CNDS SUMMARY

E1: Networks, Architectures, Services and Protocols

The main goal of computer networks is to exchange information.

Major network components:

1) End-systems: An end system is a device that is connected directly to a computer network, creating an interface that individual users can access. They offer services to users (customers) and they are fixed or mobile. Examples: terminals, computer, nodes, hosts (IETF).
2) Intermediate systems: network equipment (e.g., routers) that do not directly support users, but forward received data onward towards the intended recipient. Intermediate systems do not need to understand the information being sent between the users but do understand and may modify the information added by the network to provide the communication. They make sure that end-systems can connect between each other. An intermediate system is a source and a sink, it means that it can receive messages from a source, but it can also act as a source to transmit the data.
   simple: concentrators and bridges, complex: switches, IP/Wi-Fi routers.
   - examples: bridges, routers, gateways, LAN switches.
3) Links: a link is a communication channel that connects two or more devices/systems for the purpose of data transmission.
   Access-link: type of link that is the first source to the network.
   Backbone link: type of link used by providers that cannot be seen.
   Examples of links: phone lines, cables, circuits, channels, radio links,
   coaxial cables, fiber optics.

A computer network is a set of (at least two) autonomous computers exchanging information, where eachmachine operates on a dedicated memory (to store data), processor and operating system. Major difference between CN and DS is transparency (in DS you don't see anything). Examples: LAN, PAN, WAN, MAN, GAN.

System: a system is a regularly interacting or interdependent group of components forming a unified whole. Systems together form a unified activity.

Distributed System: it is a set of geographically distributed (in different physical locations), autonomous and interconnected computers, which offer a service based on cooperation. Examples: World Wide Web, peer-to-peer networks, telephone networks and cellular networks.

Application: An application is a computer program designed to carry out a specific task other than one relating to the operation of the computer itself, typically to be used by end-users. It is a single host-based service. Examples: Word processing, drawing, media players

Distributed application: it is an application running on multiple, cooperating, geographically distributed computers and delivering a specific service to a user. A distributed application implements a special instance of a service in a distributed system. Example: e-commerce platform that distributes different functions of the application to different computers in its network.

Data: data means “that what is given”, digital data are coded in discrete characters and analog data are represented as continuous function. Data is captured, stored, transmitted, or handled only in signal form. Every data representation requires a signal representation. Data is the serialization of information. Examples: sounds of language (data), transmitted as waves while talking (signal), JSON encoded weather report (data).

Information: data that is processed, interpreted, structured and organized in a context to make it useful information. It involves human rules and activities. It is a rule to represent thoughts. Examples: graph plot built from the titanic dataset, onset of rainfall at 8PM.

Knowledge: it is the practical understanding of a subject. It is at the highest abstraction level. It is a rule to discuss. It is something you can reason on based on information.

Signals: define a physical representation of data by well-defined space or time-based changes of physical parameters. They are the physical representation of data. Highest concreteness level. Examples: sound waves, electrical currents, optical signals of abstract alphabet characters, electromagnetic waves in an Ethernet cable.

Message: data in digital representation to be used for transmission. The structure of messages and its exchange rules are defined in communication protocols. It is a rule to transmit data. Example: MAC PDU (a bit string)

Communication: it is the share or exchange of information.

Tele-Communication : communication via technology support.

Data (Tele-)Communication: exchange of data using intangible carriers.

Intangible carriers: flows of energy, current, waves

We classify networks by their range/distance:

Distance	Coverage	Type	
1 m	Desk	PAN	Wireless keyboard and mouse
10 m	Room	LAN/SAN	
100 m	Building	LAN	
1 km	Campus	LAN	Interconnected LAN
10 km	City	MAN	
100 km	Country	WAN	
1,000 km	Continent	WAN	
10,000 km	Planet	GAN	The Internet

PAN: Personal Area Network
SAN: Storage Area Network
LAN: Local Area Network
MAN: Metropolitan Area Network
WAN: Wide Area Network
GAN: Global Area Network

Personal Area Network (PAN): The smallest and most basic type of network, a PAN is made up of a wireless modem, a computer or two phones, printers, tablets, etc., and revolves around one person in one building. These types of networks are typically found in small offices or residences

and are managed by one person or organization from a single device. It is for an individual person, 1m distance., wireless. It is an ad-hoc network. Example: a keyboard transmits keystrokes to a laptop using the Bluetooth protocol.

