NSWI090: **Computer Networks**

http://www.ksi.mff.cuni.cz/~svoboda/courses/232-NSWI090/

Lecture 4

# Techniques

**Martin Svoboda**
martin.svoboda@matfyz.cuni.cz

13. 3. 2024

**Charles University**, Faculty of Mathematics and Physics

# Lecture Outline

**Physical layer**

- Analog / digital transmissions
- **Baseband** transmissions
  - Synchronization
  - Line coding
- **Passband** transmissions
  - Modulation

**Data link layer**

- Framing
- Protocols

# Physical Layer

# Transmission Media

**Transmission** media (**physical** transmission paths)

- Guided
  - **Metallic**: twisted pairs, coaxial cables
  - **Optical**: optical fibers
- Unguided
  - **Wireless**: radio, infrared and other transmissions

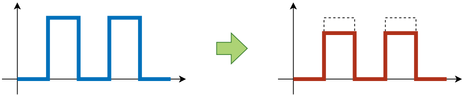
**Signal transmission**

- In all the cases, various forms of **electromagnetic waves** with continuously varying measurable characteristics are carried
  - **Electrical signals**: voltage, current, …
  - **Light pulses**: intensity, …
  - **Radio waves**: frequency, intensity, phase, …

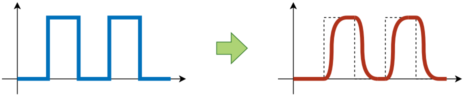
# Properties of Physical Media

**Undesired alterations** of the transmitted signal

- Attenuation: **weakening** of the transmitted signal



- Distortion: **deformation** of the transmitted signal



- Interference: **interweaving** with other signals
  - Including crosstalks from other communication channels

# Properties of Physical Media

Theoretical **objective**

- Receive exactly the same signal as was transmitted
  - Or at least a signal with just enough fidelity so that the original one can be reconstructed

Unfortunately, **real-world paths** are **never optimal**

- Attenuation, distortion, interference, …
- Two parallel wires always act as antenna

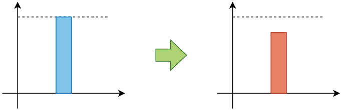
⇒ overall **transmission** potential is always **limited**

- Ability to transmit various signals depends on **frequency** and nature of **transitions**
  - Signals out of bandwidth cannot be transmitted at all
- Impact is **proportional to the distance**

# Analog and Digital Transmissions

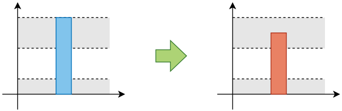
## Analog transmissions

- We are directly interested in the actual measured values
  - E.g.: transmitted voltage $3.4$ V vs. received $3.3$ V



## Digital transmissions

- Space of possible values is divided into discrete intervals
  - E.g.: low level ($0$ V – $1$ V) and high level ($3$ V – $5$ V)



# Analog and Digital Transmissions

**Physical paths** always transmit a certain **analog quantity**

- Only the **interpretation** of received signals differs

**Analog transmissions**

- Will never be optimal = **information always gets damaged**
  - Impact can be reduced, but never completely removed
    - The more we try, the more expensive it will be
    - Moreover, **chaining** of individual paths and passive or active elements within the network only makes it worse

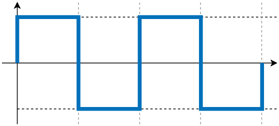
**Digital transmissions**

- Can be optimal
  - **More efficient** (require smaller bandwidth)
  - Without the chaining effect (signal is always regenerated)

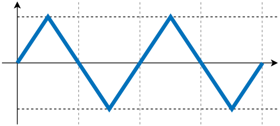
# Waveforms

**Waveform** = basic shape of signal graph as a function of time

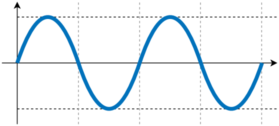
- **Square**



- **Triangle**



- **Sine**
  - $y = A \cdot \sin(\omega \cdot t + \varphi)$
    - **Amplitude** $A$
    - **Frequency** $\omega$
    - **Phase** $\varphi$



- ...

