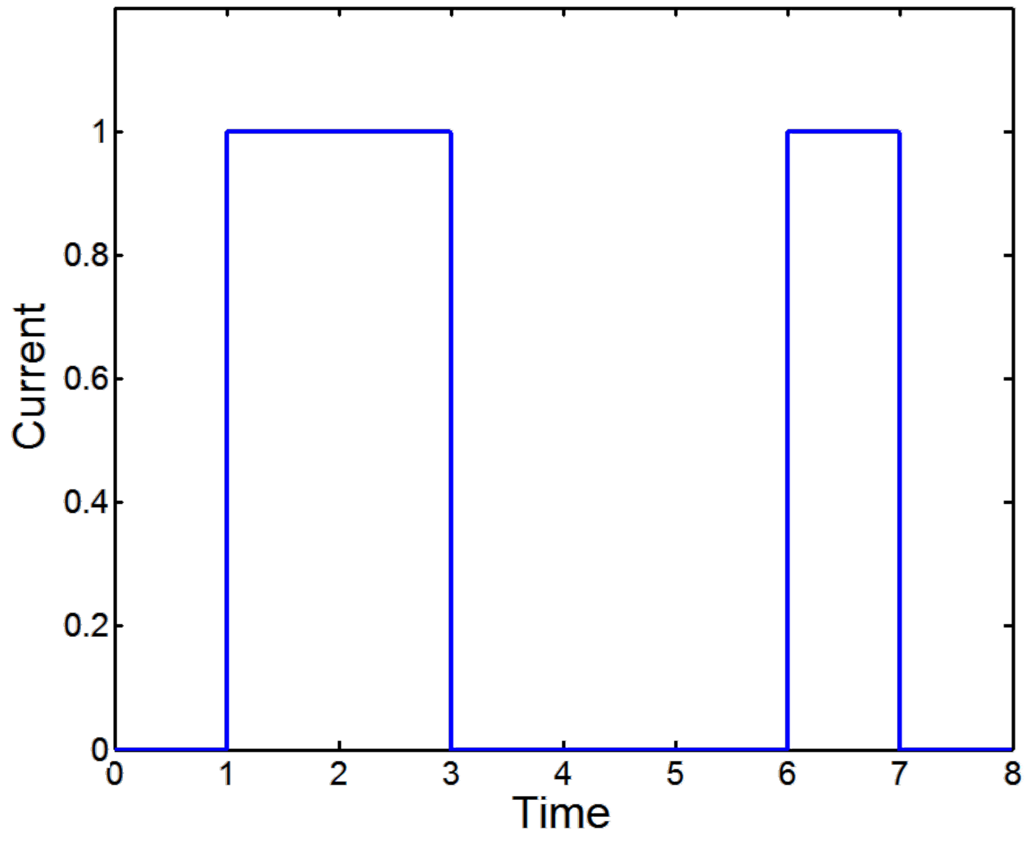

Bits
Bits
SAP of Layer 1
SAP of Layer 1
Layer 1 “Bit to signal conversion”
Layer 1 “Signal to bit conversion”
Layer 0 ”Physical connection”
Pair of copper wires

Example: Transmit Bit Pattern for Character “b”

Represent character “b” as a sequence of bits

Use ASCII code ➞ “b” = 98, as binary number 01100010

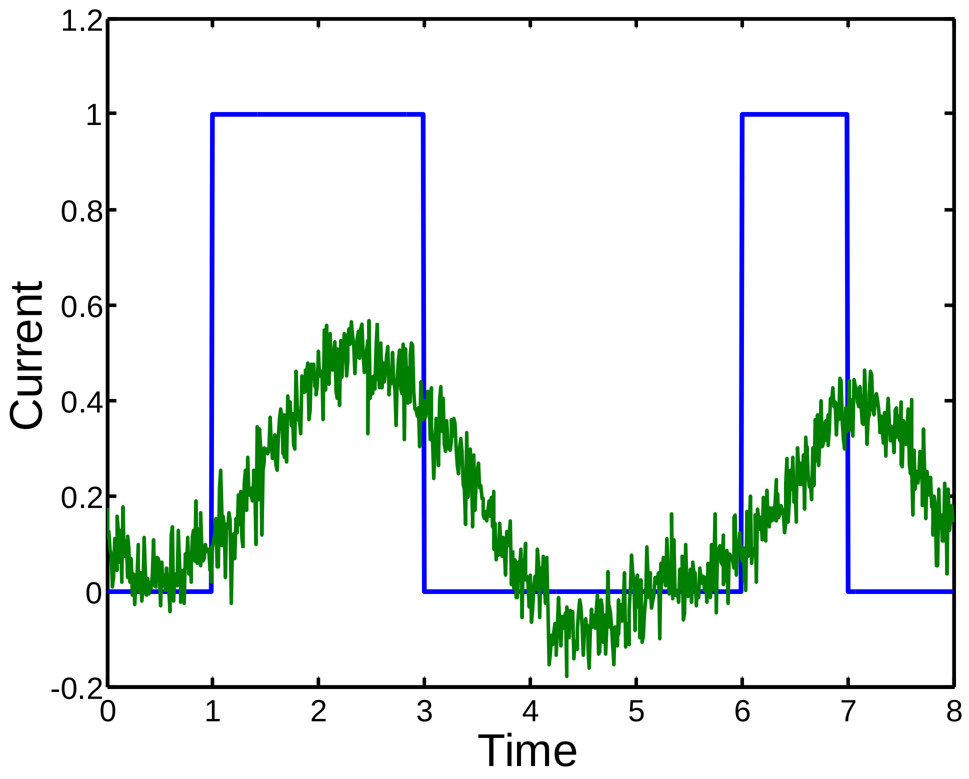
Resulting current on the wire:



Note: Abstract ***data*** is represented by physical ***signals*** – changes of a physical quantity in time or space!

What Arrives at the Receiver?

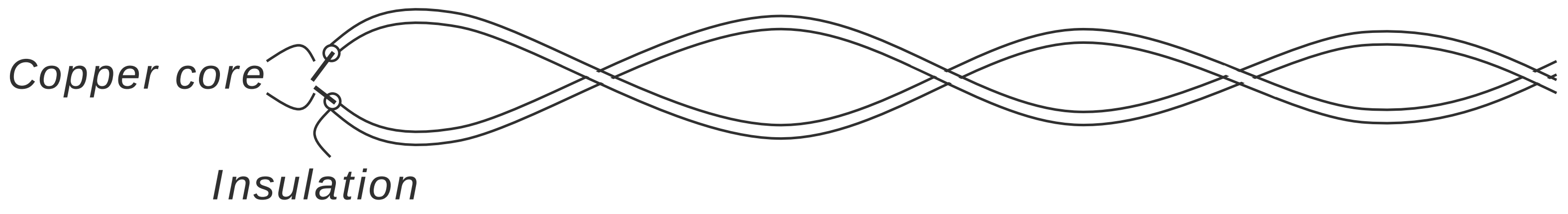
Typical pattern at the receiver:



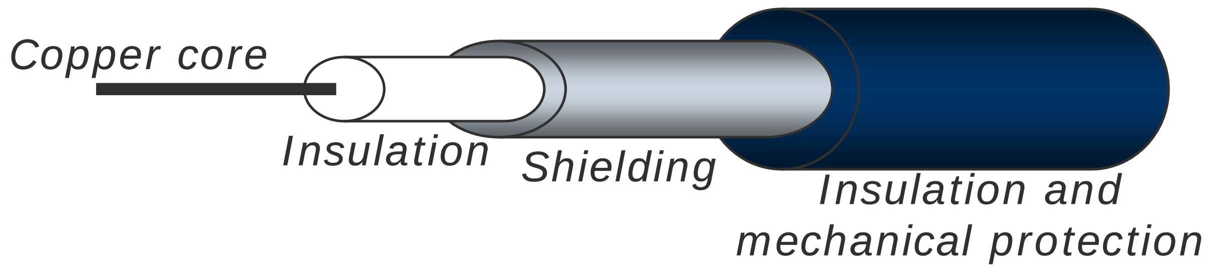
➢What is going on here and how should we convert the signal back to a “b”?

Wires

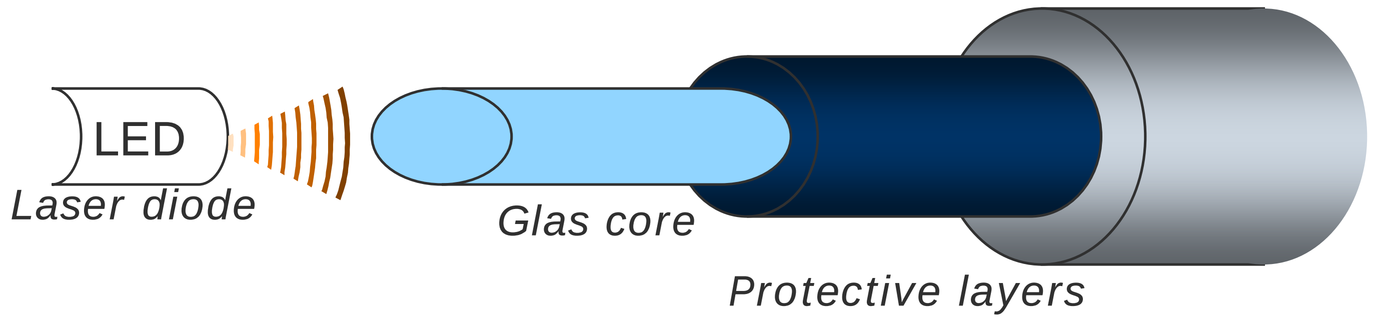
Twisted pair



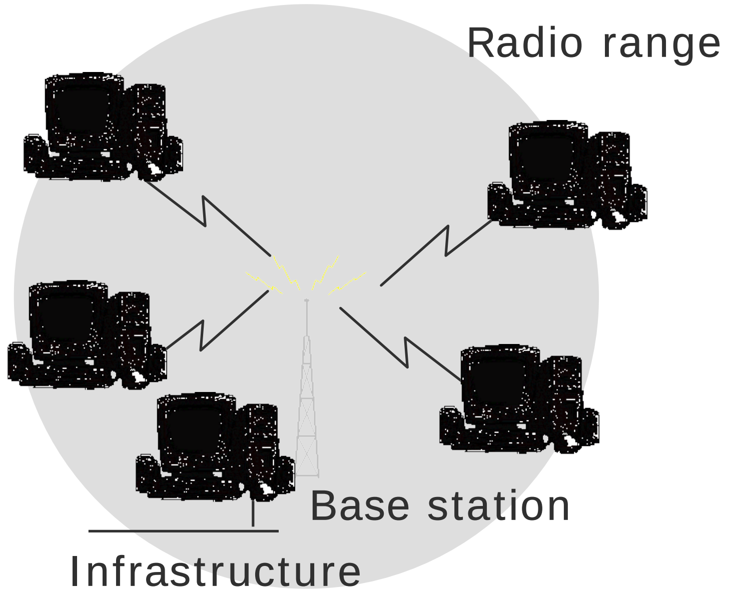
Coaxial

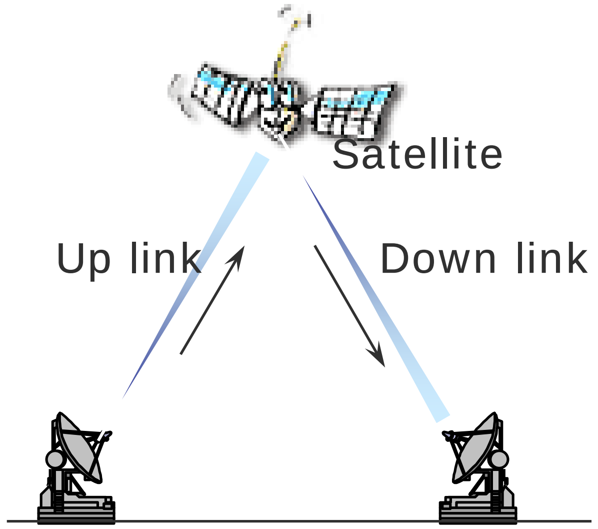


Optical fiber



Wireless Transmission





Examples: 802.11 (WLAN), mobile phones, Bluetooth

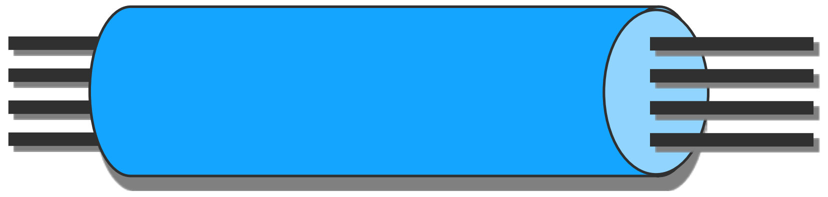
Examples: Television, deep space communication

Multiplexing

Communication medium is scarce resource

➢Optimize medium usage by *multiplexing*

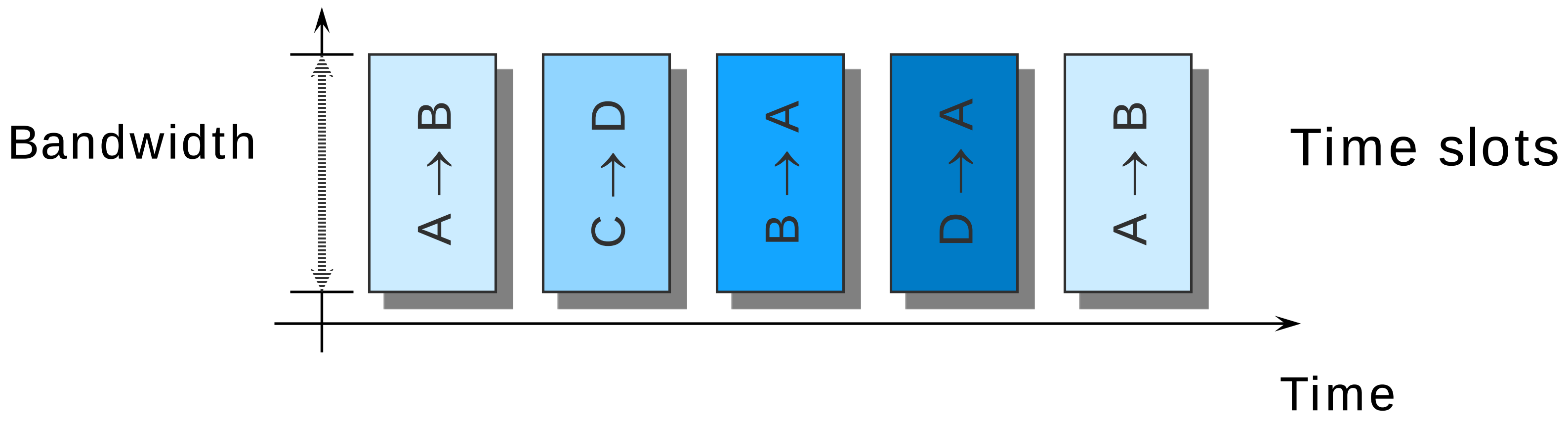
Space Division Multiplexing (SDM)



Bundling of wires

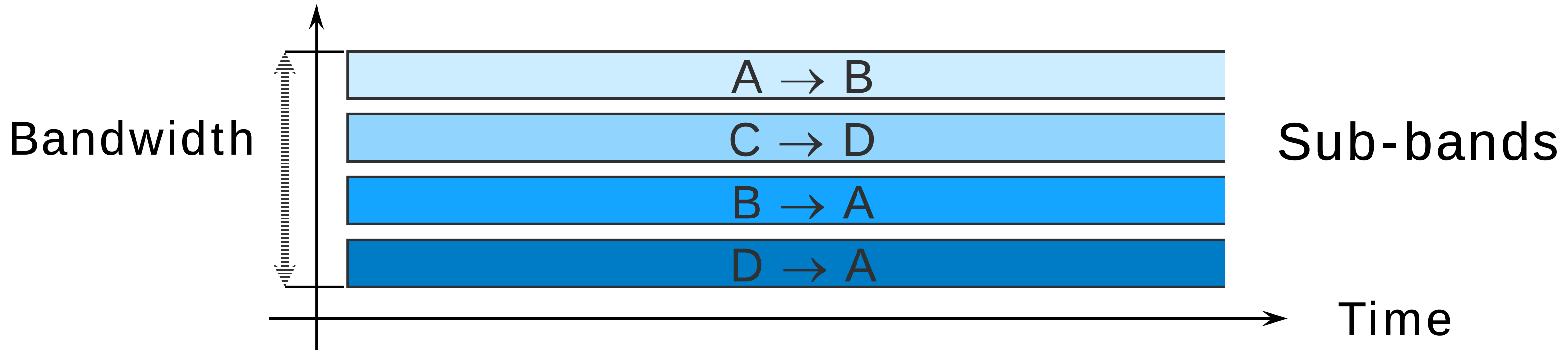
Multiplexing

Time Division Multiplexing (TDM)



Multiplexing

Frequency Division Multiplexing (FDM)

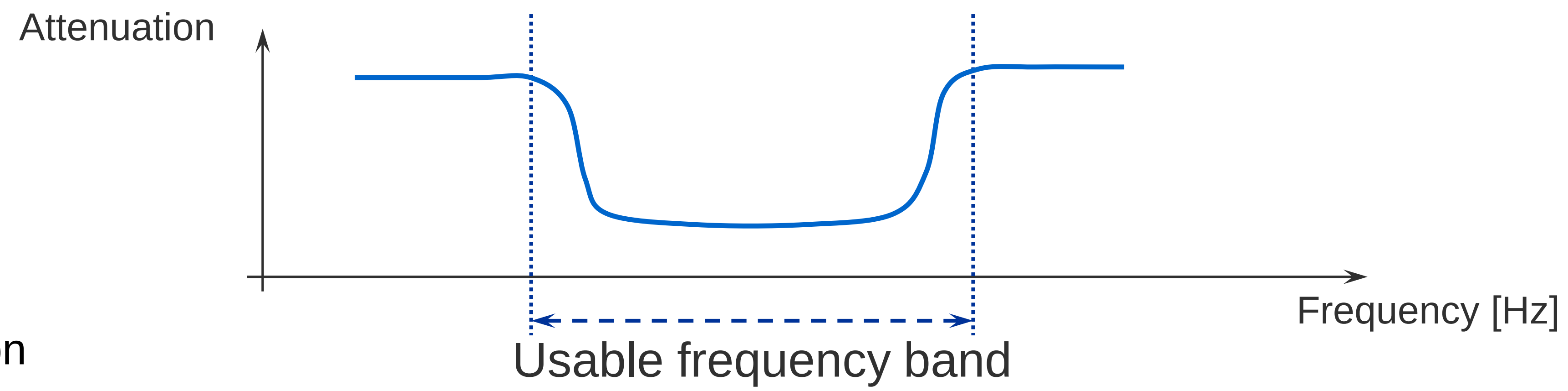


➢Multiplexing in general allows for a more efficient usage of a medium

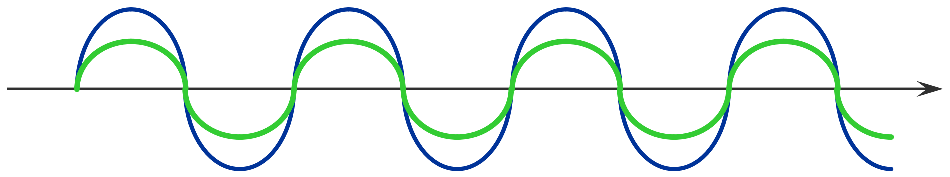
- Discretized resource can be managed, allocated, scheduled...