Storage Area Network (SAN): dedicated high-speed network that connects shared pools of storage devices to several servers, these types of networks don't rely on a LAN or WAN. Instead, they move storage resources away from the network and place them into their own high-performance network.

Local Area Network (LAN): LANs connect groups of computers and low-voltage devices together across short distances (within a building or between a group of two or three buildings in close proximity to each other) to share information and resources. Enterprises typically manage and maintain LANs. Using routers, LANs can connect to wide area networks (WANs, explained below) to rapidly and safely transfer data. The area is 10/100 meters, works in rooms. Examples: Alice prints a PDF over Wi-Fi, people connect through same Wi-Fi via ethernet.

Wireless Local Area Network (WLAN): Functioning like a LAN, WLANs make use of wireless network technology, such as Wi-Fi. Typically seen in the same types of applications as LANs, these types of networks don't require that devices rely on physical cables to connect to the network.

Metropolitan Area Network (MAN): These types of networks are larger than LANs but smaller than WANs – and incorporate elements from both types of networks. MANs span an entire geographic area (typically a town or city, but sometimes a campus). Built with fiber cables. Example: the CSG uses a network provided by SWITCH to create a backup of servers located on Irchel campus.

Wide Area Network (WAN): network that connects computers together across longer physical distances. This allows computers and low-voltage devices to be remotely connected to each other over one large network to communicate even when they're miles apart. Allows communications between continents, with fiber cables. Example: an undersea cable provides interconnection between data centers in New York and Amsterdam.

Global Area Network (GAN): network composed of different interconnected networks that cover an unlimited geographical area (Example: Internet).

Standardization: it defines the rules for data communications that are needed for interoperability of networking technologies and processes. It is a must for communication! It ensures interoperability and compatibility of communication.

De facto standards: a technology, method or product that is used so widely that it is considered a standard for a given application although it has no official status (examples: HTTP, Bluetooth). They are an informal convention.

De jure standards: a technology, method or product that has been officially endorsed for a given application. They are specified by a formal organization body. The standardization process is time-consuming and costly.

Standards allow communication between devices, without standards it would be difficult, if not impossible, to develop networks that easily share information. Standards are a must for the compatibility and interoperability of communications. Examples: ISO, ETSI, ITU-T, IP.
The Ipv4 has been standardized by the IETF (Internet Engineering Task Force) and it is the fourth version of the Internet Protocol (IP), an internet standard.

Interoperability: refers to the ability of different devices to connect and exchange information with one another.

Host: any hardware device that has the capability of permitting access to a network via a user interface, specialized software, network addresses, protocol, stack, or any other means. Some examples include computers, personal electronic devices and multifunctional devices.

Unicast/Dialog: from a single source to a single destination. Two hosts exchange data across a point-to-point link. It can also consist of multiple one-to-one connections. The message is replicated by the sender. Unicast can be “wasteful” in terms of network capacity. Example: uzh sends a letter to each enrolled student (multiple unicast transmissions, not to confuse with multicast).

Multicast: from a single source to a (specific) group of destinations. Like in a class course at the Uni. The replication of the message is done by the network/service. Example: uzh sends a letter to all computer science students whose age is between 18 and 21.

Broadcast: from a single source to all destinations. It covers all areas where signals are possibly accessible. It targets everyone (everyone in the network who is sharing the medium will receive a message, whether they want to receive it or not), the message is not sent individually nor to a subgroup. Example: Alice enters a room and starts shouting.

Anycast: from a single source to the nearest member of a group. Example: uzh sends a mail to the student whose localization is closest to the Kanzlei.

There are two different transmission schemes: serial transmission, where we can access 1 bit at a time and it is the one, we use nowadays and as second parallel transmission, which is faster but also more costly (needs a higher effort to synchronize each line). It is partially or not at all used nowadays. Only 1 receiver out of N possible receivers. The specific target is not critical/important, there are redundant receivers.

Transmission mode means how data is transferred between two devices. It is also known as a communication mode.