# Waveforms

Observation: **Fourier transformation**

- **Arbitrary wave** can be decomposed to / approximated as a (possibly infinite) **sum of sinusoidal waves**
  - Each with different parameters (frequency, …)

However, **real-world media** have **limited bandwidth**

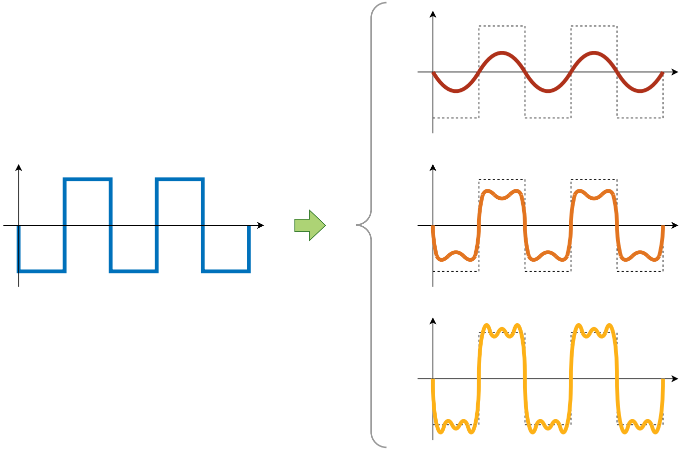
- I.e., range of frequencies that can be transmitted safely
  - Since they only suffer from minimal attenuation and distortion
- **Frequencies out of this range cannot be transmitted**
  - Extent of damage follows the bathtub curve

$\Rightarrow$ **sharp changes** cause complications

- Simply because **higher harmonics are truncated**
  - The less of them are influenced, the higher the signal fidelity

# Waveforms

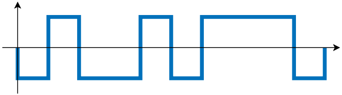
Possible impact of **limited bandwidth** on **sharp changes**



# Baseband / Passband Transmissions

**Baseband transmissions** (unmodulated transmissions)

- **Sequence of pulses with directly encoded data**
  - Near-zero frequency range is used
  - Based on **square waveform** in practice
  - Whole bandwidth carries only one data signal

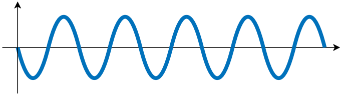


- **Line coding**
  - Information is represented by different signal levels or edges
- Features
  - Easier to implement, common for wired media
  - Only for **shorter distances** (attenuation, distortion)

# Baseband / Passband Transmissions

**Passband transmissions** (modulated transmissions)

- **Harmonic carrier wave**
  - Usually shifted to **higher frequency spectrum**
  - Only frequencies around the carrier are used
    - So that we stay within a given frequency band

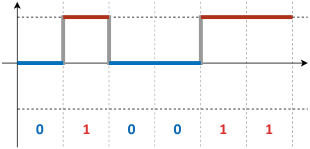


- **Modulation**
  - Information is represented by changes in wave parameters
- Features
  - Even for **longer distances** and with **higher rates**
  - Common for wireless and optical channels

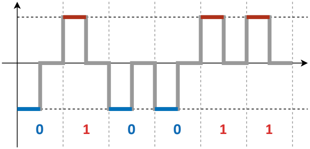
# Baseband Transmissions

**Line code** examples

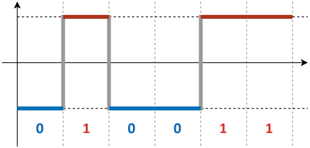
- Unipolar



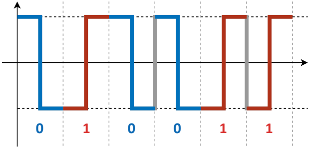

- Bipolar RZ




- Bipolar NRZ




- Manchester



# Baseband Transmissions

Baseband transmissions

- **Sequence** of **electric / light pulses** with **directly encoded data**
  - Frequency of signal changes $\approx$ frequency of data changes

Basic principles

- **Polarity**: number of recognized levels
  - Unipolar (low, high) / bipolar (negative, positive and zero)
- NRZ (**non-return-to-zero**) / RZ (**return-to-zero**)
- Biphase: at least one transition per bit period is required
- **Encoding** of useful values
  - Being at a particular level
  - Making a specific transition
    - E.g.: Manchester (direction of the mid-bit transition)

