PERSONAL - Introduction to Networking

Layer 1 - Physical Layer

- **Layer 1**: *transmission and reception* of *raw bit streams* between a *device and a shared physical medium*
- **Hub**: anything the hub receives via a port is *retransmitted* to all other ports $\rightarrow$ *single* collision domain, impacts *all* devices
- **(!)** *no device addressing* at this layer! $\rightarrow$ *all data* is processed by *all devices*
- **(!)** no collission detection / resistance!

Information Content, Entropy

- **Information Content (Informationsgehalt)**: the level of "surprise" of a particular outcome (the *less probable* an event is, the *more surprising* it is and the *more information* it yields)

    - an event with probability 100% is *perfectly unsurprising* and yields *no information* (i.e. $I(x) = 0$)
- **Entropy**: average level of "surprise"

Signal Processing

- **Fourier Series**: defining a *periodic (!) function* as a *sum of trigonometric functions* ($\sin, \cos$)
- **Fourier Transform**: integral transform that takes a *non-periodic function* as input and outputs another function that describes the extent to which various frequencies are present in the original function
- **Sampling (Abtasten)**: reduction of a *continuous-time signal* to a *discrete-time signal*
- **Quantization (Quantisierung)**: process of mapping input values from *a large, continuous set* to output values in a *smaller, countable set* (i.e. splitting up the graph into segments based on amplitude)
- **Attenuation (Dämpfung)**: maximum amplitude of signal is limited / lowered
- **Low Pass Filter (Tiefpassfilterung)**: signal edges are smoothed (e.g. rectangular signal, smooth transition from 1 to 0 instead of abrupt)
- **Nyquist Rate**: $f_N = 2B$ for cutoff / highest frequency (bandwidth) B in Hz
    - *lower bound* for sampling rate to allow complete reconstruction of original signal (i.e. to prevent aliasing)
        - **aliasing**: *overlapping* of frequency components; different signals become *indistinguishable* from each other when sampled

Coding Theory

- **Source Coding (Quellenkodierung)**: take the source data and make it smaller by removing redundancy (i.e. data compression, see Huffman code)
- **Channel Coding (Kanalkodierung)**: add redundancy to correct or at least detect many errors (e.g. linear codes)
- **Line Coding (Leitungskodierung)**: representing the digital signal to be transported (i.e. the bits) by a time-discrete signal (NRZ, RZ, Manchester, MLT3...)
    - **Self-clocking Signal (Taktrückgewinnung)**: basically, if there's a long sequence of 1s or 0s, you should be able to tell how many 1s or 0s were transmitted independent of a separate "clock" and when the signal has stopped transmitting (idle)
    - **No DC Bias / DC Free (Gleichstromfreiheit)**: waveform with zero mean
- **Modulation**: the data (base) modifies the carrier signal (some high frequency signal) to encode information for long-range communication, simultaneous wireless internet connections etc.
    - **Amplitude Shift Key (ASK)**: modify the *amplitude* of the carrier
    - **Frequency Shift Key (FSK)**: modify the *frequency* of the carrier
    - **Phase Shift Key (PSK)**: modify the *phase* of the carrier

- **Quadrature Amplitude Modulation (QAM)**: modify the *amplitude and phase* of the carrier

Layer 2 - Data Link Layer

- **Layer 2**: handles *how data moves in and out* of a physical link in a network; *local* delivery of *frames* between nodes on the same level of the network
- **Frame**: format for sending information over a layer 2 network; *specific to local network*
  - **MAC header**:
    - **preamble**: start of frame
    - **destination / source MAC addr.**
    - **ethertype**: layer 3 protocol (e.g. IPv4)
  - **payload**: 46-1500B
    - L3 data
  - **frame check sequence (CRC)**: 4B
- **MAC-Address**: unique hardware address (48 bits, 6 bytes)
  - **first 3 bytes**: OUI (organizationally unique identifier; given by the company)
  - **broadcast**: `ff:ff:ff:ff:ff:ff`

Delays

- **Serialization Delay / Transmission Delay (Serialisierungszeit)**: how long it takes to physically put the packet on the wire
  - **formally**: $t_s = \frac{L}{r}$ for L data bits and rate r in bit/s
- **Propagation Delay (Ausbreitungsverzögerung)**: the flight time of packets over the transmission link, limited by the speed of light
  - **formally**: $t_p = \frac{d}{\nu c_0}$
- **Delay (Gesamtverzögerung)**: $t_d = t_s + t_p$
- **Bandwidth-Delay Product (BDP) (Bandbreitenverzögerungsprodukt)**: the maximum amount of data on the network circuit at any given time (i.e. capacity, in bit)
  - **in other words**: how many bits the sender can serialize before the first bit reaches the receiver
  - **formally**: $C = t_p r$

Multiplexing (simplified)

- **Multiplexing (muxing)**: multiple analog or digital signals are combined into one signal over a shared medium
- **Time Division Multiplexing (TDM)**: for a certain time slice, the bandwith belongs to one connected terminal only