There are three transmission modes in Computer Networks:

1) Simplex mode: In Simplex mode, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit, the other can only receive. The simplex mode can use the entire capacity of the channel to send data in one direction. There is no uplink (transmission).
   Example: Keyboard and old monitors. The keyboard can only introduce input, the monitor can only give the output. Other examples: Fire Alarms, Sensors, Pager.
2) Half-Duplex mode: In half-duplex mode, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa. The half-duplex mode is used in cases where there is no need for communication in both directions at the same time. The entire capacity of the channel can be utilized for each direction. Both can listen and speak, but not simultaneously. Both can be the role of sender and receiver just not at the same time.
   Example: Walkie-talkie in which message is sent one at a time and messages are sent in both directions. Inter phone or some GSM[1] connections.
3) Duplex mode: In duplex mode, both stations can transmit and receive simultaneously. Signals going in one direction share the capacity of the link with signals going in another direction, this sharing can occur in two ways: either the link must contain two physically separate transmission paths, one for sending and the other for receiving or the capacity is divided between signals traveling in both directions. Duplex mode is used when communication in both directions is required all the time. Example: Telephone Network in which there is communication between two persons by a telephone line, through which both can talk and listen at the same time.

<u>Quality of Communications</u>

User requirements: adequacy (ease of use, satisfaction of needs) and subjective ratings (MOS: Mean Opinion Score).

Technical requirements: - performance (bandwidth or capacity, latency or delay),

- reliability (error tolerance, resilience, noise immunity, availability),

- security (privacy, authentication, authorization),

- safety

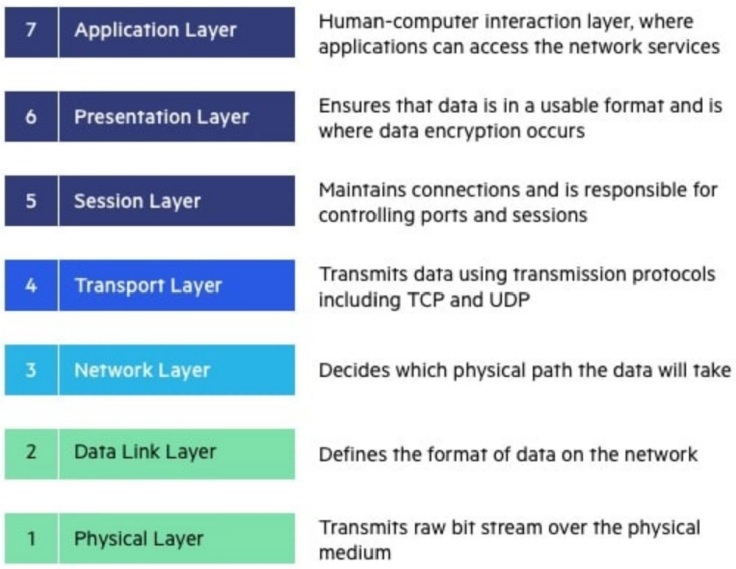
- Costs (investment, operation, maintenance costs)

Communication Networks follow a layer stack:

ISO/OSI Reference model: The Open Systems Interconnection (OSI) model describes seven layers that computer systems use to communicate over a network. It was the first standard model for network communications, adopted by all major computer and telecommunication companies in

[1] Global System for Mobile Communications

the early 1980s. The modern Internet is not based on OSI, but on the simpler TCP/IP model. However, the OSI 7-layer model is still widely used, as it helps visualize and communicate how networks operate, and helps isolate and troubleshoot networking problems.



OSI end-system: system which operates according to OSI standards.

The physical layer is responsible for transmitting bit sequencies, dealing with physical connections (e.g., cables, plugs) and getting bits on a wire.

The Data Link layer detects and corrects the error from L1. It is responsible for the transformation of the bit streams into frames.

The network layer interconnects two or more layer 1 links and breaks up segment into network packets and reassembles them on the receiving end. It also finds the best path across a physical network.

The transport layer addresses service users, and it is responsible for the flow control and congestion control. It allows the transfer between end systems (e.g., a web browser making an http request, applications, processes).

The Session layer and the Presentation layer are responsible for synchronization, session management and data syntax specification.

The Application layer (top layer) provides user services (application specific e.g., e-mails or files) and generic services (services used by other user services).