# Synchronization

**Synchronization** issue

- **Bit period** = time interval needed to send one bit
- **Receiver** must be **synchronized** with the **transmitter**
    - I.e., **timing** between both the devices must be maintained
    - So that the received signal can be **correctly sampled**
- If not, **bit slips** can occur
    - Loss or gain of a bit or more bits, caused by clock drift


- Accurate clocks would help
    - Unfortunately, they are expensive for commodity equipment

# Synchronization

Possible **solutions** for **fully synchronized** transmissions

- **Separate clock signal**
  - No big deal for computer buses, wasteful for most networks
    - I.e., not used in practice
- Self-clocking **approaches**
  - **Clock signal** is **embedded** in the actual **data transmission**
  - Necessary condition
    - **Sufficient number of transitions** in the signal must be ensured
  - Clock recovery
    - Process of extracting the timing information
    - If not ideal, received signal will not be sampled at optimal times

# Synchronization

Common isochronous **self-clocking** techniques

= clock signals are sent at the same time as data

- Direct **recovery**
  - Clock ticks are defined by data transitions themselves
    - It is guaranteed they occur on a regular basis in each bit period
  - Technique: redundant coding
- Indirect **recovery**
  - Clock ticks must be derived from the actual sequence of data
    - Assumption: synchronization is preserved for a certain time
    - Apparently, **longer** runs of the same bit **must be avoided**
  - Techniques: bit stuffing, block coding

Alternative anisochronous **self-clocking** techniques

= clock signals are sent at different times to data

# DC Component and Disparity

**DC component** = **mean amplitude of the waveform**

- Motivation
  - **Long-distance** paths cannot reliably transport a DC component
- Desired situation: **balanced DC** (no DC component, …)
  - I.e., mean amplitude is zero
- Approaches
  - Constant-weight code: each symbol is balanced on its own
  - Paired disparity code: balancing across individual symbols

**Running disparity**

- **Difference between the number of transmitted 0 and 1 bits**
- Desired situation: **bounded disparity**
  - Disparity should be **as near to neutral as possible**

# Line Coding Techniques

## Redundant coding

- **Every bit period inherently contains at least one transition**
- Ensures synchronization
- **100% overhead**
  - Doubles the required bandwidth
- Examples
  - **Manchester**
    - Direction of the mid-bit transition determines the data bit
    - **Clock signals occur at these mid-bit transitions**
    - Self-clocking, DC balanced, 100% overhead
    - Unsuitable for higher data rates
    - Usage: Ethernet (10 Mb/s), NFC, …
  - **Bipolar RZ**

# Line Coding Techniques

## Bit stuffing

- **Artificially added opposite bit is sent after every long run**
  - Receiver automatically removes such extra bits
- Ensures synchronization
- Overhead tends to 0%

## Scrambling

- **Bits to be sent are mixed with a pseudo-random sequence**
  - Receiver must also be able to generate exactly the same sequence so that the original data can be reconstructed
- Helps with bounded disparity and synchronization

# Line Coding Techniques

**Block coding schemes**

- **Tuples of $n$ bits are translated to $k > n$ bits before sending**
  - Based on fixed **dictionary mappings** or other rules
  - E.g.: 0001 $\rightarrow$ 01001, …
- Features
  - Output tuples with the most **changes** are preferred
    - So that sufficient number of transitions is achieved
  - Multiple **alternatives** may exist for one input tuple
    - May intentionally be chosen based on the previously sent data
  - Several output tuples may remain unused
    - Serve as **low-level control signals** or for simple **error detection**
- **Reasonable overhead**

# Line Coding Techniques

Block coding **schemes** (cont'd): **examples**

- **4B5B**
  - **Maps groups of 4 bits onto groups of 5 bits**
  - Self-clocking, DC balanced with scrambler only
  - 25% overhead
  - Usage: Fast Ethernet (100 Mb/s), …
- **8b/10b**
  - **Maps groups of 5+3 bits onto groups of 6+4 bits**
  - Self-clocking, DC balanced, bounded disparity
    - **Run-length limit** of 5 consecutive equal bits
    - **Running disparity** is guaranteed to be not more than $\pm 2$
  - 25% overhead
  - Usage: Gigabit Ethernet (1 Gb/s), HDMI, SATA, USB 3.0, …