Typical Effects of Transmission

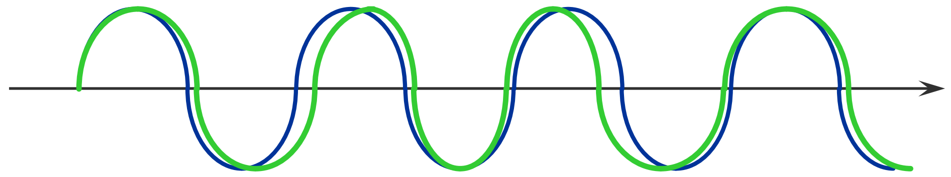
Limited bandwidth



Signal attenuation



Jitter, dispersion, …



Interference

Noise
- Background noise
- Thermal

Echoes
- E.g. at connections

Crosstalk
- E.g. interference across wires

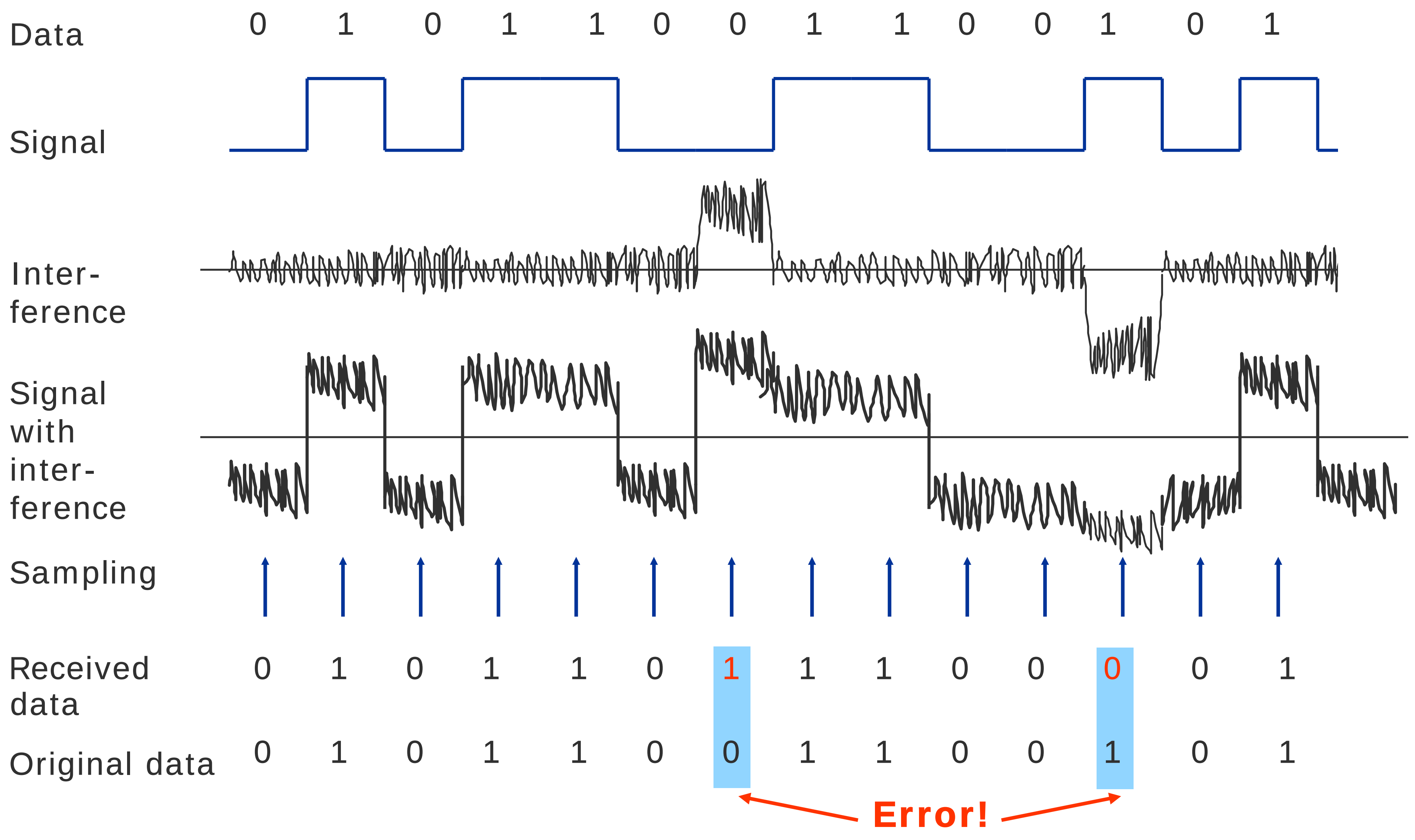
ELF
- Extreme low frequency, e.g. 50/60 Hz AC

Spikes
- Short, high amplitude

…

Plus: attenuation, dispersion, refraction, phase shift …

Example: Results of Interference



When to Sample Received Signal?

How does the receiver know *when* to check the received signal for its value?

- One typical convention: In the middle of each symbol
- But when does a symbol start?
    - The length of a symbol is usually known by convention via the symbol rate

The receiver has to be *synchronized* with the sender at bit level

- Link layer will have to deal with *frame synchronization*
- There is also *character synchronization* – omitted here

Overly Simplistic Bit Synchronization

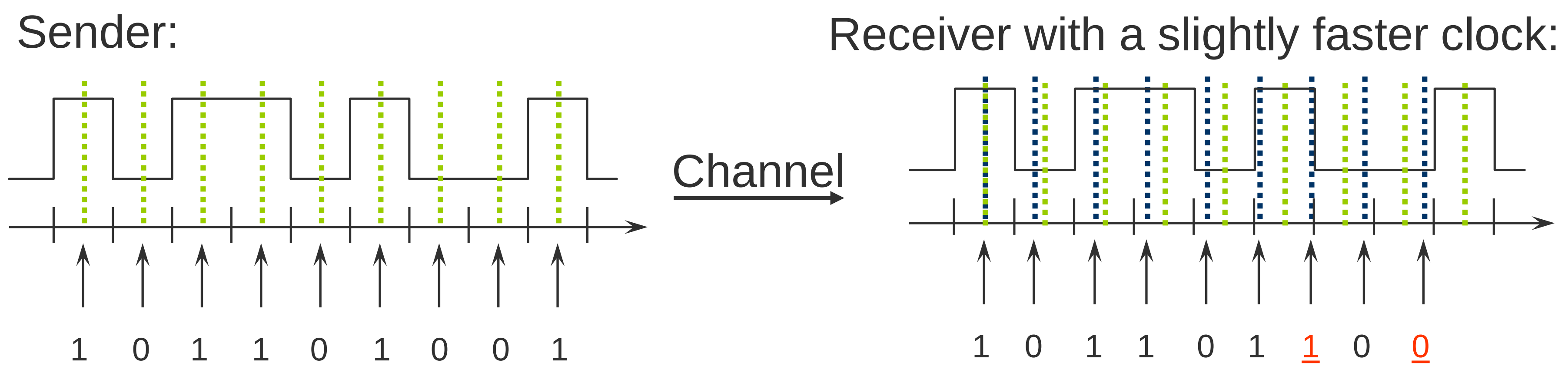
One simple option, assume

- ... that sender and receiver are synchronized at some point in time
- ... that both have an internal clock that tics at every symbol step

➢Usually, this does not work due to *clock drift*

- Two different clocks never stay in perfect synchrony

➢Errors, if synchronization is lost:



Options to Tell Receiver When to Sample

Relying on clock synchronization does not work

1. Provide an explicit clock signal
   - Needs parallel transmission over some additional channel
   - Must be in synch with actual data, otherwise pointless
   - ➢ Useful only for short-range communication

2. Synchronize receiver at crucial points, e.g. start of character or block
   - Otherwise, let receiver clock run freely
   - Relies on short-term stability of clock generators (do not diverge too quickly)

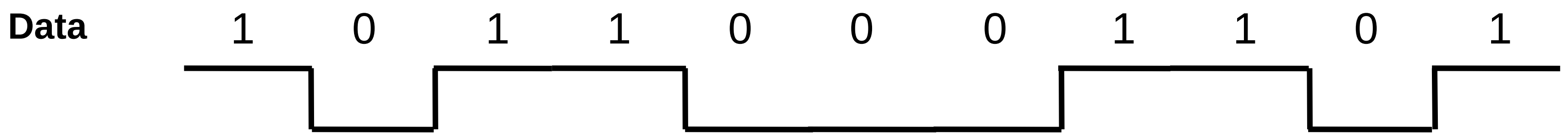
3. Extract clock information from the received signal itself

Extract Clock Information from Signal

Put enough information into data signal itself so that receiver can know immediately when a bit starts/stops

➢Would the simple 0/low, 1/high mapping of bit/symbol work?

Receiver can use 0-1-0 transitions to detect length of a bit



Fails depending on bit sequences, e.g. long runs of 1s/0s

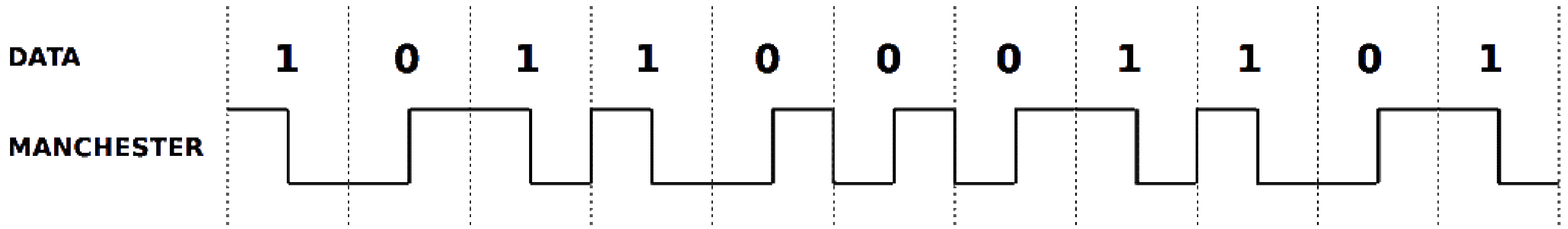
- ➢Receiver can loose synchronization

➢Not nice not to be able to transmit arbitrary data

Manchester Encoding

Idea: At each bit, provide indication to receiver that this is where a bit starts/stops/has its middle

- For a 0 bit, have symbol change in middle of bit from low to high
- For a 1 bit, have the symbol change in middle of bit from high to low



The signal is self-clocking since one transition per period is guaranteed

Disadvantage: bit rate is as half as high as baud rate (i.e. changes in signal values)

Baseband vs. Broadband Transmission

Baseband transmission (all schemes described so far)

- Put digital symbol sequences directly onto the wire
    - At different levels of current, voltage, …
- Problems:
    - Limited bandwidth reshapes signal at receiver
    - Attenuation and distortion depend on frequency
        - Baseband transmissions have many different frequencies because of their wide Fourier spectrum
        - ➢Rectangular signal causes “infinite” spectrum

Possible alternative: broadband transmission

- Needed for wireless transmission (antennas) and frequency division multiplex

Broadband Transmission

Idea: Get rid of wide spectrum needed for baseband transmission

Use carrier (sine wave) for the symbols to be transmitted

- Typically, sine wave has high frequency
- But only a *single* frequency

Pure sine waves have no information, so its shape has to be influenced according to symbols to be transmitted

➢Carrier has to be *modulated* by symbols (widening the spectrum)

Three parameters that can be influenced:

- Amplitude *a*
- Frequency *f*
- Phase ϕ

$$f(t) = a \sin(2\pi f t + \phi)$$

Amplitude Modulation (AM)

Amplitude modulated sine wave $f_A(t)$ is given as

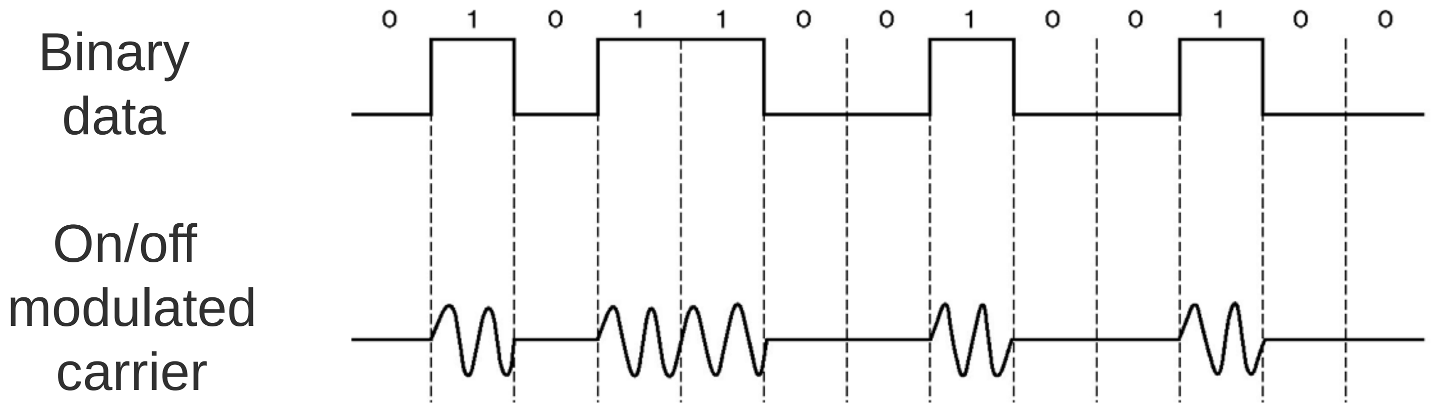
$$f_A(t) = s(t)\sin(2\pi f t + \phi)$$

- Amplitude is given by the signal $s(t)$ to be transmitted

Special cases:

- s(t) is an *analog* signal – *amplitude modulation*
- s(t) is a *digital* signal – *amplitude keying*
- s(t) only takes 0 and *a* as values – *on/off keying*

Examples:



Used in: LF, MF, and HF radio

Frequency Modulation (FM)

Frequency modulated sine wave $f_F(t)$ is given by

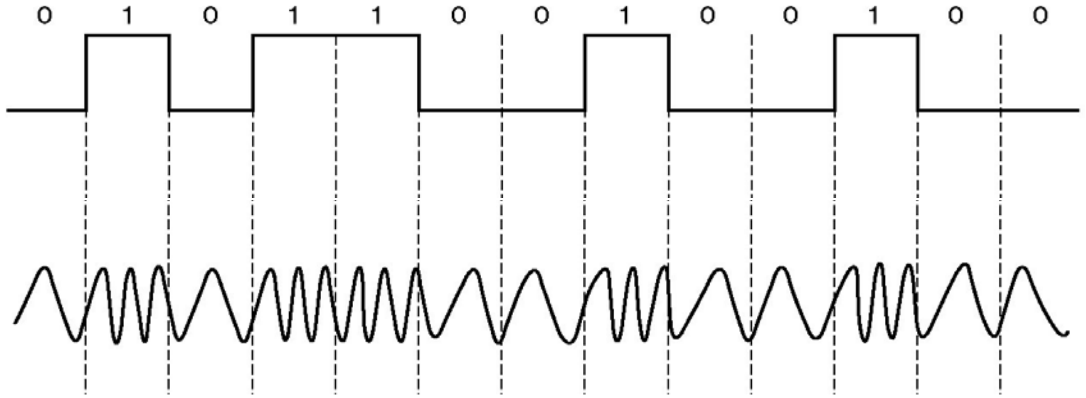
$$f_F(t) = a \sin(2\pi s(t)t + \phi)$$

- Modulation/keying terminology like for AM

Example:

Binary data

Frequency-modulated carrier



Note: s(t) has an additive constant in this example to avoid having frequency zero

Used in: VHF radio, TV, VHS, synthesizers

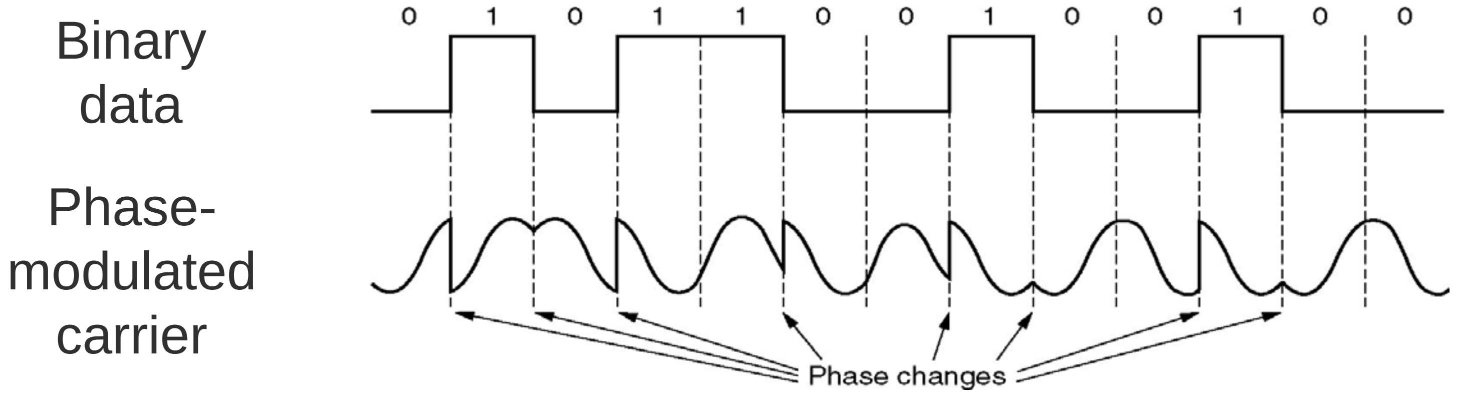
Phase Modulation

Similarly, phase modulated carrier is given by

$$f_P(t) = a \sin(2\pi ft + s(t))$$

- Modulation/keying terminology again similar

Example:



s(t) is chosen such that there are phase changes when the binary data changes

- Typical example for *differential coding*

Use More Than 0 and 1 in Channel

➢Who says we can only use 0 and 1 as possible levels for the transmitted signal?

Suppose the transmitter can generate signals (current, voltage, …) at *four different levels*, instead of just two

- ➢To select one of four levels, *two bits* are required

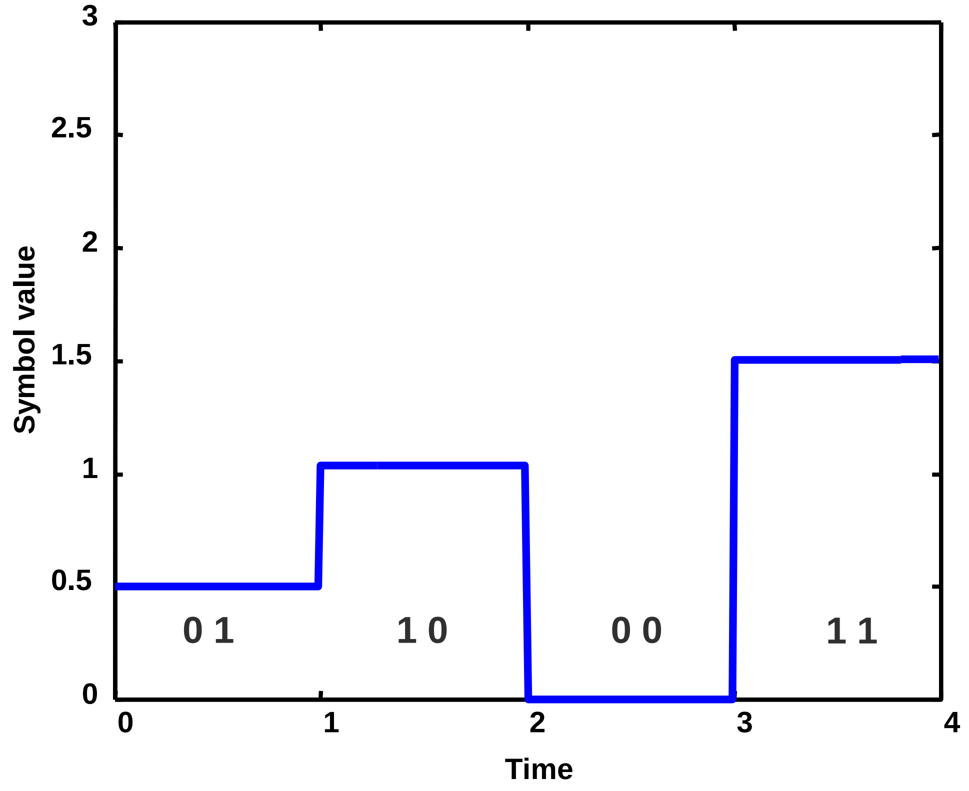
Terminology

- - *Bits* are 0 or 1, used in “higher” layers
- - *Symbols* (with multiple values) are transmitted over channel
- ➢*Symbol rate*: Rate with which symbols are transmitted
    - - Measured in baud
- ➢*Data rate*: Rate with which physical layer processes incoming data bits
    - - Measured in bit/s

Example: Bits and Symbols

Use 4-level symbols to encode 2 bits:

- Map
  00 ➔ 0
  01 ➔ 1
  10 ➔ 2
  11 ➔ 3



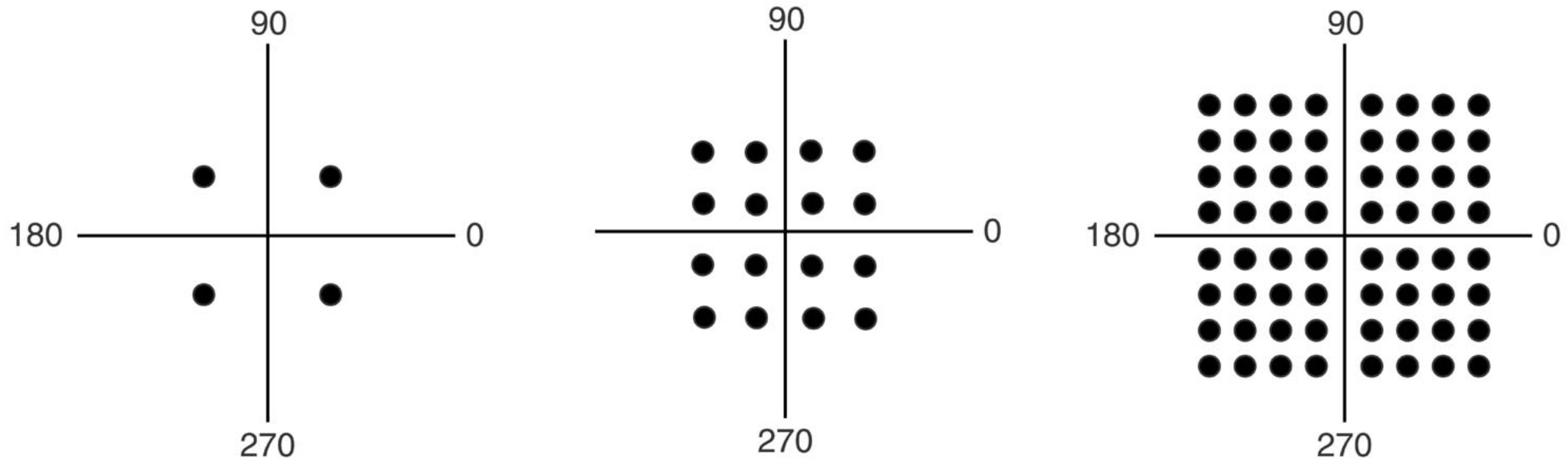
➢Symbol rate is only half the data rate as each symbol encodes two bits

Example: Bits and Symbols

Today’s systems:

- Many bits per symbol
- Often Phase Shift Keying/Amplitude Shift Keying combined

Constellation diagram encodes amplitude and phase of symbols



Data Rate with Multi-valued Symbols

Using symbols with multiple values increases data rate

Nyquist's formula:

$$\boxed{\text{maximum data rate [bit/s]} = 2H \log_2(V)}$$

V: number of symbol values

H: bandwidth in Hz

Indicates that unlimited data rate can be achieved when enough symbol levels are used

But: More and more symbol levels have to be spaced closer and closer together

- Even small random noise would then result in one symbol being misinterpreted for another

Achievable Data Rate with Noise

Achievable data rate is limited by noise

- More precisely: by relationship of signal strength compared to noise, i.e., Signal-to-Noise Ratio (SNR, S/N)

Shannon's formula:

$$\boxed{\text{maximum data rate [bit/s]} = H \log_2 (1 + S/N)}$$

S: signal strength

N: noise level

H: bandwidth in Hz

S/N commonly expressed in dB:

- S/N [dB] = $10 \times \log_{10}(S/N)$

This theorem formed the basis for information theory

Modem Technologies

Cable modem

- Data transmission via broadband cable (TV)
- Infrastructure must be bi-directional
- Data rates in the Gbit/s range but always shared medium

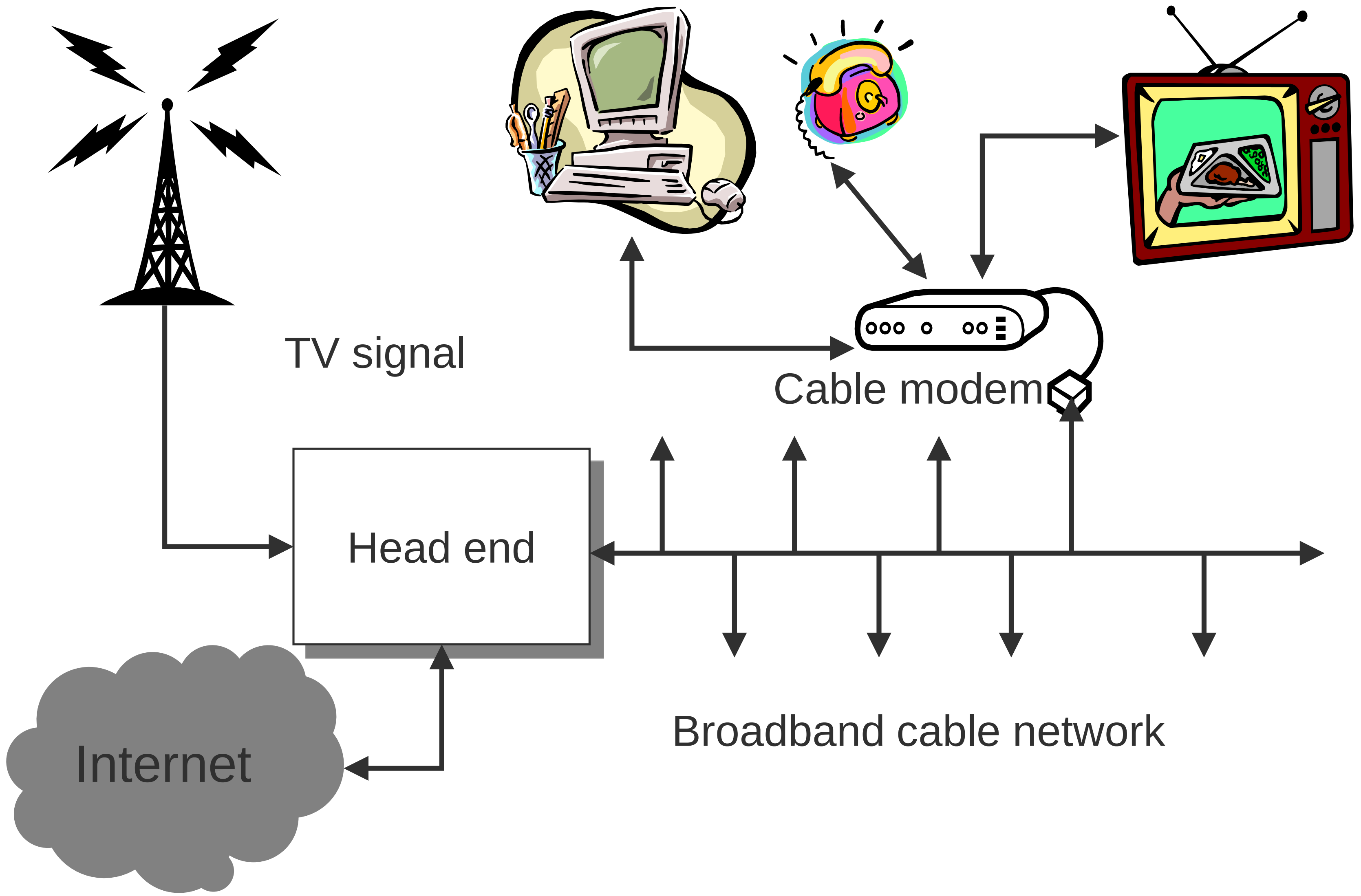
Powerline communications

- Data transmission via power lines
- Couple high-frequency signals into standard power lines
- Data rates up to several Mbit/s but shared medium

xDSL modems

- Higher data rates using conventional phone lines
- Typical data rates up to 16 Mbit/s downstream, up to 1 Mbit/s upstream (ADSL)
- Not everywhere, rates depend on location, interference …
- Special technologies for faster response (“FastPath”)
- Up to 50/100/200 Mbit/s at certain places (VDSL)
- Symmetrical for companies/servers

Broadband Cable: Digital TV Plus Internet



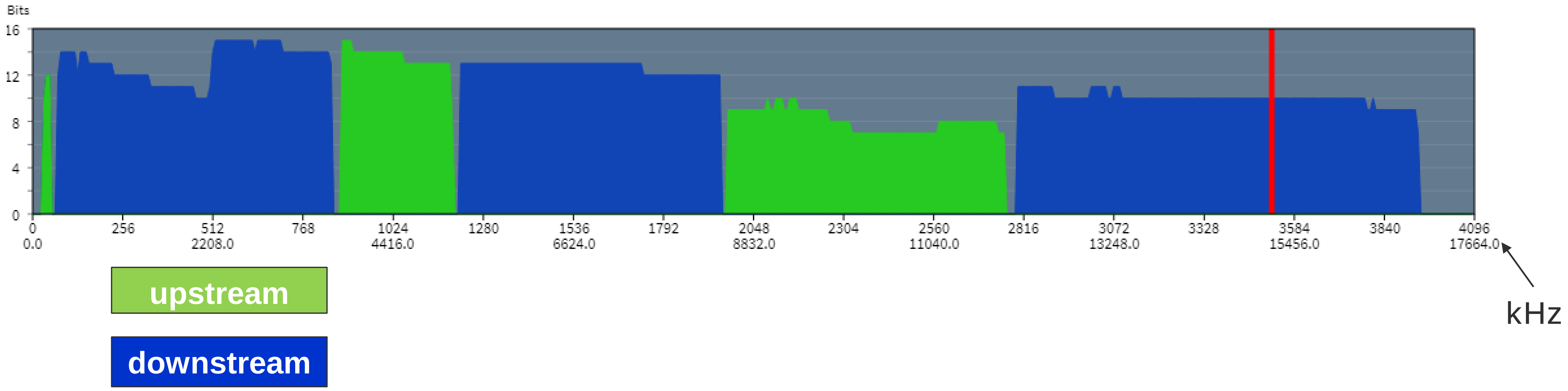
xDSL

xDSL: Different DSL (Digital Subscriber Line) technologies

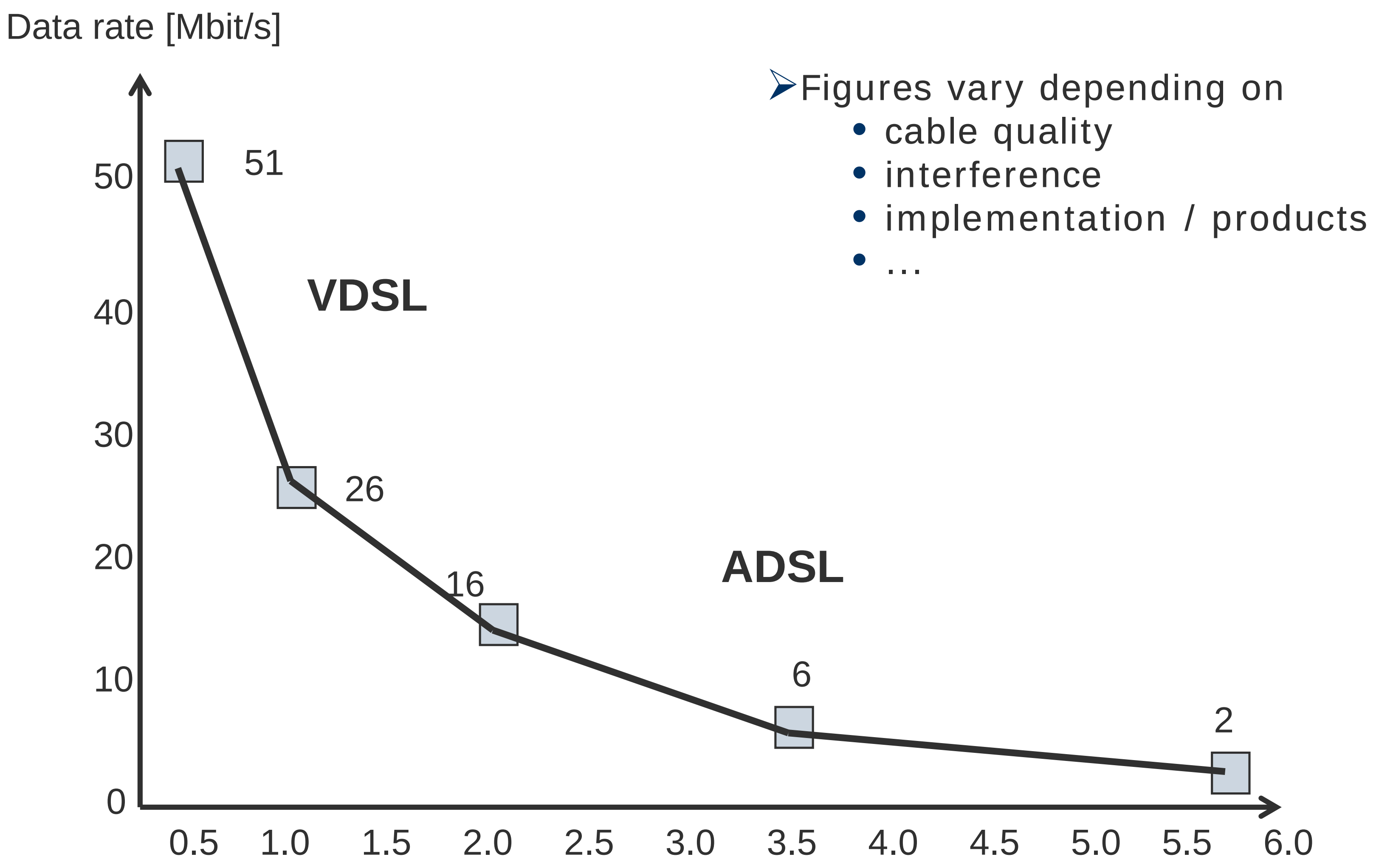
Goal: Use already existing phone lines (simple twisted pair, unshielded) for higher data rates

- Works around last mile problem
- Similar to ISDN: Replacement of analog phone system by a fully digital system (offering *n* 64 kbit/s channels)

Co-existence of classical analog (or digital ISDN) phone system plus high data rates possible – or full replacement of classical telephony by voice over IP



Typical Downstream Data Rates



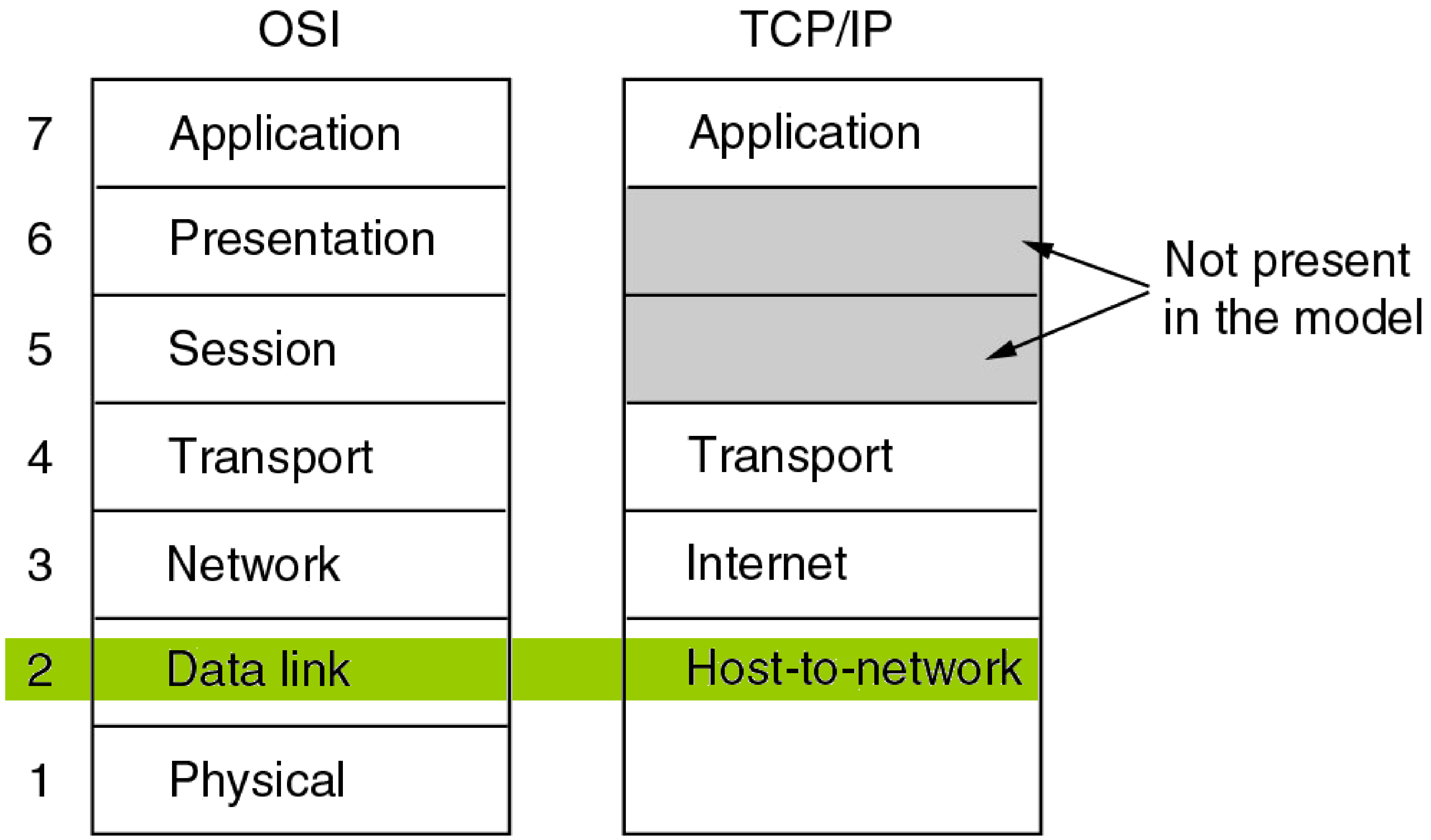
- ➢ Figures vary depending on
  - cable quality
  - interference
  - implementation / products
  - …

Source: www.dslforum.org

Questions & Tasks

- What are sources for errors in received bits?
- How to synchronize sender and receiver?
- How does the physical layer correct bits?
- Check the hidden slides for much more information about media types, baseband/broadband transmission, multiplexing etc.!
- Why can it be problematic to use 0-1-0 transitions in the bit stream for synchronization? What could be done without using e.g. Manchester coding?

Data Link Layer



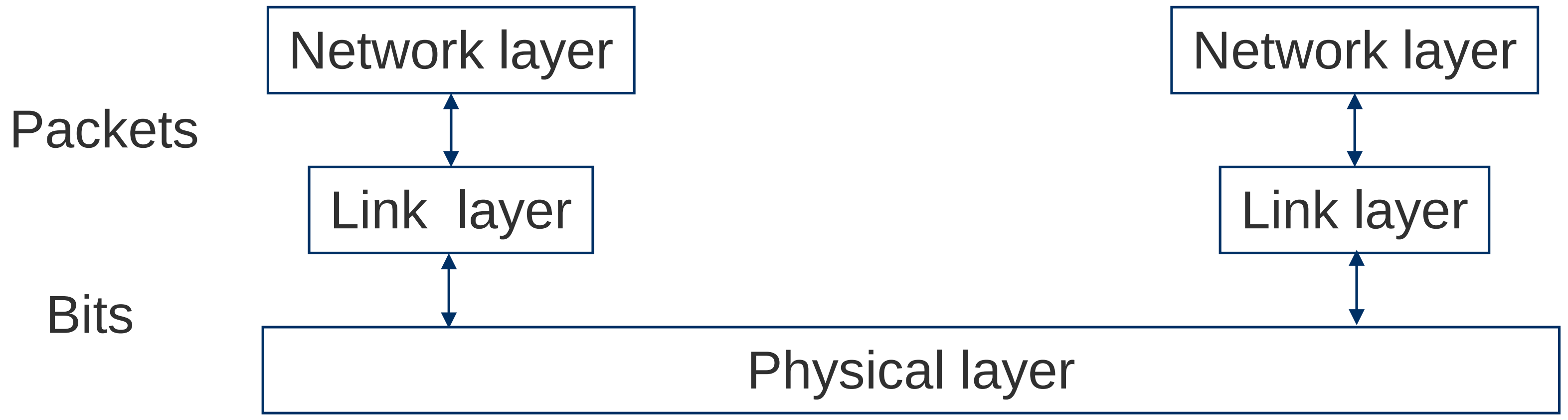
Setting for Data Link Layer

Link layer sits on top of physical layer

- Can thus use a bit stream transmission service
- But: Service might have incorrect bits

Expectations of the higher layer (networking layer)

- Wants to use either a packet service
    - Rarely, a bit stream service
- Does not really want to be bothered by errors
- Does not really want to care about issues at the other end



Services of Data Link Layer

Given setting and goals, the following services are required:

Transparent communication between two directly connected nodes

Framing of a physical bit stream into a structure of frames/packets
- Frames can be retransmitted, scheduled, ordered, ...

Error control
- Detection and correction

Connection setup and release
- Signaling and resource management on hosts

Acknowledgement-based protocols
- Make sure that a frame has been transmitted

Flow control
- Arrange for appropriate transmission speed between hosts

FRAMING

Link Layer Functions – Framing

How to turn bit stream abstraction (as provided by physical layer) into individual, well demarcated frames?

- Usually necessary to provide error control
  - Not obvious how to do that over a bit stream abstraction
- Frames and packets are really the same thing
  - Term “frames” used in link layer context by convention

Additionally: Fragmentation and reassembly

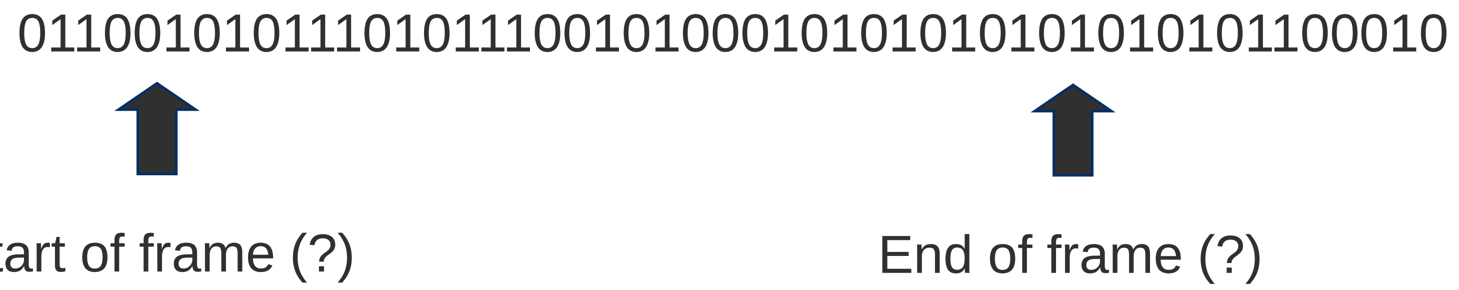
- If network layer packets are longer than link layer packets
- Commonly network layer set Maximum Transmission Unit (MTU) to a value that avoids link layer fragmentation
  - 1492 bytes on 802.3 (Ethernet)
  - 2272 bytes on 802.11 (WLAN)

Framing

How to turn a bit stream into a sequence of frames?

- More precisely: how does a receiver know when a frame starts and when it ends?