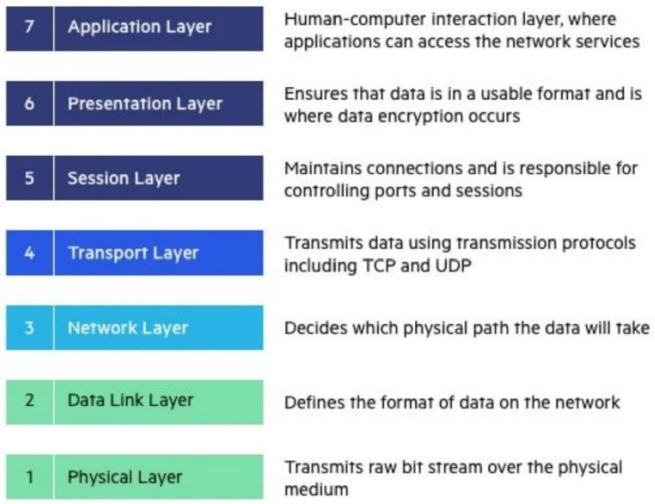
There exist many types of (N)-Entities. 2 Entities in the same layer communicate through a protocol. An Entity is for example an http sent from Google Chrome or a Linux system, we don't care much about the entity but we care about the protocol.

Connection-oriented services: a connection-oriented service is a service used to transport data at the session layer. Packets are transmitted to the receiver in the same order the sender sent them. An example is the Telephone System. A connection-oriented Service is based on a 3-phase model: the first phase is called connection setup and it is responsible for the generation of context in the network and host and for the allocation of the resources. The second phase is called data exchange and the third and last one Connection tear down, this last phase is responsible for context deletion and the de-allocation of the resources. The context being established during the connection setup phase includes addressing information. The state of the information is maintained, certain guarantees can be implemented.
An address defines a unique identification of a communication partner's host.

Connectionless services: it is a service used to transfer data from one end to another without any connection. Establishing a connection between the sender and the receiver before sending the data is not required. A connectionless service does not know any relation between any transmission service, it does not support any sequence of delivery and it does not enable any negotiation between partners. The service request includes the destination address and the service indication includes the source address (optional). Each data transfer is considered separately, and no connection status is known or even required. There is no relationship between the participants. There are no guarantees.
Examples of connectionless services: UDP, IP.

Computer Networks follow a layer stack:

1) ISO/OSI Reference model:



The physical layer is responsible for transmitting bit sequencies, dealing with physical connections (e.g., cables, plugs) and getting bits on a wire.

The Data Link layer detects and corrects the error from L1. It is responsible for the transformation of the bit streams into frames.

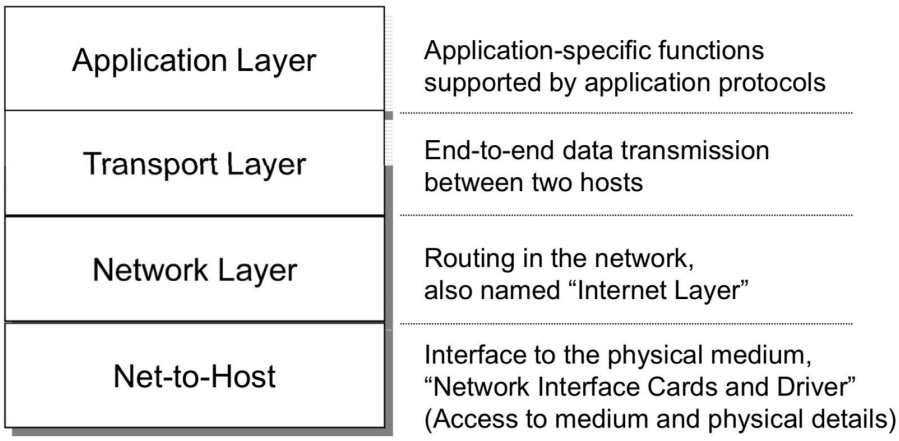
The network layer interconnects two or more layer 1 links and breaks up segment into network packets and reassembles them on the receiving end. It also finds the best path across a physical network.

The transport layer addresses service users, and it is responsible for the flow control and congestion control. It allows the transfer between end systems (e.g., a web browser making an http request, applications, processes).

The Session layer and the Presentation layer are responsible for synchronization, session management and data syntax specification.

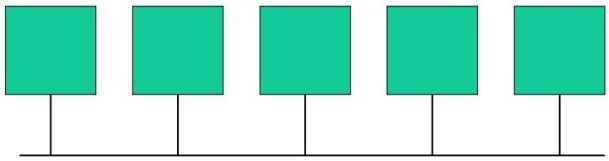
The Application layer (top layer) provides user services (application specific e.g., e-mails or files) and generic services (services used by other user services).