- **Frequency Division Multiplexing (FDM)**: the total bandwidth is divided into a series of non-overlapping frequency bands, each of which is used to carry a separate signal

ALOHA, Slotted ALOHA, CSMA

- **(Pure) ALOHA**: random access protocol; whenever a frame is available, send as soon as possible (*continuous time interval*) to a *central (base) station*
  - *collision* if 2 stations are transmitting at the same time
  - *out-of-band acknowledgement* on different frequency
  - **vulnerable period**: $2T$ for send duration T
  - **probability of successful transmission**: $p_0 = \lambda e^{-2\lambda}$
- **Slotted ALOHA**: random access protocol; frame can only be sent in *discrete, globally synchronized time slot* (hence the name...)
  - **vulnerable period**: T for send duration T
  - **probability of successful transmission**: $p_0 = \lambda e^{-\lambda}$
- **Carrier Sense Multiple Access (CSMA)**: check carrier signal (i.e. if there is any data being transmitted on layer 1) ; try sending only when medium *idle*
  - **1-persistent** (aggresive):
    - **medium idle**: begin transmission immediately
    - **medium busy**: keep checking until idle
  - **p-persistent**:
    - **medium idle**: begin transmission with probability p or wait random amount of time with probability $1 - p$, then retry this step...
    - **medium busy**: keep checking until idle
  - **non-persistent**:
    - **medium idle**: begin transmission immediately
    - **medium busy**: wait random amount of time, then check again
- **Carrier Sense Multiple Access / Collision Detection (CSMA/CD)**: CSMA + if collision gets detected, *jam signal* is sent by all devices which detect collision and backoff for a random amount of time
  - **prerequisite for collision detection**: $t_s > 2t_p$
  - no collision detected $\rightarrow$ transmission successful
- **Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA)**: CSMA + if transmission medium is sensed busy before transmission, then the transmission is deferred for a random interval
- **Binary Exponential Backoff**: on k-th attempt, choose $n \in \{0, ..., \min\{2^{k-1} - 1, 1023\}\}$ randomly and wait n slots (max. wait time 1023 slots)

- **in other words**: contention window $\{0, \dots, 2^{k-1}\}$ doubles with each attempt, then resets when successful

L2 Hardware

- **Switch**: *stores, reviews and forwards frames*; understands frames and MAC addresses; *only valid frames* are forwarded
  - **MAC address table**: contains *MAC addresses of each connected device* on each *port / interface* (e.g. `eth0`, `eth1`...), populated over time
  - when a device sends a frame via a hub...
    - if dest. MAC address is *present* in table, forward *only to corresponding* port
    - if dest. MAC address is *not present* in table, forward to *all* ports
  - **(!)** n ports $\rightarrow$ n collision domains (one collision limited only to one port)
- **Bridge**: switches with only 2 / 4 ports
- **Access Point (AP)**: allows wireless connections, *not* transparent (unlike switches)

Layer 3 - Network Layer

- **Layer 3**: provides the means of *transferring variable-length network packets from a source to a destination host via one or more networks*
- **(!)** L3 provides *no ordering mechanism*! $\rightarrow$ packets may arrive out-of-order
- **(!)** L3 cannot ensure packet delivery! $\rightarrow$ packets may go missing (e.g. hop limit exceeded)
- **(!)** L3 cannot distinguish applications! $\rightarrow$ no communication channels (i.e. besides source and destination IP, there is no further application or channel distinction)
- **(!)** L3 has no flow control $\rightarrow$ if the source sends packets faster than the destination can receive them, packet loss may happen

IP, IPv4

- **Internet Protocol (IP)**: L3 protocol for *cross-network IP addressing and routing* to move data between *different* LANs
- **Packet**: like a frame, but source and destination are global
  - *remains the same* across the entire journey from source to destination (with some minor exceptions, e.g. NAT)
  - **Maximum Transmission Unit (MTU)**: maximum packet (L2 payload) length without requiring fragmentation
- **Address Resolution Protocol (ARP)**: protocol to find MAC address for a given IP address
  - **ARP Cache**: stores MAC addresses to prevent multiple ARP requests
- **IPv4 Address**: 4 sets of 8 bits (32 bits, 4 bytes, 4 octets)

    - two parts:
        - **network part**: which IP network does this IP address belong to?
            - if the network part of two IP addresses match, they're on the same network (local), else they're on different networks (remote)
        - **host part**: which host in this network is this?
            - **first address**: network space (host all 0s)
            - **last address**: broadcase address (host all 1s)
    - **Slash (CIDR) Notation**: `XXX.XXX.XXX.XXX/YY` $\equiv$ `YY` bits are used for the network part (from the left), the remaining `32 - YY` for the host part
    - **Dynamic Host Configuration Protocol (DHCP)**: dynamic, automatic IP address assigned by DHCP server
- **Subnet Mask**: represents, which part of an IP address is for the network; allows host to determine if IP address *is local or remote* (i.e. if *default gateway* is needed or not)
    - binary 1: network part
    - binary 0: host part
        - starting address of network: host part of `IP & MASK` all 0s
        - end address of network: host part of `IP & MASK` all 1s