Delivered by physical layer:



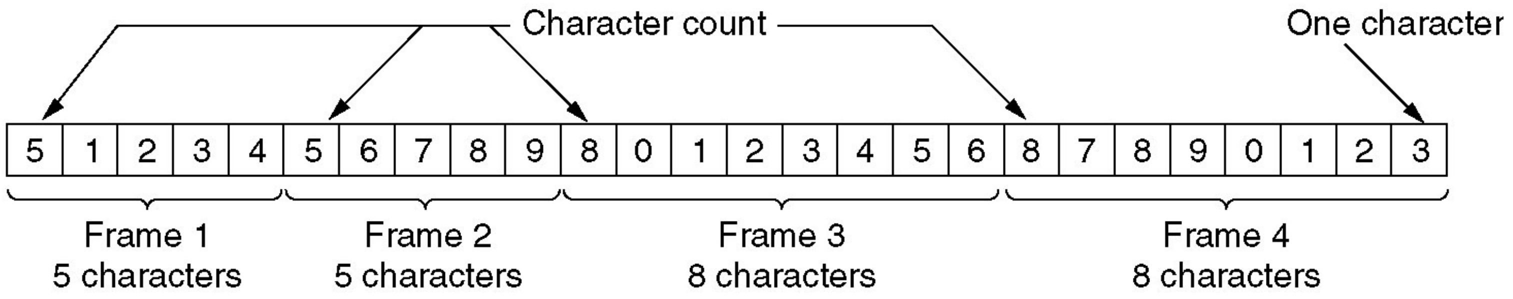
Note: Physical layer might try to detect and deliver bits when the sender is not actually transmitting anything

- Receiver tries to get any information from the physical medium

Framing by Character / Byte Count

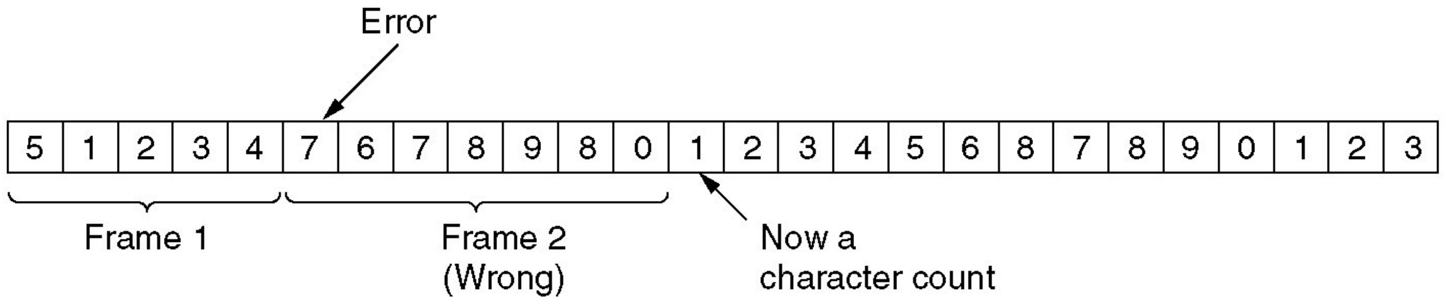
Idea: Announce number of bits (bytes, characters) in a frame to the receiver

- ➢Put this information at beginning of a frame – *frame header*



Problem: What happens if *count* information itself is damaged during transmission?

- Receiver will loose frame synchronization and produce different sequence of frames than original one

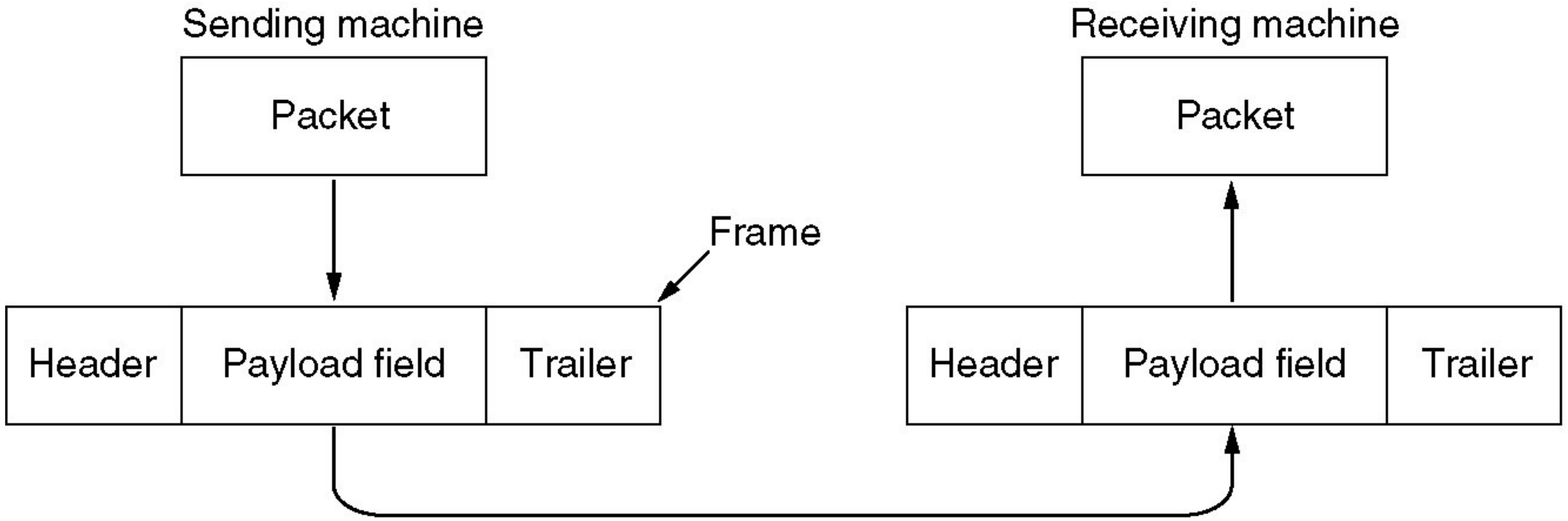


Basic Technique: Control Header

Albeit “character count” is not a good framing technique, it illustrates an important technique: *headers*

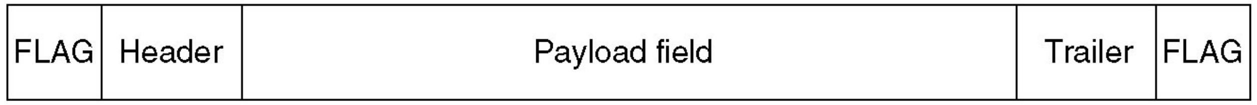
- If sender has to communicate administrative or control data to receiver, it can be added to actual packet content (“payload”)
- Usually at the start of the packet; sometimes at the end (“trailer”)
- Receiver uses headers to learn about sender’s intention

➢ Same principle applicable to all packet-switched communication



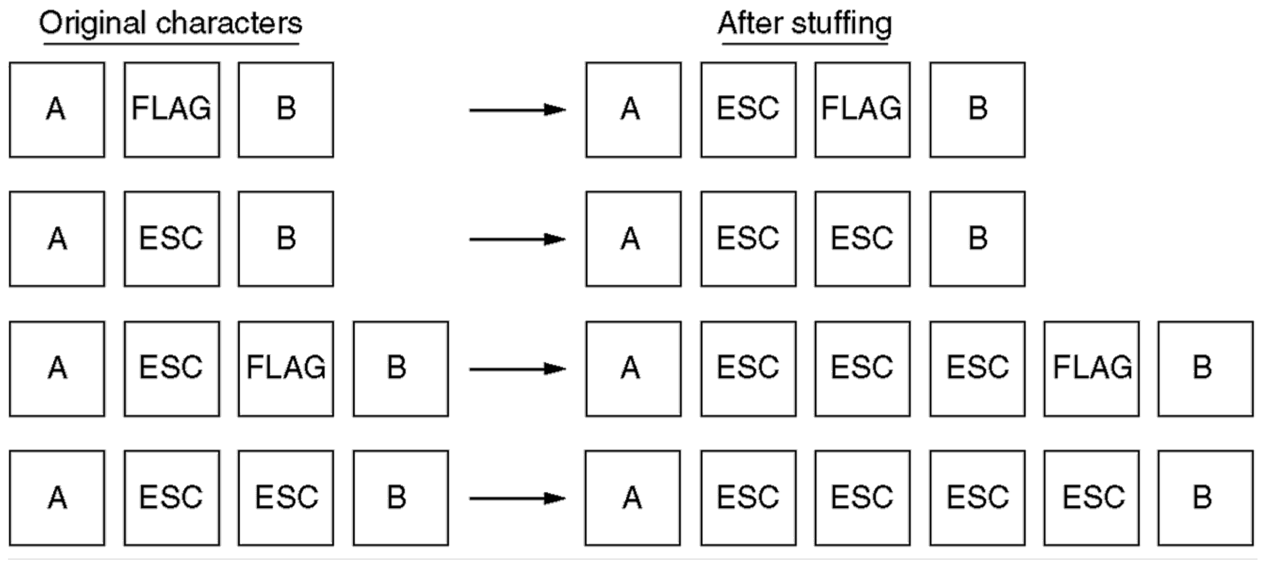
Framing by Flag Bytes / Byte Stuffing

Use dedicated flag bytes to demarcate start/stop of frame



➢What happens if the flag byte appears in the payload?

- Escape it with a special control character – *byte stuffing*
- If that appears, escape it as well



Framing by Flag Bit Patterns / Bit Stuffing

Byte stuffing is closely tied to characters/bytes as fundamental unit – often not appropriate

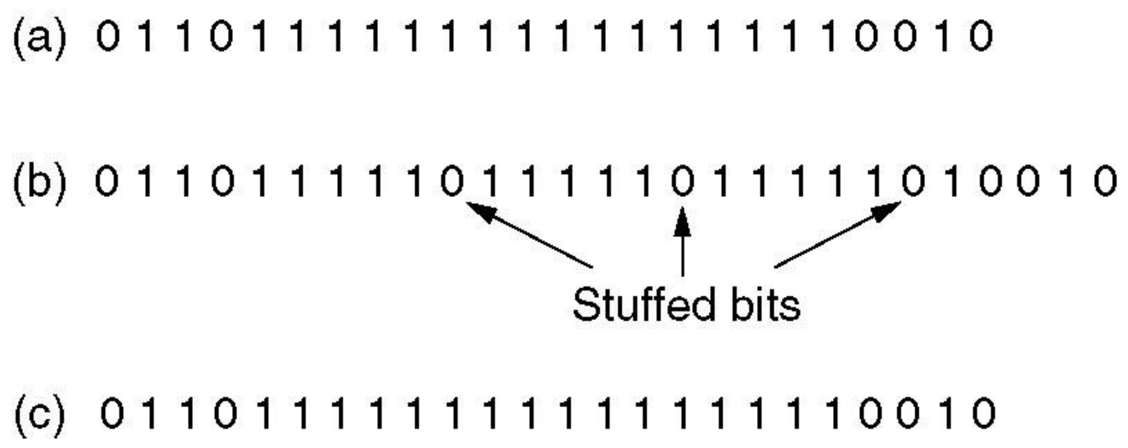
➢Use same idea, but stick with the bit stream abstraction of the physical layer

- Use bit pattern instead of flag byte – often, 01111110
    - Actually, it IS a flag byte
- Bit stuffing process:
    - Whenever sender sends five 1s in a row, it automatically adds a 0 into bit stream – except in flag pattern
    - Receiver throws away (“destuffs”) any 0 after five 1s

Original payload

After bit stuffing

After destuffing


(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0
Stuffed bits
(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Framing by Coding Violations

Suppose the physical layer's encoding rules "bits ! signals" still provide some options to play with

- Not all possible combinations that physical layer can represent are used to represent bit patterns

➢Example: Manchester encoding: only low/high and high/low is used

When "violating" these encoding rules, data can be transmitted, e.g., start and end of frame

➢Example: Manchester – use high/high or low/low

  - This drops self-clocking feature of Manchester, but clock synchronization is sufficiently good to hold for a short while

Powerful and simple scheme, e.g. used by Ethernet networks

- But raises questions regarding bandwidth efficiency as coding is *obviously* not optimal

Questions & Tasks

- What is the basic service of the Data Link Layer?
- Why framing and how to do it?
- Given a fixed data rate on the medium – how does bit stuffing influence the available user data rate at the link layer?

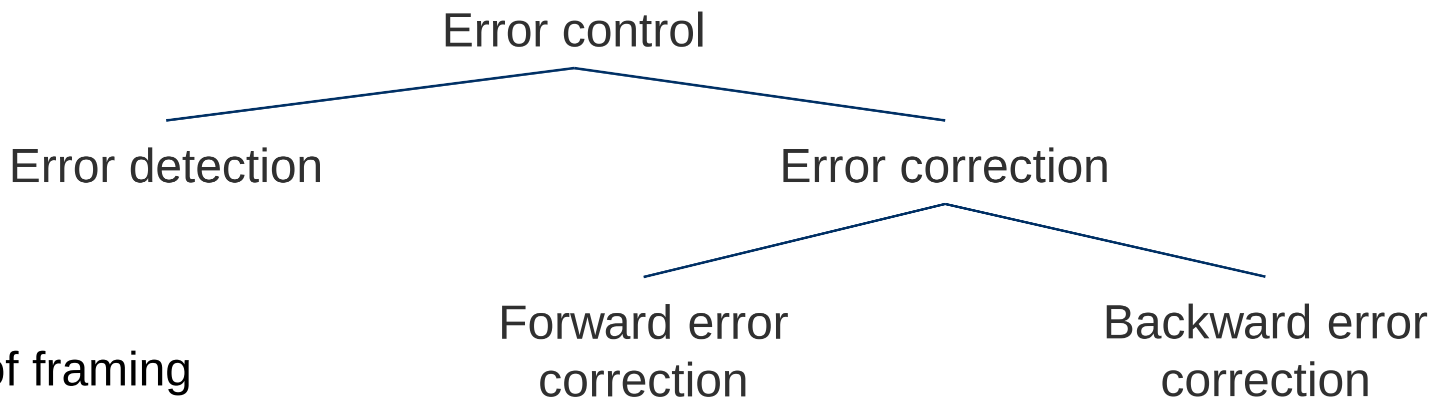
ERROR CONTROL

Link Layer Functions – Error Control

Error detection – Check for incorrect bits

Error correction – Correct erroneous bits

- Forward error correction (FEC) – invest effort *before* error happened; avoid delays in dealing with it
  - ➢Redundancy / overhead
- Backward error correction – invest effort *after* error happened; try to repair it ➔ ARQ (Automatic Repeat reQuest)
  - ➢Delays



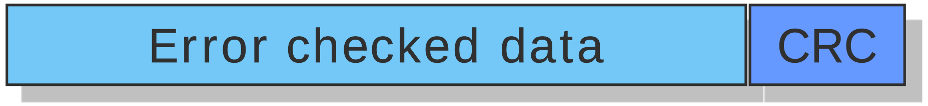
➢Usually build on top of framing

Error Detection: Cyclic Redundancy Check (CRC)

CRC can check arbitrary, unstructured sequence of bits

A check sequence (CRC code) is appended to checked data

- Typically calculated by a feedback shift register in hardware



When calculating the CRC code a generator polynomial is used which is known to both sender and receiver

Calculation of CRC code

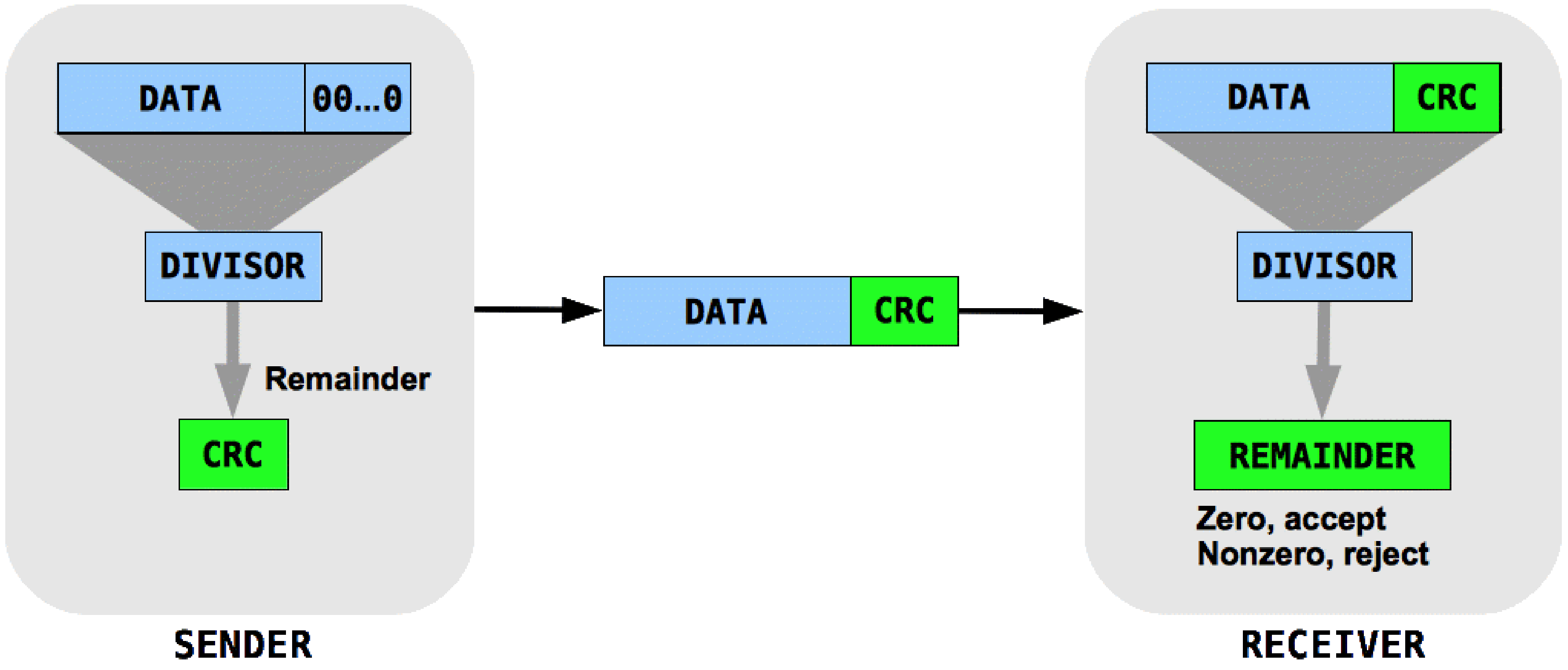
1. View bit sequence as polynomial with binary coefficients:
   - **100010** is viewed as **1**$*x^5$+**0**$*x^4$+**0**$*x^3$+**0**$*x^2$+**1**$*x^1$+**0**$*x^0$
2. Expand polynomial with n 0s, n is the degree of the generator polynomial
3. Divide expanded bit sequence (i.e. polynomial) by generator polynomial

➢ CRC code is the remainder of the division, result is discarded

4. Receiver again divides received bit sequence (including the CRC code) by generator polynomial

➢ If no error occurred the remainder is 0

Illustration of CRC



CRC Example (Sender)

Transmitted payload: 110011

Generator polynomial: $x^4 + x^3 + 1$

- ➢Translates into sequence of coefficients: 11001
- - Addition or subtraction equal simple bitwise XOR
  - ➢Special arithmetic for polynomials modulo 2

Length of CRC = degree of generator polynomial = 4

Calculation of CRC:

```
1100110000 ÷ 11001 = 100001 (mod 2)
11001
0000010000
     11001
      1001  = remainder
```

➢Transmitted bit sequence: 1100111001

CRC Example (Receiver)

Reception of a correct bit sequence:

```
        1100111001 ÷ 11001 = 100001 (mod 2)
        11001
        0000011001
             11001
             00000  = remainder
```

➢No remainder, thus the received bits *should* be error free

Reception of a erroneous bit sequence:

```
        1111111000 ÷ 11001 = 101001 (mod 2)
        11001
        0011011
          11001
          00010000
             11001
             01001  = remainder ≠ 0
```

➢There is a remainder unequal 0, thus there was *definitely* a transmission error

Properties of CRC

CRC can detect the following errors:

- All single bit errors
- All double bit errors (if $(x^k + 1)$ is not divisible by generator polynomial for $k \leq$ frame length)
- All errors affecting an odd number of bits (if $(x+1)$ is a factor of the generator polynomial)
- All error bursts of length $\leq$ degree of generator polynomial

Internationally standardized generator polynomials:

- CRC-12 $= x^{12} + x^{11} + x^3 + x^2 + x + 1$
- CRC-16 $= x^{16} + x^{15} + x^2 + 1$
- CRC-CCITT $= x^{16} + x^{12} + x^5 + 1$

CRC-16 and CRC-CCITT detect

- All single and double errors
- All errors affecting an odd number of bits
- All error bursts of length ≤ 16
- 99,997% of all error bursts of length 17
- 99,998% of all error bursts of length ≥ 18

FEC Example (Sender)

CRC uses redundancy to *detect* errors

FEC uses redundancy to *correct* errors

Simple FEC scheme: Repeat data several times, then use majority decision

- ➢ Problematic with regard to overhead

More advanced: XOR, Reed-Solomon, …

Example (XOR):

- To send the following three packets:

	0101 - P1
	1111 - P2
	0000 - P3

1. Calculate a fourth packet using XOR:

   1010 - P4

2. Send all four packets to the receiver
   - ➢ Overhead: (4 – 3) / 4 = 25%

FEC Example (Receiver)

➢For the receiver, it is sufficient to receive any three out of the four packets.