# Line Codes

**Line code** = particular line coding approach

- May mutually combine various generic techniques

Main tasks

- Optional **higher level**
  - Sequence of input bits is logically **scrambled** and / or **converted**
- **Lower level**
  - Prepared sequence is then physically **transmitted**
    - Patterns (distinct levels or transitions) of voltage, current, or photons must be devised to represent the data to be sent

# Line Codes

Generic **objective**

- Ensure **regular, frequent and evenly distributed changes** in the transmitted **logical data** / **physical signal**
- Obstacles
  - Input data is **not under our control**
  - Features unique to each physical medium must be reflected
    - Attenuation, distortion, interference, …
- Practical consequences
  - **Higher rates** and **higher reliability**

Particular **goals**

- Facilitate synchronization, eliminate a DC component
- Minimize transmission hardware

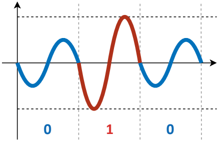
# Passband Transmissions

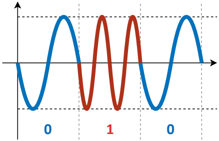
**Passband transmissions**

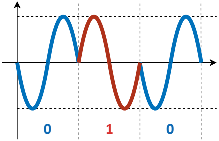
- **Harmonic carrier wave** with **modulated data**
  - Frequency of signal changes $\gg$ frequency of data changes

(Analog) **modulation**

- Process of signal shaping so that it can convey useful data
- **Amplitude (AM) / frequency (FM) / phase (PM) modulation**




- Usage of varying amplitudes, frequencies and / or phases
  - Including (some of) their **mutual combinations**

# Passband Transmissions

Keying = **digital modulation**

- Fundamental techniques
  - Amplitude-Shift Keying (**ASK**): different **amplitudes**
  - Frequency-Shift Keying (**FSK**): different **frequencies**
  - Phase-Shift Keying (**PSK**): different **phases**
- Only **limited number of states** is recognized
  - In contract with the traditional analog modulation
- Observations
  - Once again, **mutual combinations** are possible
  - Reliability of **change detection** ability varies
    - Phase modulation is the most efficient one (because of the sharpest changes)

# Quadrature Amplitude Modulation

Example: **Quadrature Amplitude Modulation** (QAM)

- Family of both analog and digital modulation techniques

Alternatives

- **16-QAM**: 16 states, 1 symbol = **4 bits**
- **64-QAM**: 64 states, 1 symbol = **6 bits**
  - Wi-Fi 2 (802.11a), Wi-Fi 3 (802.11g), Wi-Fi 4 (802.11n), DVB-T
- **256-QAM**: 256 states, 1 symbol = **8 bits**
  - Wi-Fi 5 (802.11ac), DVB-T2
- **1024-QAM**: 1024 states, 1 symbol = **10 bits**
  - Wi-Fi 6 (802.11ax)

# Quadrature Amplitude Modulation

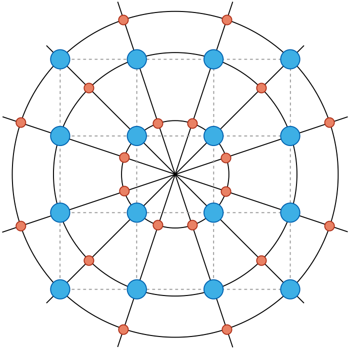
## 16-QAM

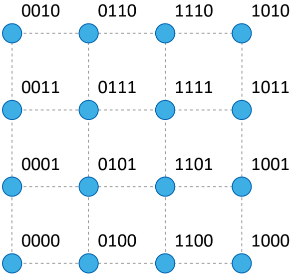
- **2 carriers** shifted by $90^\circ$ ($\Rightarrow$ quadrature / orthogonal)
  - $1^{st}$ wave: **amplitude** modulation with 3 states
  - $2^{nd}$ wave: **phase** modulation with 12 states
- Altogether 36 states
  - Only 16 of them are actually considered
    - Those with higher mutual distances
- **Mapping of states to symbols**: **Gray code**
  - Adjacent symbols differ in exactly 1 bit position
    - So that small burst of noise does not lead to many bit errors