Autonomous Networks

- **static routing**: routing inside of an autonomous system via *routing tables*
- **dynamic routing**: routers communicate with eachother to share connections
    - **distance vector protocol**: routers only know next hops and costs to get to destination
    - **link state protocol**: routers communicate and share if a node goes down
- **autonomous network**: network of networks; central router to allow connections to / from other ANs and broadcast direct connections of AN

L3 Hardware

- **Router**: *moves* packets of data *across different networks*; *removes frame encapsulation, adds new frame encapsulation* at every hop (for each step of the way from the source to the destination...)
    - **Route Table**: collection of routes (destination; next hop), used with *longest prefix match*

Layer 4 - Transport Layer

- **Layer 4**: provides *end-to-end communication services* for applications
- **Transmission Control Protocol (TCP)**: used for "good", reliable connections; *guarantees* that data is received *correctly, in-order*

    - **segment**: *data container*, encapsulated in a *packet*
        - **port(s)**: allows application-specific communication, since the complete protocol is based on a combination of *IP addresses and ports* to identify *communication channels* (TCP/IP)
        - **sequence number**: solves out-of-order problem
        - **acknowledgement**: each transmitted segment needs to be acknowledged $\rightarrow$ ensures packet delivery
        - **window**: number of bytes you're willing to receive between acknowledgements; once reached, sender will *pause* until you acknowledge that amount of data $\rightarrow$ flow-control
    - **connection-oriented**: connection between two devices needs to be setup beforehand via 3-way handshake $\rightarrow$ bidirectional communication channel
        1. **SYN**: client to server
        2. **SYN-ACK**: server to client
        3. **ACK**: client to server
    - **Maximum Segment Size (MSS)**: maximum size of TCP payload
    - **Flow Control (Flusskontrolle)**: prevent overloading the *receiver* (via *window* in TCP header)
    - **Congestion Control (Staukontrolle)**: prevent overloading the *network* (via *congestion window*)
    - **when?** reliability, error-correction, ordering of data; slower
    - **examples**: HTTP(S), SSH...
- **User Datagram Protocol (UDP)**: *connectionless* - doesn't establish session, doesn't guarantee data delivery
    - **when?** less reliable; faster (no TCP overhead); DNS
- **Network Address Translation (NAT)**: router translates *private IPs (and possibly ports) of hosts* inside private network to *public IP address of router* when accessing another network (and *vice-versa*)
    - **NAT Table**: stores local IPs, local ports (can have duplicates) and global ports (must not have duplicates)
    - **port forwarding**: manually tell router where to forward incoming requests with a certain port to

C

- `struct sockaddr_in`: contains address family (`AF_INET` or `AF_INET6`), port number and IP address (need to be converted to network byte order!)
- `struct in_addr`: IPv4 address in *network byte order*

UDP

- `socket()`: creates new socket, returns socket descriptor
- `bind()`: associates address to previously created socket (descriptor), returns 0 on success

- `select()`: monitors multiple file descriptors, waiting until one or more of the file descriptors become "ready", then modifies file descriptor set to only contain ready FDs; returns how many FDs are ready
- `recvfrom()`: read from socket
- `sendto()`: send via socket

TCP

- `socket()`: same as before...
- `bind()`: **server side**, same as before...
- `connect()`: **client side**, connect to server
- `listen()`: **server side**, mark socket as *passive* and wait for incoming connections
- `select()`: same as before, used here to choose a connection
- `accept()`: accepts connection and *creates a new socket*
- `recv()`: read from socket
- `send()`: send via socket

Layer 5 - Session Layer

- **Layer 5**: allows the two sides to establish and use a connection
- **Domain Name System (DNS)**: resolves domain names to IP addresses
  - **Name Server**: stores information about a fraction of the domain namespace
    - **Root Server**: top of DNS hierarchy, redirects to corresponding TLD servers
    - **TLD Server**: stores address information for specific TLDs (e.g. .com, .net, .org...)
    - **Authoritative Name Server**: responsible for knowing everything about the domain, including the IP address (e.g. Google's ANS knows everything about every subdomain of Google and the main website itself)
  - **DNS Zone**: section of domain name space that some administrator has control over; for manageability
  - **Resolver**: middleman between client and nameservers; sends requests to nameservers and passes information back to client
  - **Reverse DNS**: determine the domain name associated with an IP address
- **Fully Qualified Domain Name (FQDN)**: sequence of *labels* separated by dots
  - **example**: `www.example.com.` (implicit `.` at the end)
    - `.`: root domain
    - `com`: TLD (top level domain)
    - `google`: SLD (second level domain)

- `www`: subdomain