Reconstruct missing packet based on received packets:

- P1 is lost:

```
1111
0000
1010
0101 -> P1
```

- P2 is lost:

```
0101
0000
1010
1111 -> P2
```

- P3 is lost:

```
0101
1111
1010
0000 -> P3
```

However, receiver still has to know which packet was lost

➢Requires packet numbering

Questions & Tasks

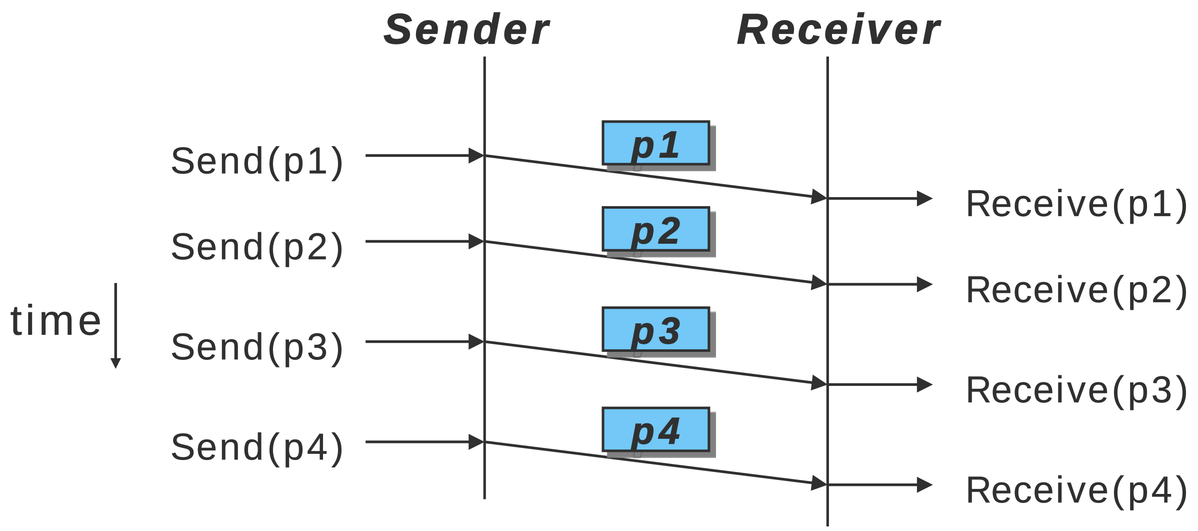
- Can we correct all errors? Can we detect all errors? Can we correct all errors that we detected?
- What do we know, if the data link errors does not find an error?

FLOW CONTROL

Link Layer Functions – Flow Control

Assumptions in an ideal world:

- Sender/receiver are always ready to send/receive
- Receiver can handle amount of incoming data
- No errors occur that cannot be handled by FEC



➢What happens if packets are lost?

➢What happens if the sender floods the receiver?

- Imagine a web server sending data to a mobile phone…

Very Simple Solution: Stop-and-Wait

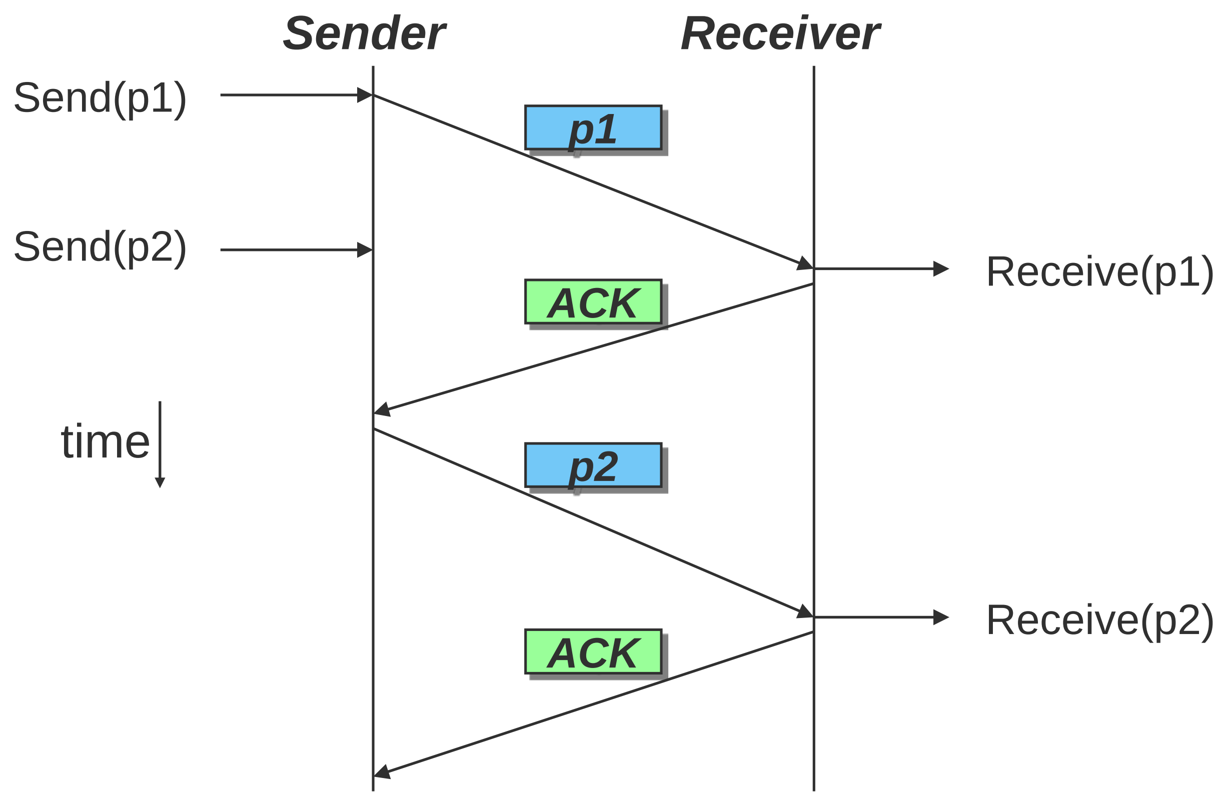
Concentrate on one single packet

Receiver acknowledges correct reception of the packet

Sender has to wait for that acknowledgement before continuing with next packet

No overloading of the receiver possible!

➢ Basic flow control

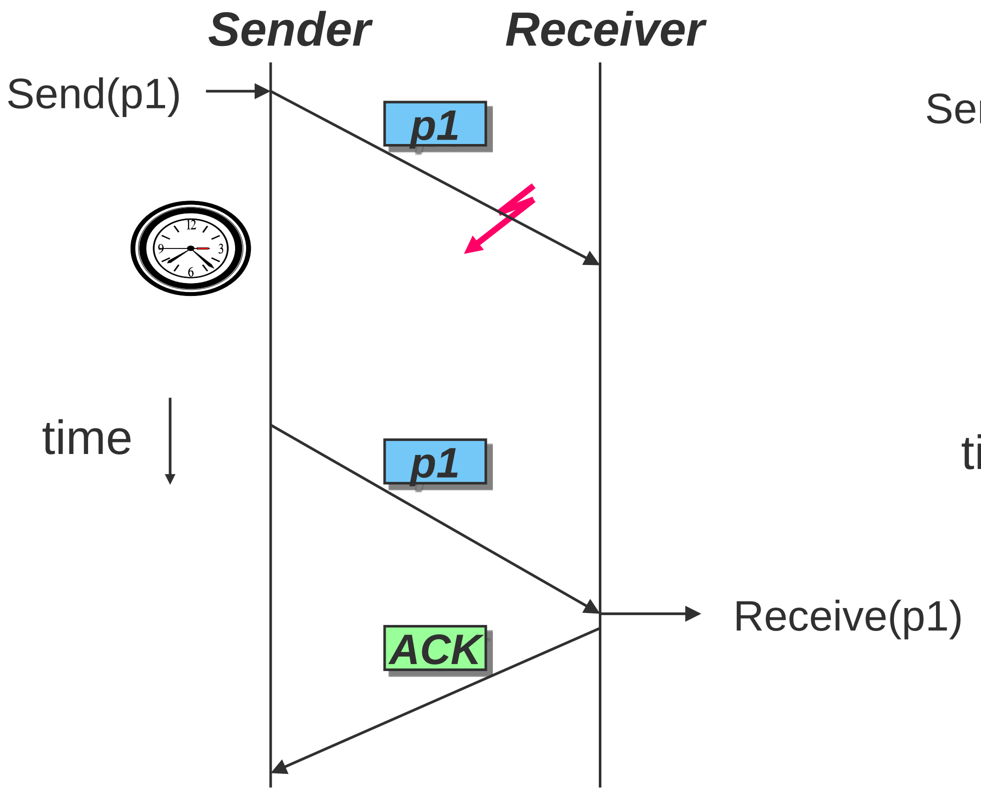


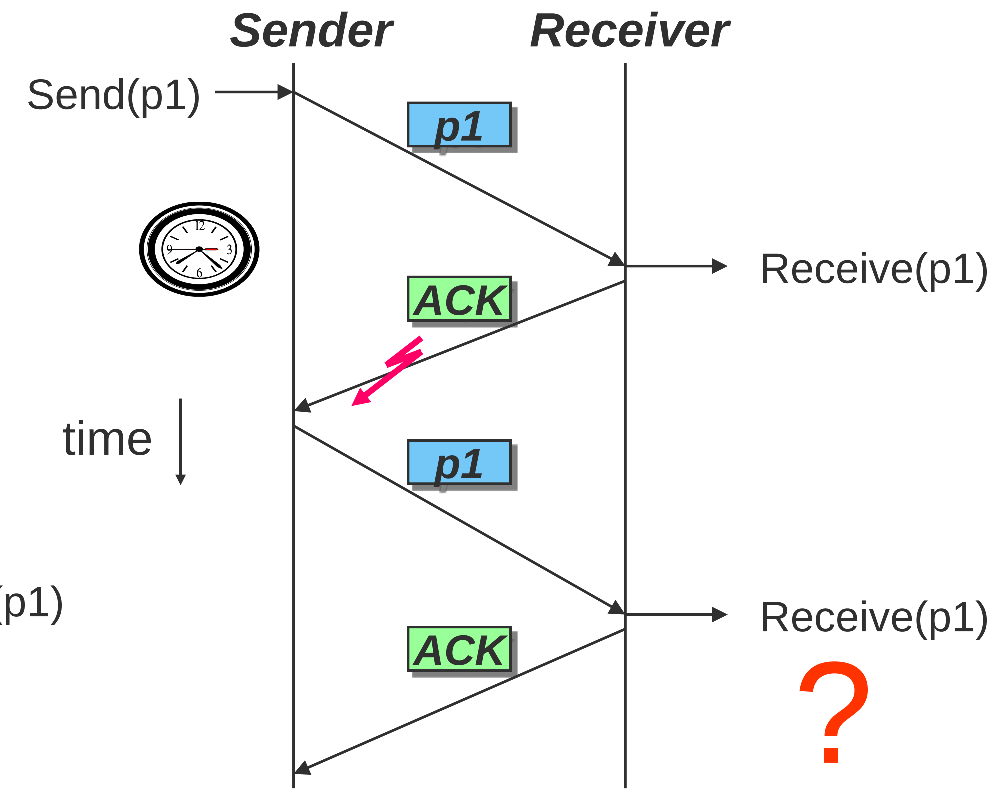
Problems of Stop-and-Wait

What happens if errors occur?

- Lost packet? Lost acknowledgement? Is there a difference?

Basic solution: ARQ (Automatic Repeat reQuest)





Problem of Stop-and-Wait ARQ

Sender cannot distinguish between lost packet and lost acknowledgement ➔ Has to re-send the packet

Receiver cannot distinguish between new packet and redundant copy of old packet ➔ Additional information is needed

➢Put a *sequence number* in each packet, telling the receiver which packet it is
- Sequence numbers as *header information* in each packet
- Simplest sequence number: 0 or 1

Needed in packet and acknowledgement
- One convention: In ACK, send sequence number of last correctly received packet
- Also possible: Send sequence number of next expected packet

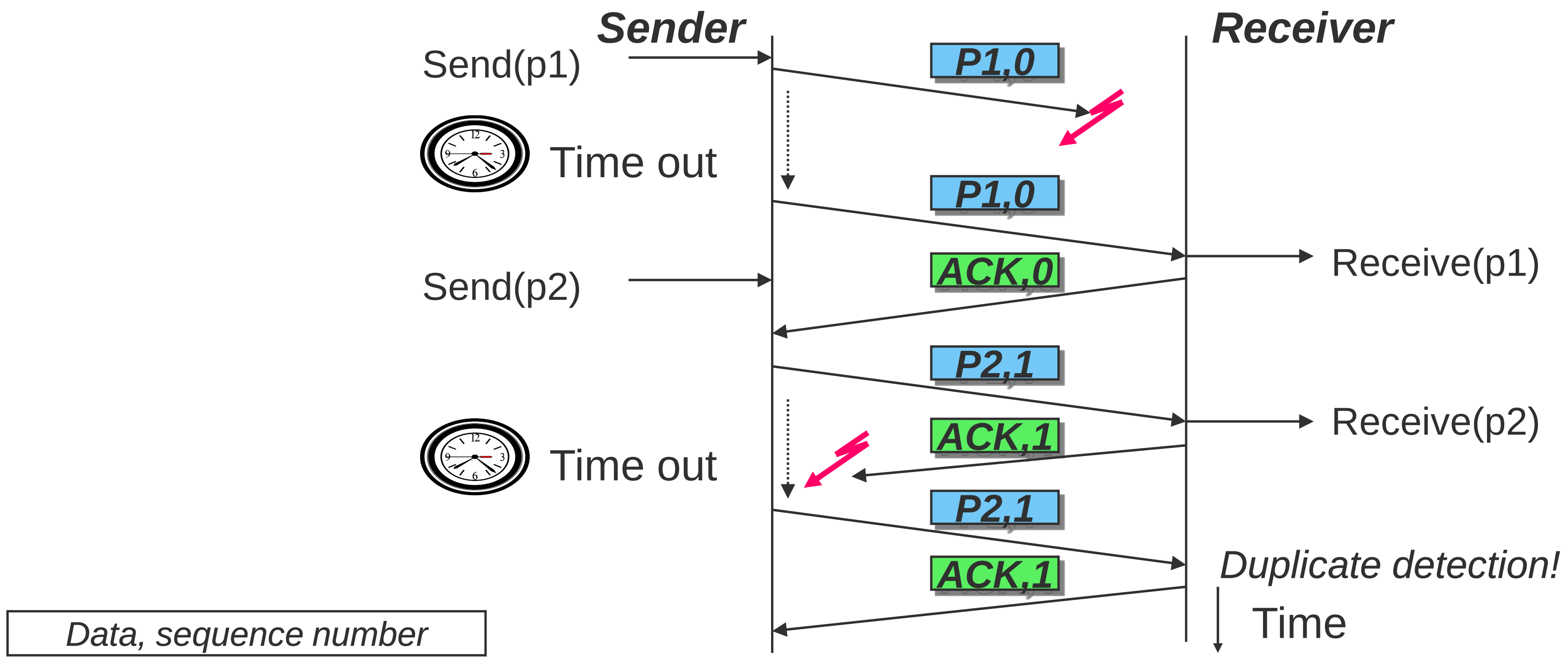
  ➢Be aware: Some protocols count bytes instead of packets

Alternating Bit Protocol

Simple, but reliable protocol over noisy channels

Uses 0 and 1 as sequence numbers, ARQ for retransmission

Simple form of flow control (here combined with error control, other protocols separate these functions)



Handling Multiple Outstanding Packets

Introduce a larger sequence number space

- ➢E.g., *n* bits or 2^n sequence numbers

Not all of them may be allowed to be used simultaneously

- - Recall alternating bit case: 2 sequence numbers, but only 1 may be “in transit”

Use *sliding windows* at both sender and receiver to handle these numbers

- - Sender: *sending window* – set of sequence numbers it is allowed to send at given time
- - Receiver: *receiving window* – set of sequence numbers it is allowed to accept at given time

➢Window size corresponds to flow control

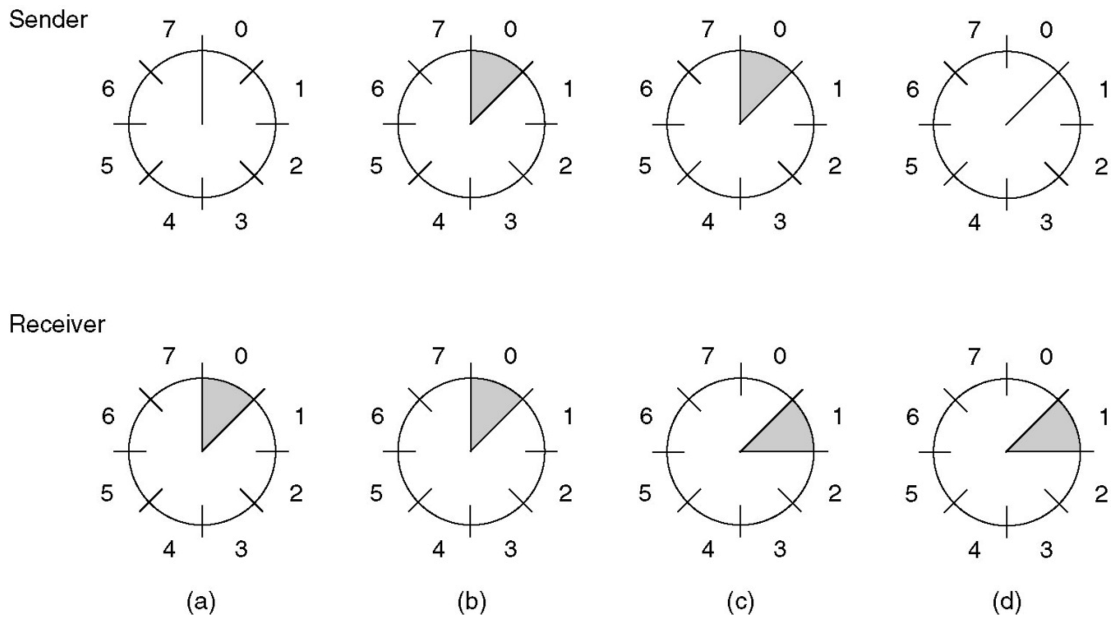
- - May be fixed in size or adapt dynamically over time

Simple Example: Sliding Window

Simple sliding window example for *n*=3, window size fixed to 1

Sender tracks currently unacknowledged sequence numbers

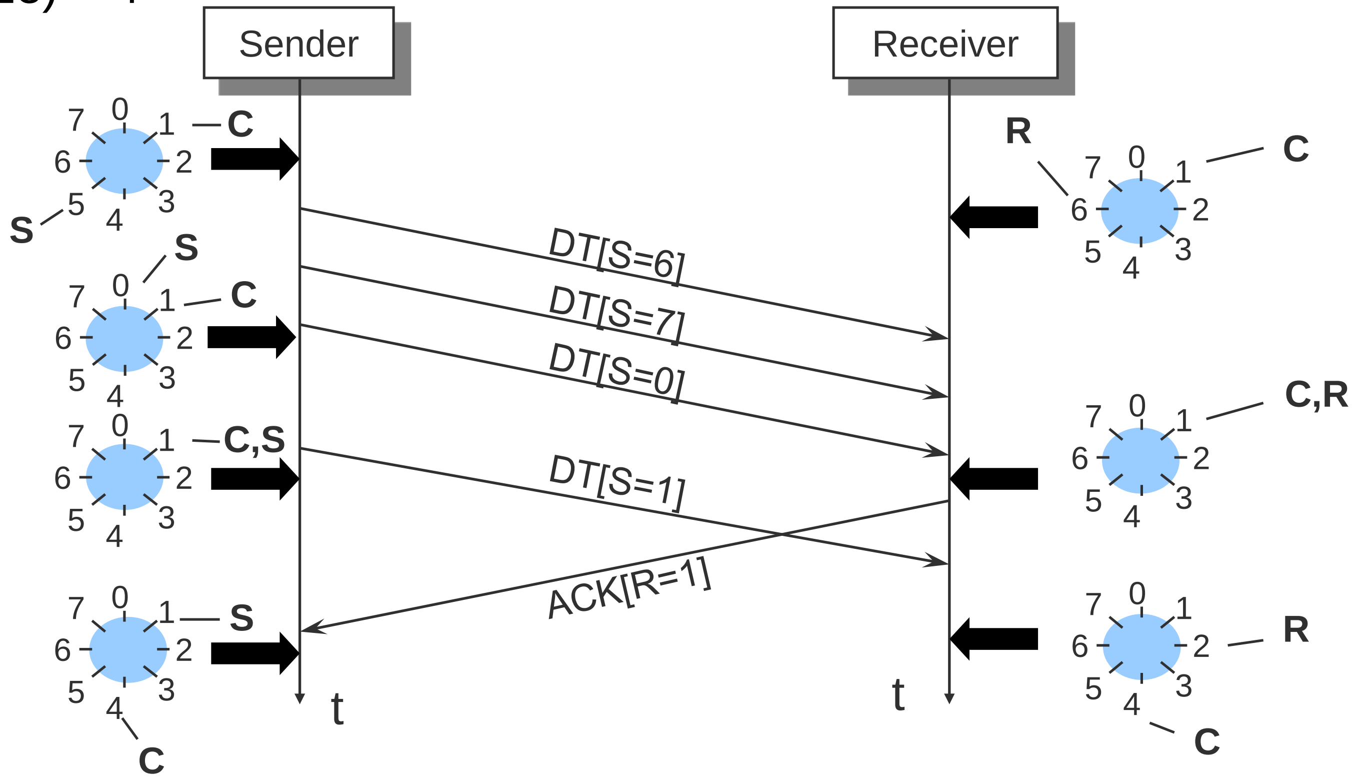
- If maximum number of unacknowledged frames is known, this is equivalent to sending window as defined on previous slide



a) Initially, before any frame is sent
b) After first frame is sent with sequence number 0
c) After first frame has been received
d) After first acknowledgement has arrived

Advanced Example: Sliding Window

Sender credit (window size) = 4



S: Sequence number (last sent packet)
R: Next expected sequence number = Acknowledges packets up to R-1
C: Upper window limit (current max. sequence number)

➢Disadvantage: Coupling of flow control and error control

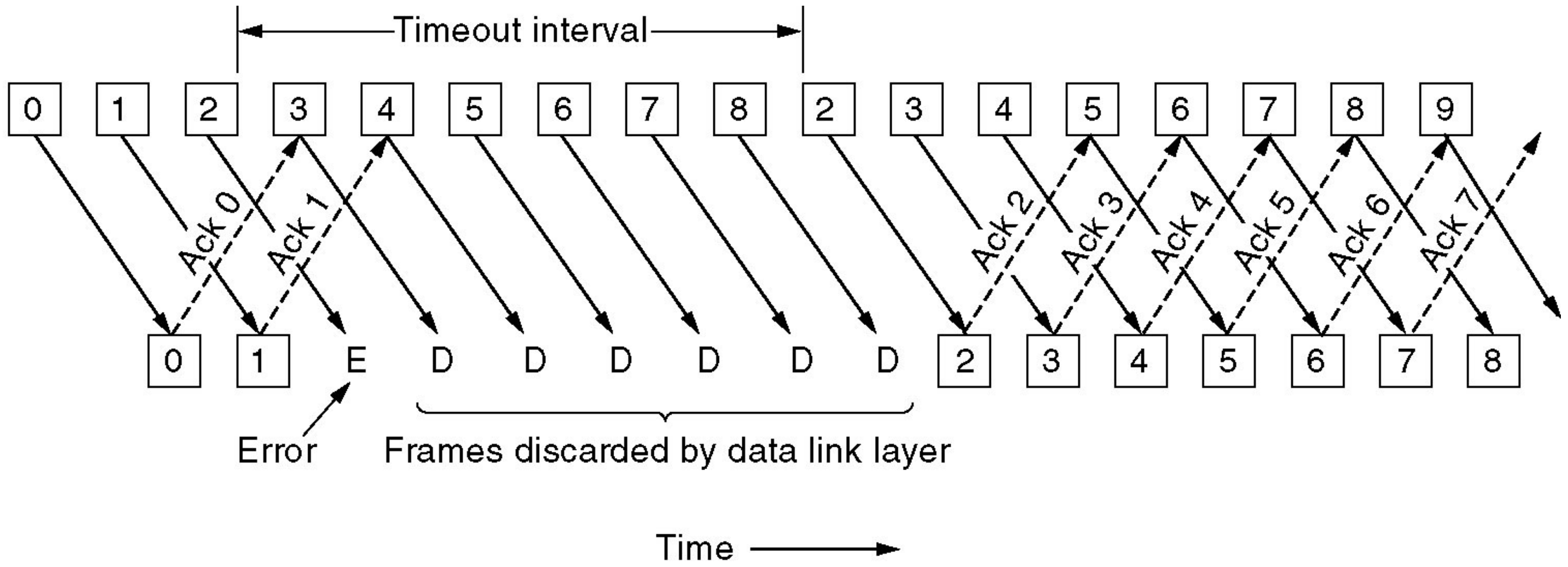
Transmission Errors and Receiver Window Size

Assumption:

- Link layer should deliver all frames correctly and in sequence
- Sender is pipelining packets to increase efficiency

What happens if packets are lost (discarded by CRC)?