# Quadrature Amplitude Modulation

## 16-QAM

- **Constellation diagram**





# Data Link Layer

# Transparency

## Transparency

- **Separation of useful data from control signals**
  - Since they need to be treated differently

**Useful data**

- **Data to be transmitted** for the higher layer
- **Without any changes**
  - More precisely, certain modifications may be unavoidable, but recipient must be able to reconstruct the original data

**Control signals**

- **Commands** to be correctly **interpreted and executed**
  - Allow to fulfill objectives of data link layer as such

# Transparency

**Strategies** for **stream** transmissions

- Separate path
  - One wire is dedicated to control signals only
  - E.g.: older modems for dial-up Internet access via PSTN
  - Cumbersome, complicated
- **Escaping**
  - Two modes: data / commands
  - Mechanisms for **interpretation switching** must exist
  - Usage
    - Newer dial-up modes
    - Peripheral devices such as mouses, printers, …

**Strategies** for **block** transmissions

- **Framing**

# Framing

## Encapsulation

- **Construction of a frame** (PDU in general)
  - **Frame format** needs to be defined
    - Internal structure: **header**, **body**, **footer**
    - Individual **header fields** and their meaning

## Framing

- **Delineation of frame boundaries**
  - So that recipient can correctly recognize individual frames
    - Simply because L1 only provides a **stream of raw bits**
    - And so frames as **blocks of data** (their beginnings and endings) must be found within this unstructured sequence

$\Rightarrow$ **encapsulation $\neq$ framing**

# Framing

Generic framing **techniques** (independent on L1)

- **Starting and ending flags**
  - **Special flags** (byte, sequence of bits, …) are added to explicitly **mark frame beginning and ending**
  - Occurrences of such flags inside frames (especially in payload) must be appropriately treated
    - Bit stuffing, positive / negative escaping, duplication, …
- **Starting flag and length**
  - Special flag is added to explicitly **mark frame beginning**
  - **Frame ending** is detected indirectly using the **frame length**
    - Which is provided as one of the header fields
  - Not widely used
    - Since restoring order after desynchronization can be difficult

# Framing

Specific framing **techniques** (dependent on L1)

- **Starting flag and implicit ending**
  - **Frame beginning** is marked explicitly as above
  - **Frame ending** is detected by the absence of the **carrier**
  - E.g.: Ethernet II with Manchester coding
- **Line coding violations**
  - **Special non-data symbols** provided by a given physical **line code** are used to **mark frame beginning and ending**
  - E.g.: 4B5B
- **Counting of blocks**
  - Individual blocks are counted using time-division multiplexing
    - Only works for blocks with fixed sizes
  - E.g.: digital hierarchies

# Data Link Protocols

Data link **protocols**

- Distinguished with respect to the **unit of data** they work with
  - I.e., data **granularity level** they assume
- In particular: individual characters / bits / bytes

**Groups of protocols**

- **Character-oriented protocols**
  - Older protocols, no longer in use
  - E.g.: SLIP (Serial Line Internet Protocol)
- **Bit-oriented protocols**
  - E.g.: HDLC (High-Level Data Link Control)
- **Byte-oriented protocols**
  - E.g.: Ethernet

# Stuffing

**Character / bit / byte stuffing**

- General technique allowing to **mark specific points** in data by **adding extra symbols**
  - Individual characters / bits / bytes depending on the protocol

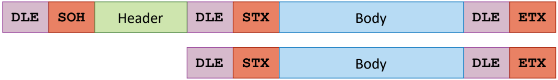
**Use cases** (at L2)

- Ensuring **transparency** itself
  - Escaping
  - Framing: flags delimiting frame beginnings and endings, …
- Treating **occurrences of flags** in useful data
  - So that their otherwise intended meta meaning is suppressed
- **Cooperation of physical and data link layers**
  - Synchronization, …

# Character-Oriented Protocols

## Character-oriented **protocols**

- Control commands are expressed using special **non-printable ASCII characters**