With receiver window size 1, all following packets are discarded as well!



Go-back-N

With receiver window size 1, all frames following a lost frame cannot be handled by receiver

- ➢They are out of sequence
- ➢They cannot be acknowledged, only ACKs for the last correctly received packet can be sent

Sender will timeout eventually

- All frames sent in the meantime have to be repeated
- ➢Go-back-N (frames)

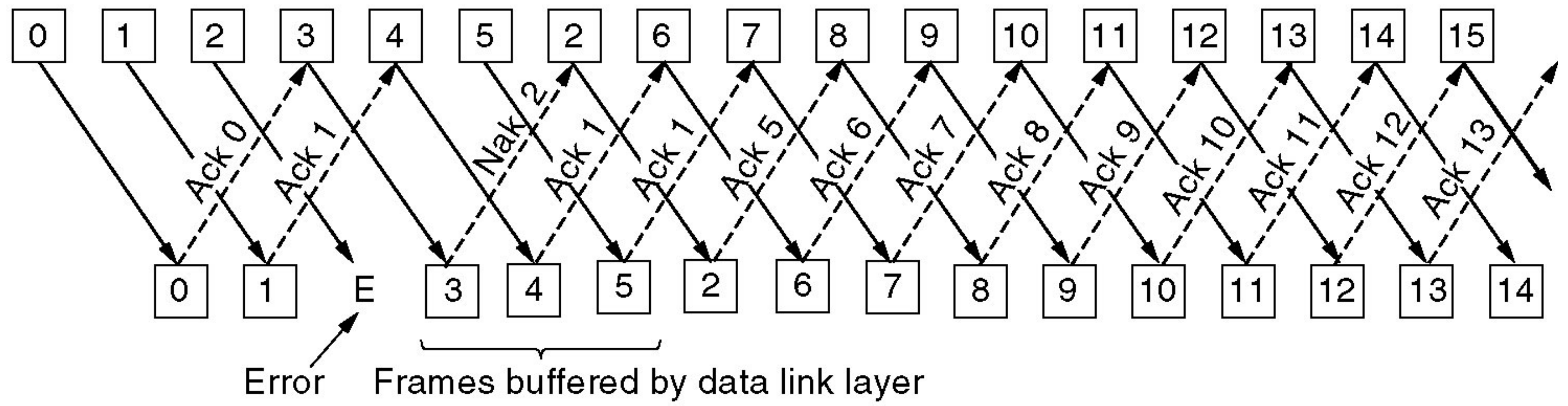
Quite wasteful of transmission resources

But saves resources at receiver

Selective Repeat

Suppose we invest into a receiver that can buffer packets intermittently if some packets are missing
- ➢Corresponds to receiver window larger than 1

Resulting behavior:



- Receiver explicitly informs sender about missing packets using *Negative Acknowledgements* (NACKs)
- Sender selectively repeats the missing frames
- Once missing frames arrive, they are all passed to network layer

➢More resources used at receiver, less overhead in case of error

Duplex Operation and Piggybacking

So far, simplex operation at the (upper) service interface was assumed

- Receiver only sent back acknowledgements, possibly using duplex operation of the lower layer service

➢What happens when the upper service interface should support full-duplex operation?

- Use two separate channels for each direction (SDMA)
  - ➢Wasteful on bandwidth/resources
- Interleave ACKs and data frames in a given direction (TDMA)
  - ➢Better, but still some overhead
- *Piggybacking*: Put ACKs from A to B into data frames from B to A (as part of B’s header to A)
  - ➢Minimal overhead
  - ➢We’ll see this principle again on layer 4 with TCP!

Conclusion

Most problems in the link layer are due to errors:

- Errors in synchronization require non-trivial framing functions
- Errors in transmission require mechanisms to
  - Correct them so as to hide from higher layers
  - Detect them and repair them afterwards

Flow control is often tightly integrated with error control in commonly used protocols

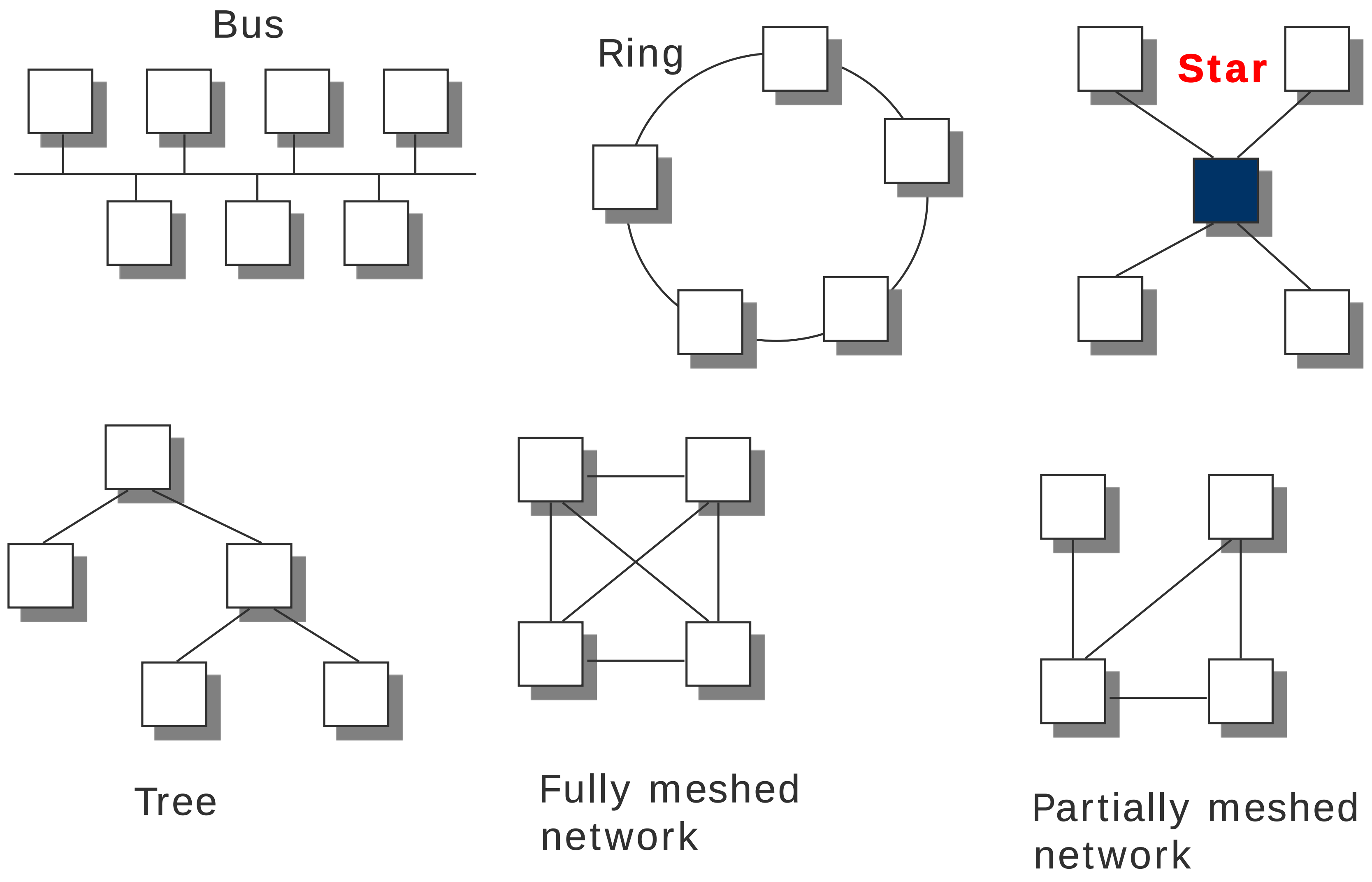
➢It *is* a separate function and can be implemented separately

Connection setup/teardown still has to be addressed

➢Necessary to initialize a joint context for sender and receiver

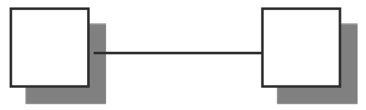
NETWORK TOPOLOGIES

Topologies by Network Structure

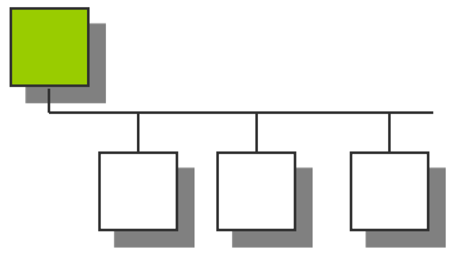


Topologies by Connectivity

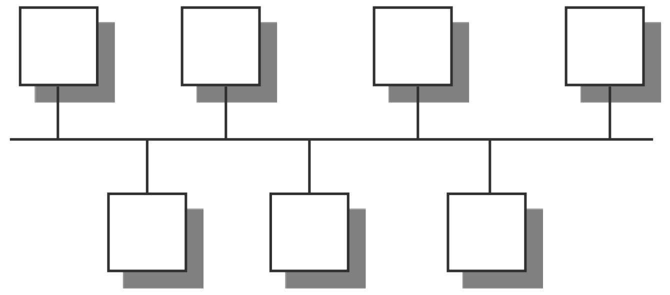
Point-to-point connection



Point-to-multi-point connection (asymmetrical)

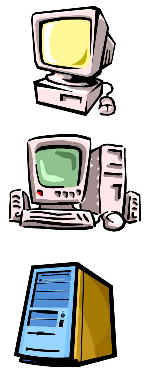


Multi-point connection (symmetrical)



➢Distinguish: Physical (layer 1) and logical (≥ layer 2) topologies

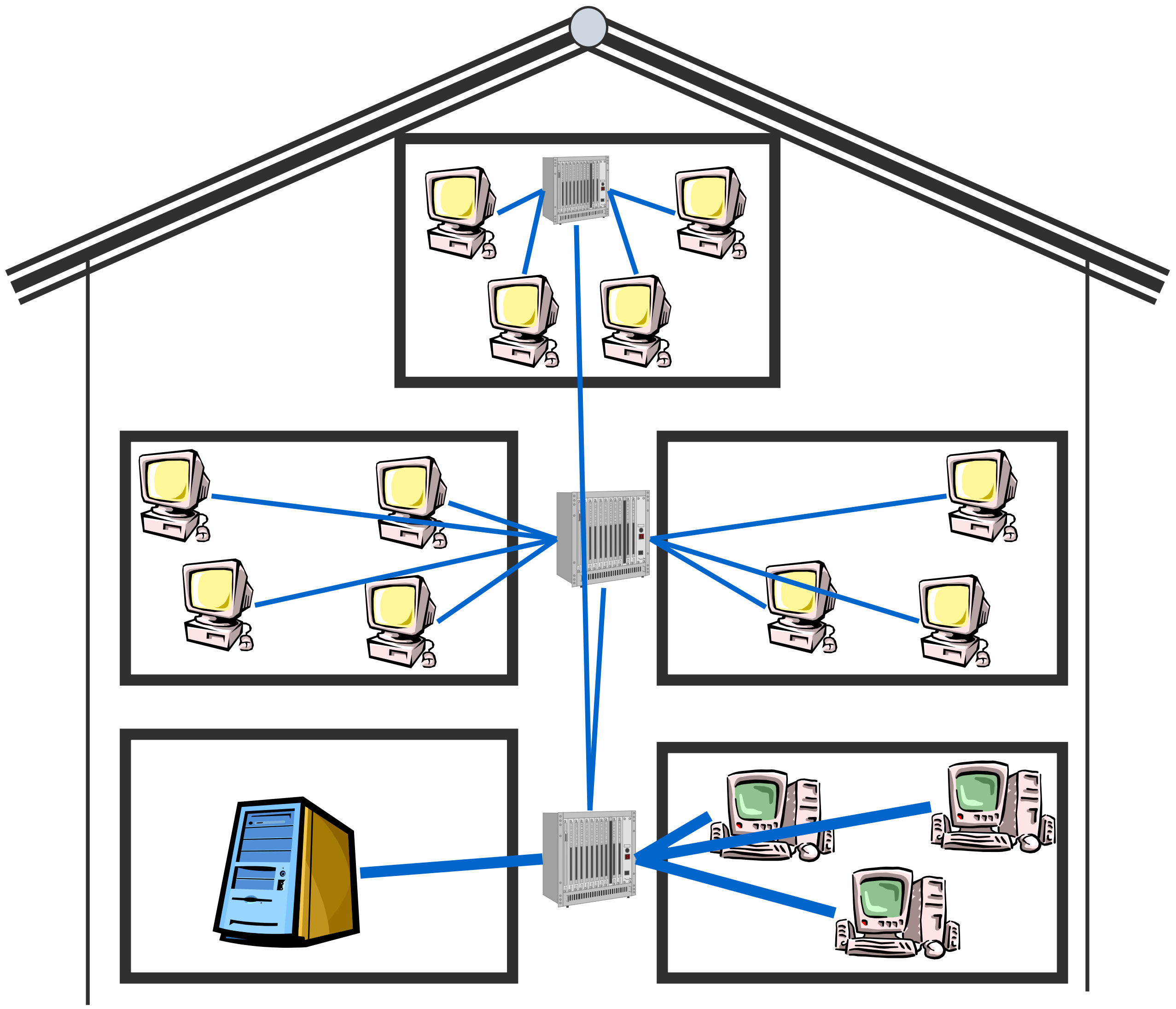
Structured Cabling



Star on each floor
Star to connect individual stars

➢Tree-like structure

➢Point-to-point connections avoid packet collisions



Star Topology – Lots of Cables

Cabling not trivial:

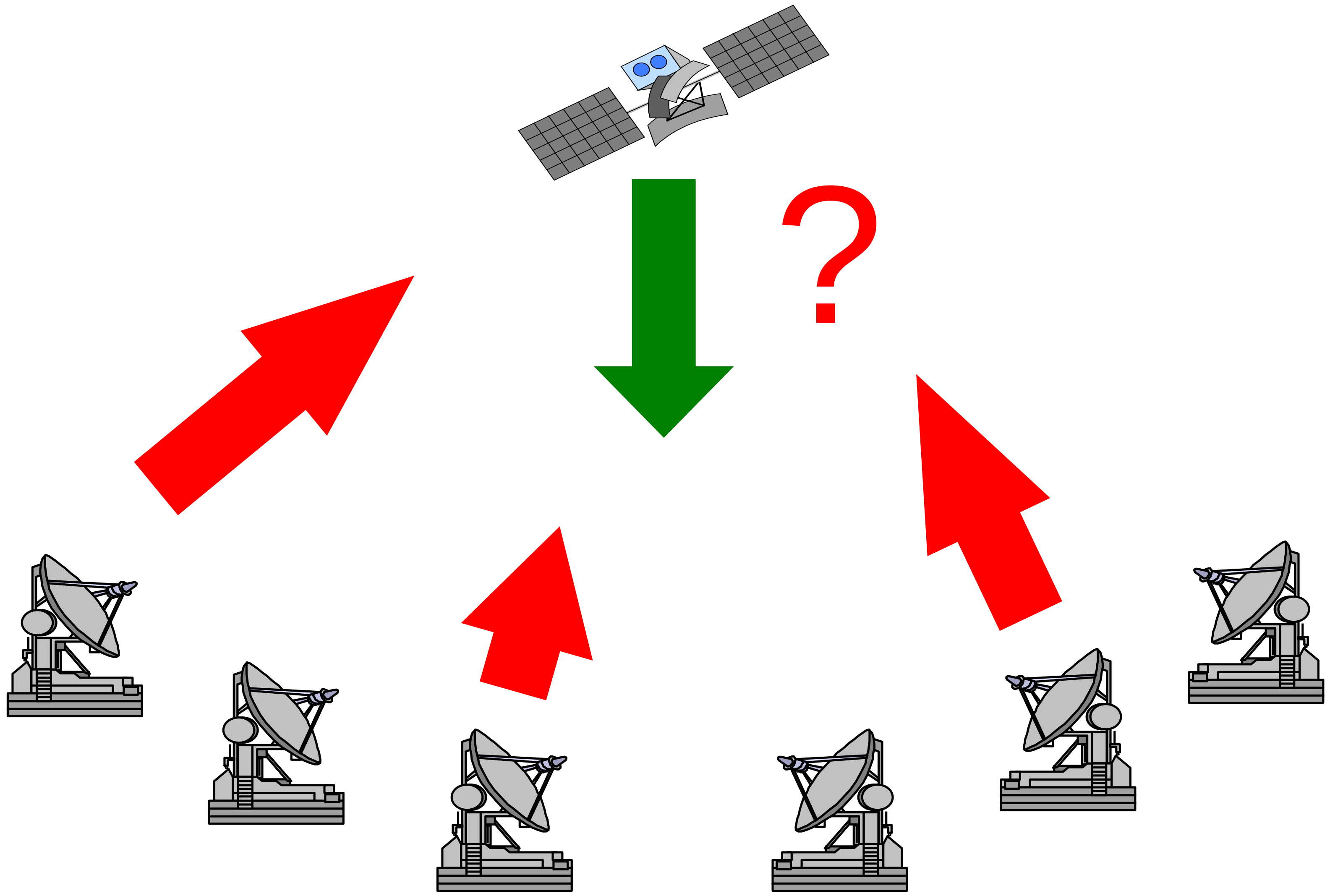
- Initial cost, upgradability, space for installation

Questions & Tasks

- Why do we need flow control at all?
- What are disadvantages of the simple stop-and-wait?
- If you are interested in advanced flow control schemes have a look at the hidden slides!
- A star topology requires a lot of cabling – why is it still attractive?
- Give examples for topologies and where you can find them!

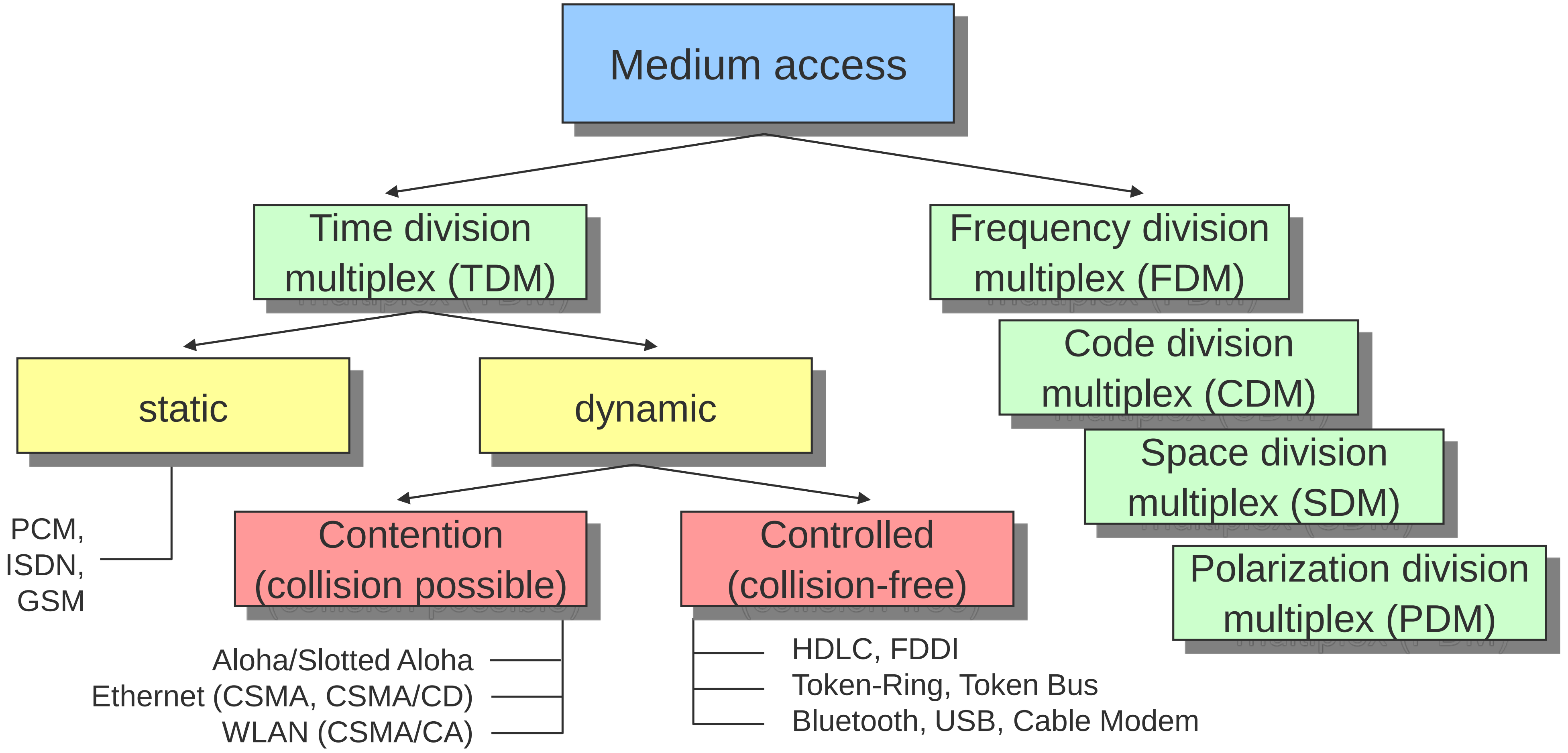
MEDIUM ACCESS CONTROL

Motivation: Satellite Medium Access



Approaches to Medium Access

Several stations want to access the same medium
– how to separate stations?



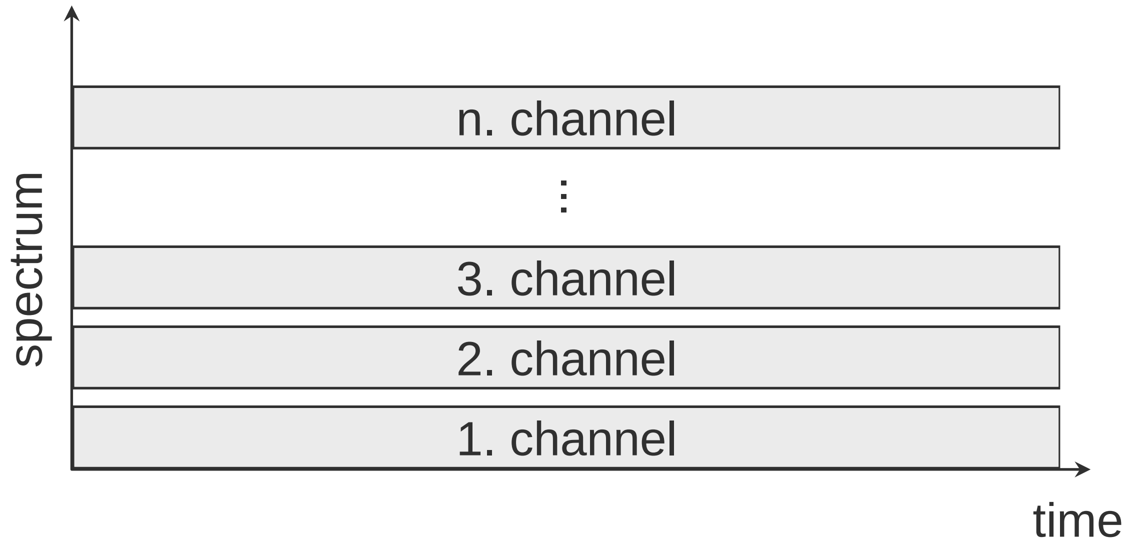
Medium access
Time division multiplex (TDM)
Frequency division multiplex (FDM)
Code division multiplex (CDM)
Space division multiplex (SDM)
Polarization division multiplex (PDM)
static
dynamic
PCM, ISDN, GSM
Contention (collision possible)
Controlled (collision-free)
Aloha/Slotted Aloha
Ethernet (CSMA, CSMA/CD)
WLAN (CSMA/CA)
HDLC, FDDI
Token-Ring, Token Bus
Bluetooth, USB, Cable Modem

FDM + Algorithm for Access: FDMA

Frequency Division Multiple Access (FDMA):

- Subdivide spectrum into sub-channels
    - Use different broadband frequencies for modulation
- Assign sub-channel to user
    - Static vs. dynamic assignment

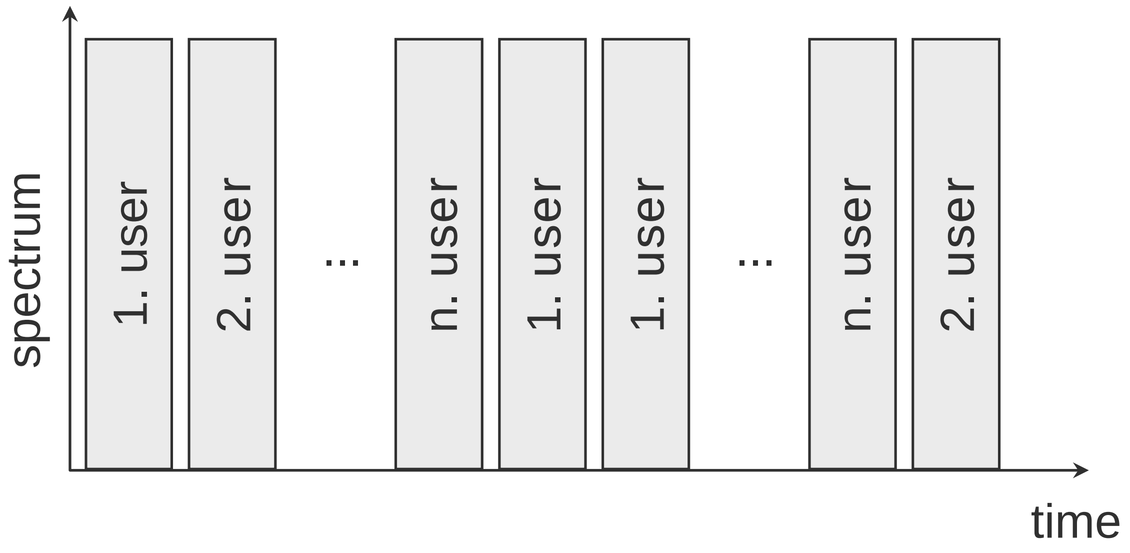
➢Examples: Radio stations, cable channels, WLAN APs



TDM + Algorithm for Access: TDMA

Time Division Multiple Access (TDMA):

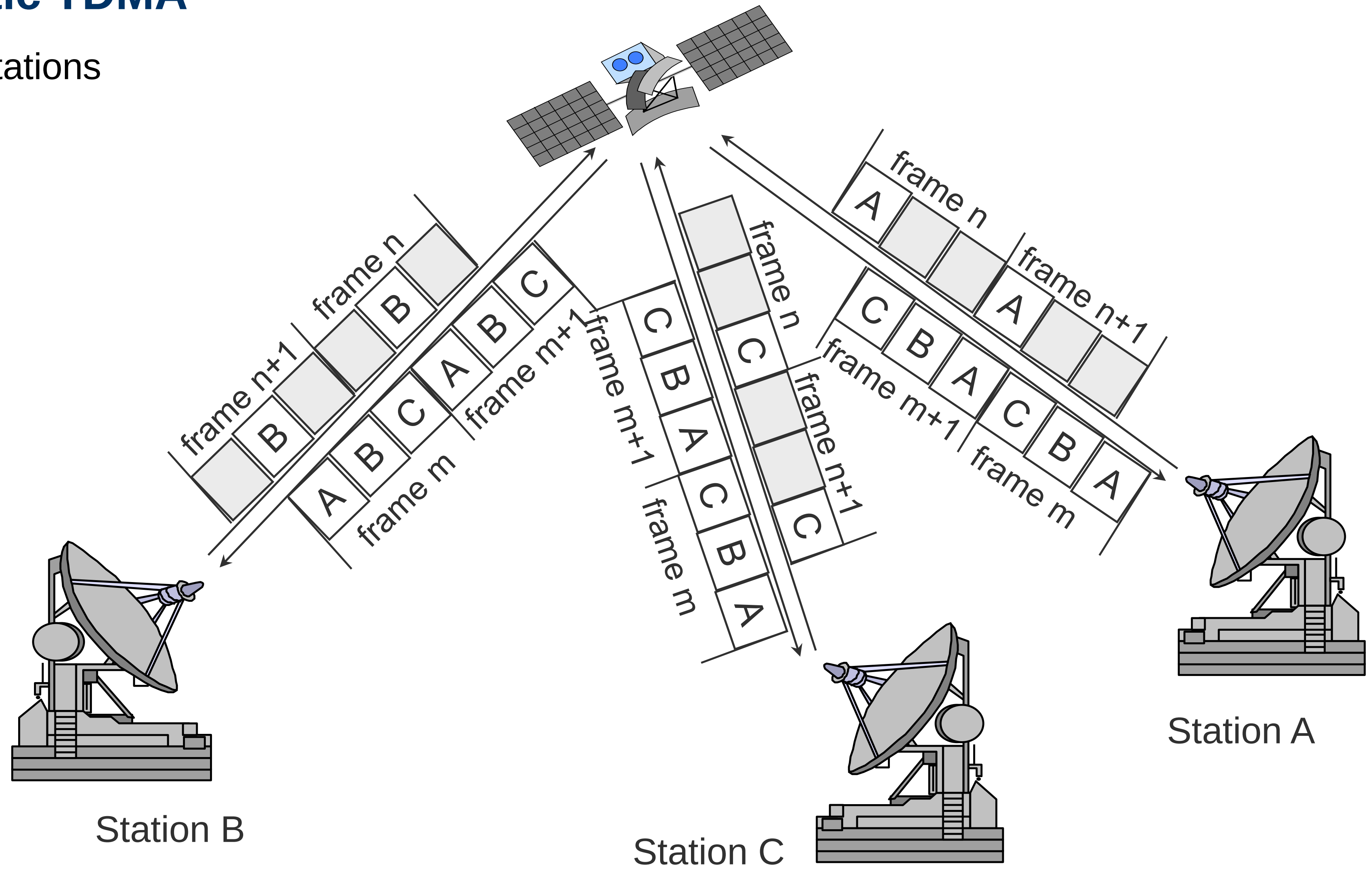
- Assign channel (static/dynamic) for certain time to user
    - Each user can use the full channel for the assigned time
- Static: Cyclic assignment (e.g. GSM, ISDN, Bluetooth voice)
- Dynamic: On-demand assignment (e.g. Ethernet, WLAN hosts, Bluetooth data)



Example: Static TDMA

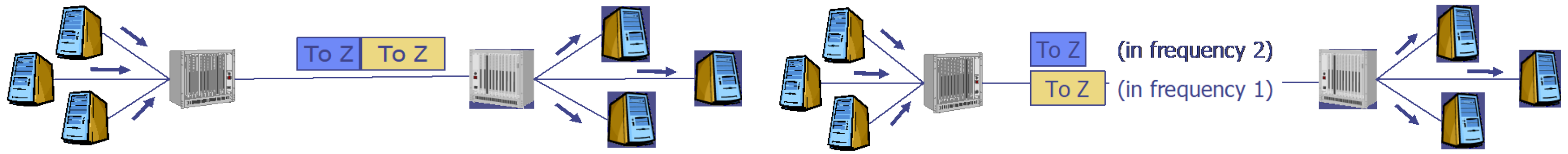
Fixed number of stations

Fixed frame size



Static Multiplexing – General Idea

A single resource can be statically multiplexed by
- assigning fixed time slots to multiple communication pairs
- assigning fixed frequency bands
- …



Assigning fixed resources to different sources is fine if
- data rate of source and multiplexed link are matched
- sources can always saturate the medium

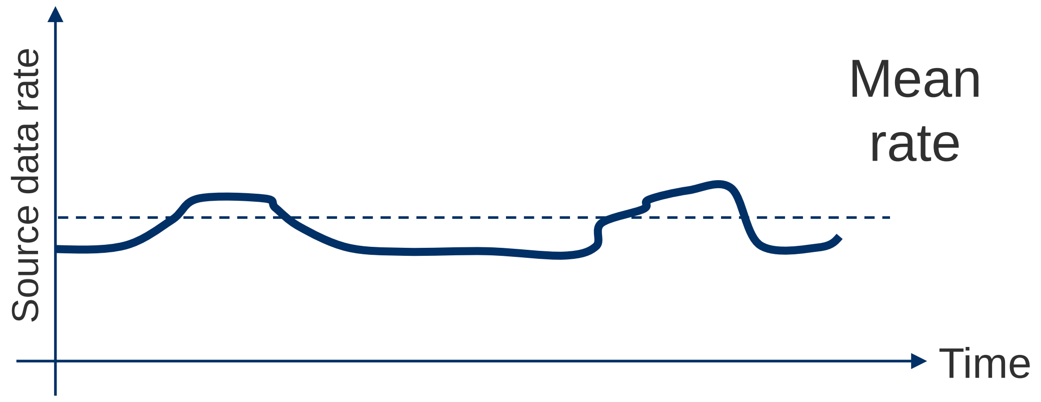
➢Otherwise medium is either not sufficient or not used optimally

Examples: Classical telephone, ISDN, GSM

➢Data rate is given by application, e.g. ISDN (64kbit/s)

Bursty Traffic

➢Problem: What happens if sources have *bursty* traffic?
- Definition: Large difference between peak and average data rate
- Example: Web traffic



In general-purpose computer networks commonly

Peak : Average = 1000 : 1

➢Static multiplexing fails to allocate medium efficiently

Dynamic Channel Allocation / MAC

Alternative: Assign channel to that source that *currently* has data to send

➢ *Dynamic* channel allocation

- Assumes some degree of independence of usage patterns over participating hosts

Terminology: Access to transmission medium has to be organized by **Medium Access Control (MAC) protocol**

Pro: Better utilization of medium based on local (per host) demand

Contra: Dynamic allocation incurs some management overhead, i.e., requires in-band signaling

Figures of Merit

How to judge the efficiency of a MAC protocol?

- ➢ Intuition: Transmit as many packets as quickly as possible

1. At high load (many transmission attempts per unit time):
   Throughput is crucial – ensure that many packets are transmitted
2. At low load (few transmission attempts per unit time):
   Delay is crucial – ensure that a packet does not have to wait for a long time

3. Fairness: Is every station treated equally?

Dynamic/Controlled MAC: Polling

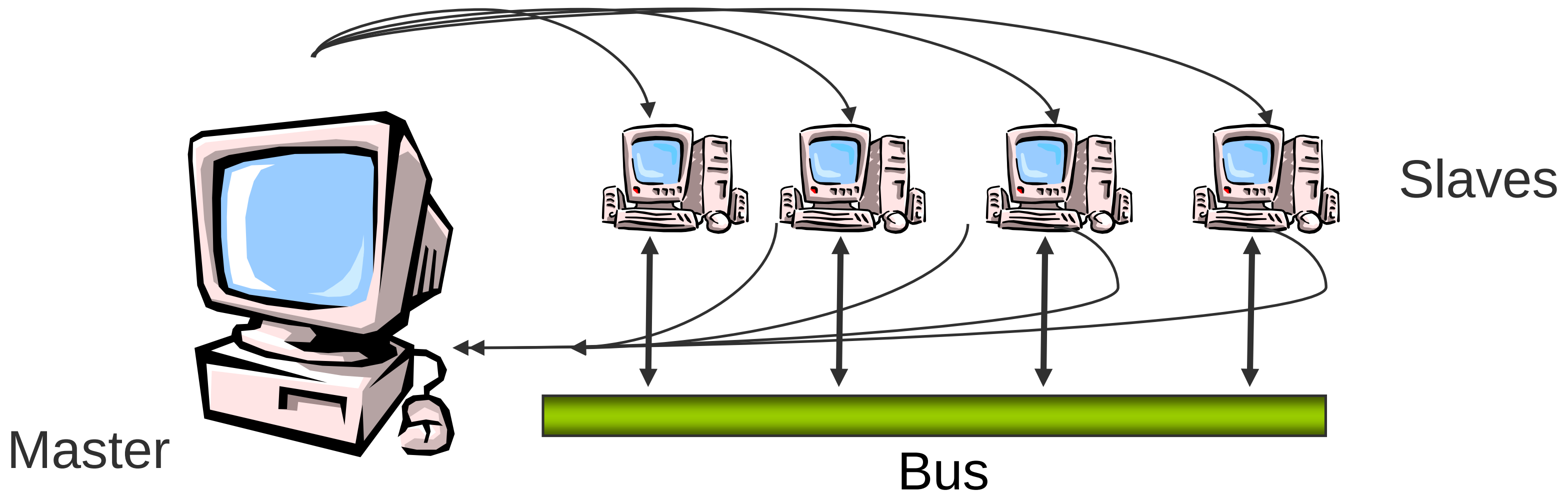
Single master station

One or more slave stations

Typically bus topology (but also tree)

Master polls slaves according to table or cyclically

Slaves may answer only after being polled



➢Examples: Bluetooth, USB, cable modems

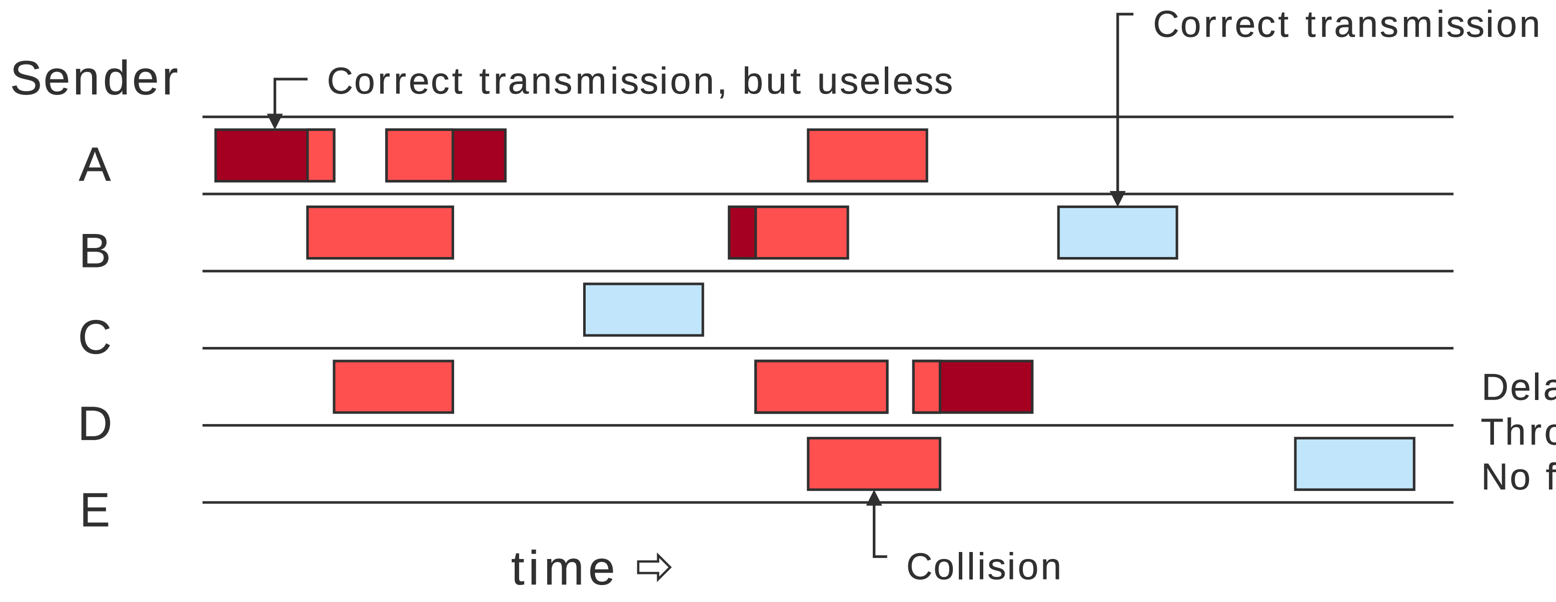
Dynamic/Contention MAC: ALOHA

No central control, no coordination between stations

Stations start sending whenever they want to

- ➢Collisions destroy frames

Receiver may send acknowledgement after correct reception



Delay: 0
Throughput: 25%
No fairness

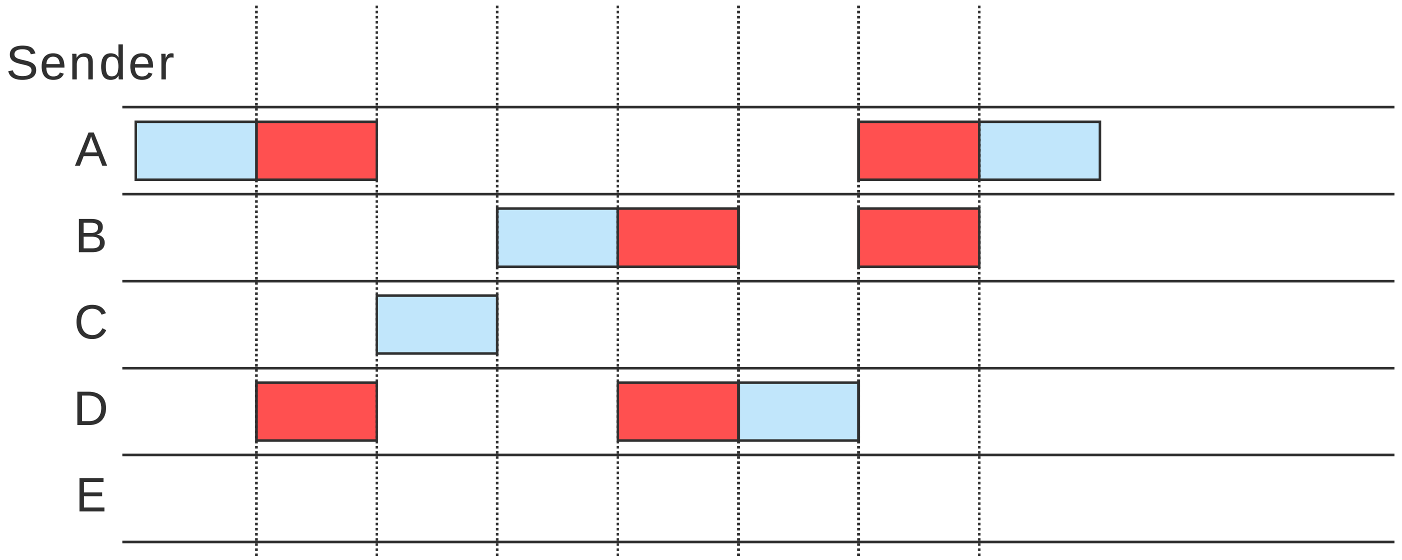
➢Best option without any carrier sensing or central station