## Frame structure

- Header and body
  - SOH (**Start of Header**): beginning of the optional header
  - STX (**Start of Text**): beginning of the payload to be sent
  - ETX (**End of Text**): ending of the entire frame
- Positive **escaping** is needed to activate the meta meaning
  - DLE (**Data Link Escape**)

# Character-Oriented Protocols

**Payload** (and header) **transparency**

- Each occurrence of DLE symbol is **doubled**

**Synchronization** for L1

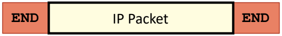
- Two synchronization characters are added before the frame beginning to help the physical layer
    - SYN (**Synchronous Idle**)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **SYN** | **SYN** | DLE | SOH | Header | DLE | STX | Body | DLE | ETX |

# Character-Oriented Protocols

**Example**: SLIP (**Serial Line Internet Protocol**)

- Simple protocol allowing for direct encapsulation of **IP packets**
  - Intended **only for P2P** and fully duplex physical paths
    - Such as using a modem over local loops in PSTN
  - Only framing is necessary, nothing else (no addressing, …)




**Framing** principles

- Frame beginning and ending are marked using `END` (`0xC0`)
- **Payload transparency**
  - Each `END` is replaced with `ESC` (`0xDB`) `ESC_END` (`0xDC`)
  - Each `ESC` is replaced with `ESC` (`0xDB`) `ESC_ESC` (`0xDD`)
  - I.e., transposed characters are used instead of the original ones

# Bit-Oriented Protocols

## Bit-oriented protocols

- Special **sequence of bits** (flag) is used for marking both **frame beginning and ending**
  - Typical structure: $N$ ones wrapped inside a pair of zeros
    - E.g.: 0111111110 ($N = 8$)

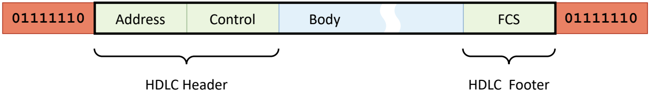

## Payload transparency

- Whenever the sender comes across a sequence of $N - 1$ ones, extra zero is added
  - Recipient removes this zero to get the original data

# Bit-Oriented Protocols

**Example**: HDLC (**High-Level Data Link Control**)

- Connection-oriented as well as connectionless protocol for P2P and P2MP paths
  - Standardized (ISO/IEC 13239:2002)
  - Inspiration for plenty of other protocols
    - IEEE 802.2 LLC (**Ethernet LLC** frames), PPP, LAPD (ISDN), …




**Framing** principles

- **Flag** with $N = 6$ ones is used
- Bit stuffing for block transparency

# Byte-Oriented Protocols

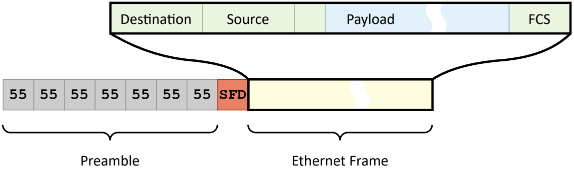
**Byte-oriented protocols**

- **Trade-off** between both the previously discussed groups
  - Usage of **flags**
    - As in **bit-oriented** protocols
    - The same idea, only **alignment to whole bytes is required**
    - E.g.: 01111110 ($N = 6$)
  - Usage of **escaping bytes**
    - For the purpose of **payload transparency**
    - As in **character-oriented** protocols
    - **Higher overhead** when compared to bit stuffing
  - **Synchronization bytes** at the beginning may also be used

# Byte-Oriented Protocols

**Example**: Ethernet

- Two basic types of frames are used (Ethernet II, 802.3)




**Framing** principles

- **Synchronization preamble** at the beginning
  - Sequence of 7 bytes `0x55` each transferred as `10101010`
  - Big Endian for bytes, Little Endian for bits within a byte
- Frame beginning
  - `SFD` (**Start Frame Delimiter**): `0xD5` transferred as `10101011`

# Lecture Conclusion

**Baseband** transmissions

- **Synchronization**: self-clocking, clock recovery
- DC component, running disparity
- **Line codes**

**Passband** transmissions

- Amplitude / frequency / phase **modulation**

**Data link protocols**

- Transparency: escaping, **framing**, **stuffing**
- **Protocols**
  - Character-oriented, bit-oriented, byte-oriented