Dynamic/Contention MAC: Slotted ALOHA

Send packets of fixed length within fixed time-slots

- ➢Requires common time-base for synchronization

Transmission starts at begin of slot only, thus only complete collisions may occur



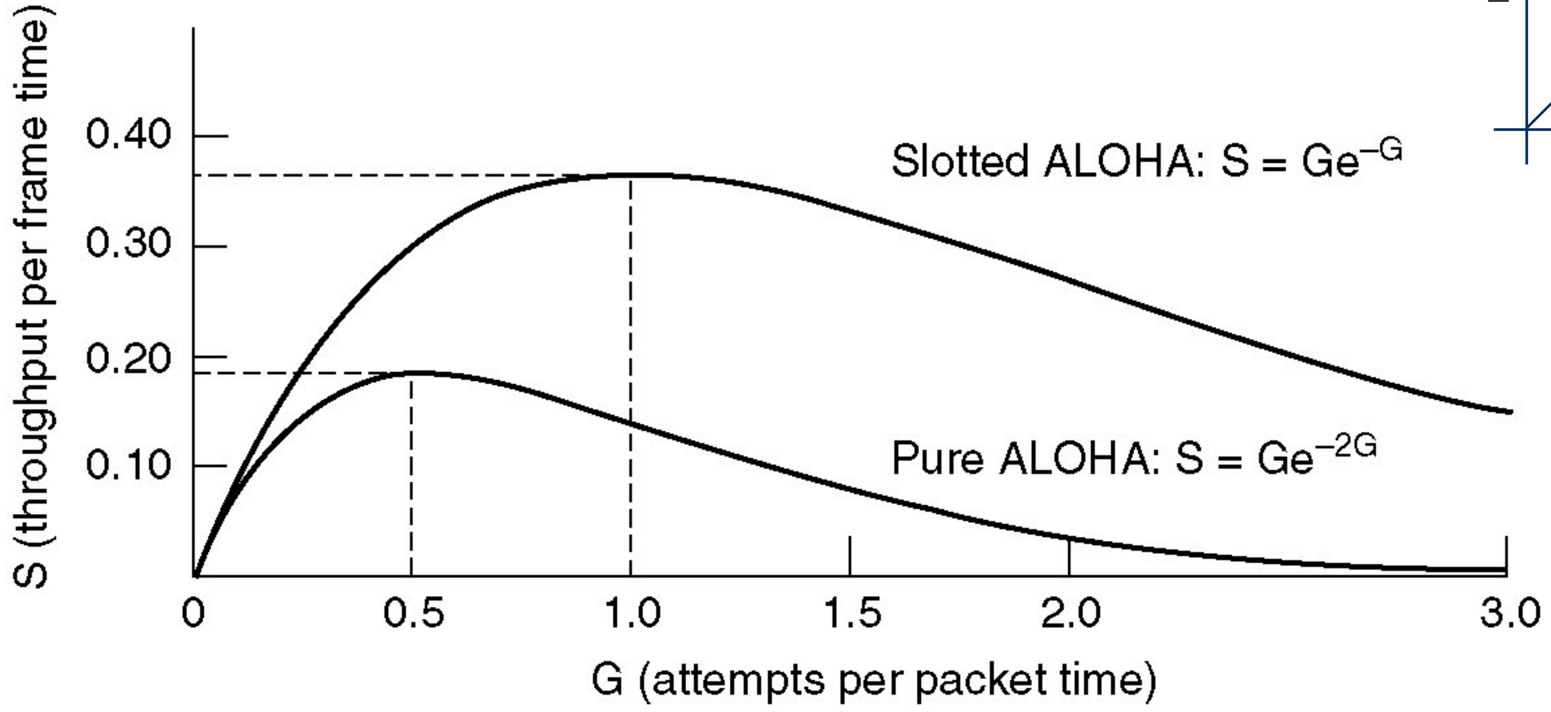
Delay: < slot time
Throughput: 45%
No fairness

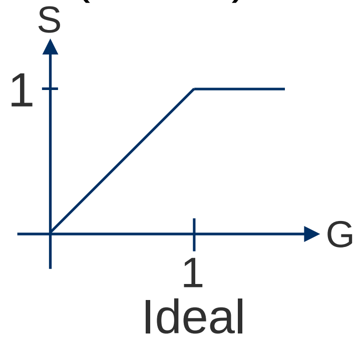
➢Best option without carrier sensing (but sync required)

➢Example: Initial medium access of GSM control channel

Performance Dependence on Offered Load

Closed form analysis (queuing theory) of throughput S as function of offered load G for (slotted) ALOHA:





➢Anything but a high-performance protocol

- Throughput collapses as offered load increases

Performance – Intuition

Aloha

Slotted Aloha

Carrier Sensing

(Slotted) ALOHA is simple, but not satisfactory

- Does not scale over offered load

➢New Approach: Be a bit more polite – **Listen before talk**

- Sense carrier to check whether medium is idle before transmitting
- Do not transmit if medium is busy (some other sender is currently transmitting)

➢**Carrier Sense Multiple Access (CSMA)**

➢But: How to behave in detail when carrier is busy?

- How long to wait before retrying a transmission?

CSMA Strategies

1-persistent CSMA:

- Medium is busy: keep listening until it becomes idle
- Medium is idle: transmit the frame immediately

➢ Problem: if more than one station wants to transmit, they are guaranteed to collide

p-persistent CSMA:

- Medium is busy: keep listening until it becomes idle
- Medium is idle: “flip a coin”
    - With probability p, transmit the frame
    - With probability 1-p, wait until next time slot and then sense the medium again (if next slot idle, flip a coin again)

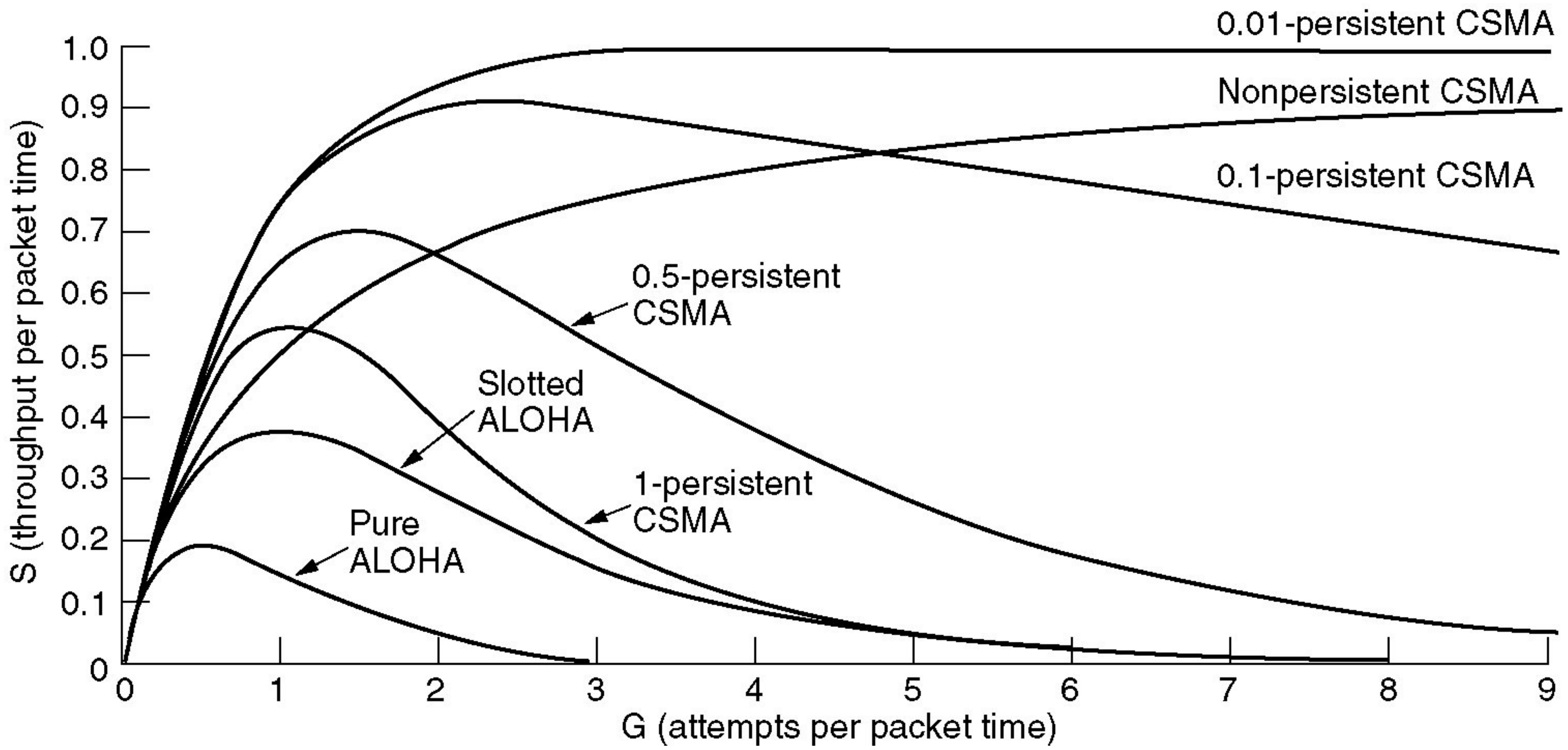
CSMA Strategies

Non-persistent CSMA:

- Medium is busy: wait a random amount of time, before sensing the medium again (don’t be greedy)
- Medium is idle: transmit the frame immediately

➢ Performance depends on random distribution used for waiting time
  - In general, better throughput than persistent CSMA for higher loads
  - At low loads, random waiting is not necessary and wasteful

Performance of CSMA



- ➢Throughput characteristic increases as p decreases
- ➢Potentially long delay before transmission starts

Example: LANs According to IEEE 802

Active Working Groups and Study Groups		Industrial Alliance
802.1	Higher Layer LAN Protocols	
802.3	Ethernet	Ethernet
802.11	Wireless LAN	WiFi
802.15	Wireless Personal Area Network (WPAN)	Bluetooth, ZigBee
802.18	Radio Regulatory	
802.19	Wireless Coexistence	
802.24	Vertical Applications TAG	

See http://www.ieee802.org/

CSMA/CD

Collision Detection:

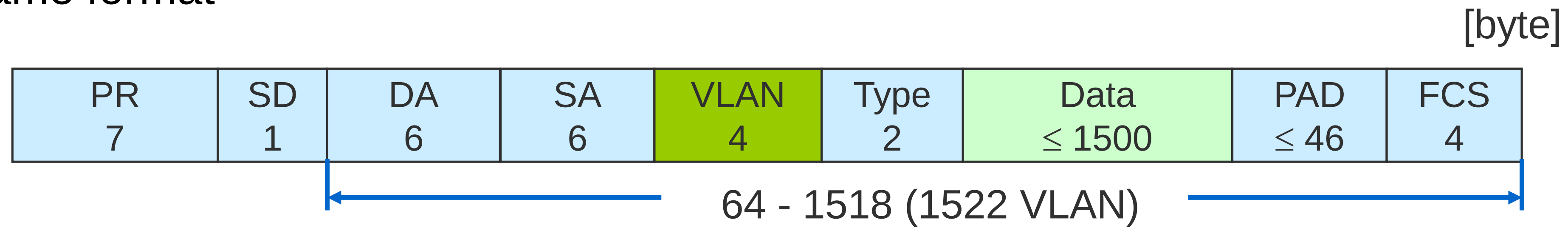
- During transmission, keep sensing the medium to detect collisions
- If collision detected, stop frame transmission and send a JAM signal to guarantee that everyone detects the collision
  - Time wasted on collisions is reduced
- Wait a random amount of time before transmitting again
  - Waiting time is determined by how many collisions have occurred before (exponential backoff algorithm)

Example: Ethernet (with hubs)

- When using switches, there is no need to use CSMA/CD
- Only applicable if nodes are able to detect collisions

IEEE 802.3/Ethernet Frame Format

Common frame format



PR:	Preamble for synchronization
SD:	Start-of-frame Delimiter
DA:	Destination MAC Address
SA:	Source MAC Address
VLAN:	VLAN tag (if present), 0x8100, 3 bit priority, 12 bit ID
Type:	Protocol type of payload (length if ≤ 0x0600), e.g. 0x0800 for IPv4, 0x86DD for IPv6
Data:	Payload, max. 1500 byte
PAD:	Padding, required for short frames
FCS:	Frame Check Sequence, CRC32

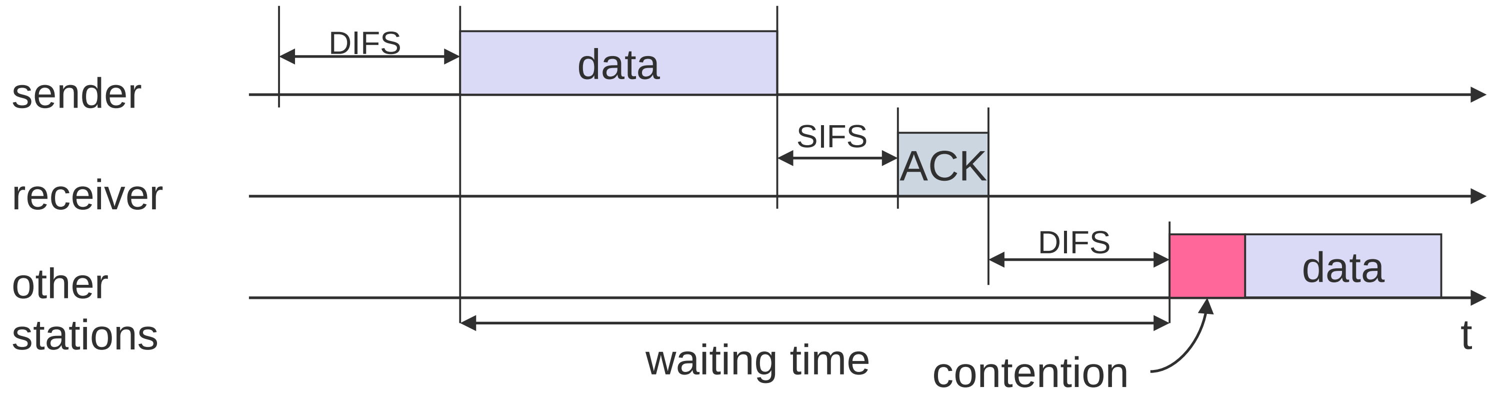
IEEE 802.11 (WLAN) – CSMA/CA

In wireless networks collisions cannot be detected reliably

- Sending and receiving (i.e. sensing collisions) at the same time is difficult
- Hidden terminal problem
- See lecture **Mobile Communications**

Goal: Collision Avoidance

- **Sender**
  - If medium is idle for a certain amount of time slots (DIFS), transmit frame
  - If no ACK received, retransmit frame
- **Receiver**
  - Check if received frame OK (using CRC), send ACK with a short time delay (SIFS)



Conclusion

MAC protocols are crucial for good networking performance

- ➢Static multiplexing just won’t do for bursty traffic

Main categories: Static vs. dynamic, contention (with collisions) vs. controlled (collision-free), hybrid approaches (limited contention)

Main figures of merit: Throughput, delay, fairness

- ➢There hardly is a “best” solution

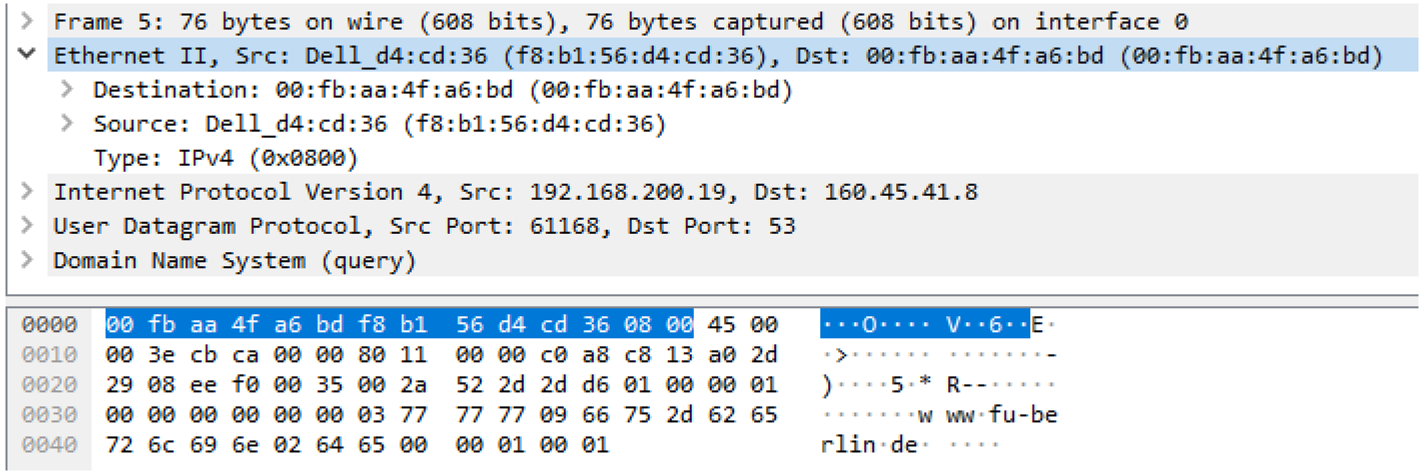
Important case study: Ethernet

- ➢Main lesson to be learned: Keep it simple!

Content

Questions & Tasks

- Why do we need medium access control? Who is controlling?
- Which applications can tolerate collision, which not?
- Compare static vs. dynamic medium access – which is better for what type of application?
- What is the behavior of Aloha in reaction to increasing load? What about polling?
- Go to www.ieee802.org and check the current working groups yourself. Why are there gaps in the numbering?
- If you want to see what's going on in your network – install e.g. WireShark and see for yourself!



```
> Frame 5: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
v Ethernet II, Src: Dell_d4:cd:36 (f8:b1:56:d4:cd:36), Dst: 00:fb:aa:4f:a6:bd (00:fb:aa:4f:a6:bd)
   > Destination: 00:fb:aa:4f:a6:bd (00:fb:aa:4f:a6:bd)
   > Source: Dell_d4:cd:36 (f8:b1:56:d4:cd:36)
     Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.200.19, Dst: 160.45.41.8
> User Datagram Protocol, Src Port: 61168, Dst Port: 53
> Domain Name System (query)

0000  00 fb aa 4f a6 bd f8 b1  56 d4 cd 36 08 00 45 00   ···O···· V··6··E·
0010  00 3e cb ca 00 00 80 11  00 00 c0 a8 c8 13 a0 2d   ·>······ ·······-
0020  29 08 ee f0 00 35 00 2a  52 2d 2d d6 01 00 00 01   )····5·* R--·····
0030  00 00 00 00 00 00 03 77  77 77 09 66 75 2d 62 65   ·······w ww·fu-be
0040  72 6c 69 6e 02 64 65 00  00 01 00 01               rlin·de· ····
```