2) Internet (TCP/IP) model:



Major differences between the OSI and the Internet models: OSI has 7 layers whereas Internet has only 4. OSI is an independent standard and generic protocol used as a communication gateway between the network and the end user. The Internet is a communication protocol that provides connection among the hosts. The Internet model is highly used, whereas the OSI’s usage is very low. OSI provides standardization to devices like routers, motherboard, switches, and other hardware devices whereas Internet does not. The Internet provides a connection between various computers. The application layer in the Internet model includes the presentation and session layer that are present in the OSI model. The physical and data link layer in the TCP/IP model are combined.

During communications with links some bits of data are often lost. A medium is a cable that links computers to a network (wire, fiber, etc.). When talking the medium is air, waves and the modulation is done by the ears of the person you are talking to. Encoding is the process of converting the data or a given sequence of characters, symbols, alphabets etc., into a specified format, for the secured transmission of data. Modulation is the process of converting data into electrical signals (for example radio waves) optimized for transmission. It is the transformation of a source signal into a different signal representation. In Wi-Fi we have different channels into the same system.
Nodes can be source or sink: source is the node from which the message is sent, and sink is the node from which the message is received. A node is for example a Pc, router, switch or embedded system.
A channel is an abstraction of a link.
Channel properties: A channel depends on the physical medium and the mode of usage. The transmission style of a channel is either point-to-point or broadcast. A channel has a propagation delay, meaning that it takes time to receive the emitted waves (ad esempio quando guardo I fuochi del primo di Agosto vedo i fuochi prima di sentire il botto). A channel has a capacity which depends on the bandwidth and the modulation/coding used, the bandwidth is defined by the medium.
A channel also can have transmission errors, since no channel/medium is perfect, this errors consist of bit errors (where one of the bits is flipped) or error bursts (where many consecutive bits are flipped). In a channels all parameters may be time-variant. For a physical medium we always send waves and not bits or bytes. The modulated signal changes according to the signal parameters amplitude, frequency and phase.



It is hard to tell where a network ends and another one starts; networks are defined by humans. Nodes connect a network to different networks.

E2: Point-to-Point links, Resource sharing, Physical Media

Circuit switching	**Packet Switching**
Different ways how connections are established	
Connection-oriented à dedicated communication channel	Connectionless
Resource reservation	Store and forward transmission

Very reliable	More flexible and potentially more efficient
delay is always the same	delay can greatly vary, since packets are routed individually
	Data is divided into packets à-Datagrams

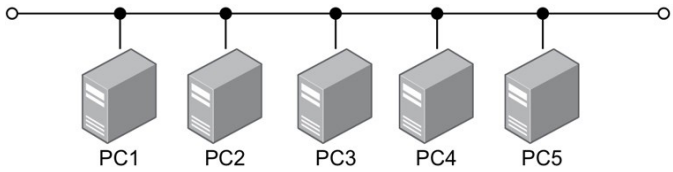
Network Topologies

Bus Topology

it is a broadcast medium where everyone can send and receive messages/data. It has the form of a bus.

+ only few cables are needed à cheap and easy to build and maintain

- limited scalability, Additional devices slow the network down and if the main cable is damaged, the network fails or splits in two, a break in the cable can so cause an entire network to collapse. This type of topology can be up to 185 meters long and a up to 30 machines can be connected.

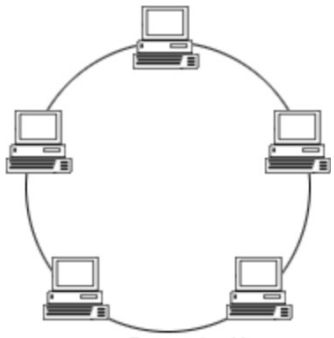


Ring Topology

Ring: consists of many point-to-point links and it has the form of a ring.

+ if cable cuts everyone can still send and receive, simplicity

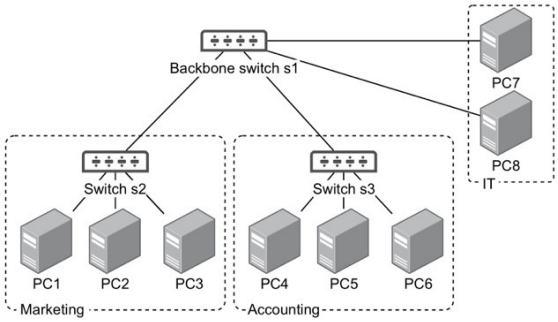
- limited scalability



Tree Topology

+ management, scalability, efficient, safe to use, cost-effective

- centralized components (network fail if the centralized node fails)

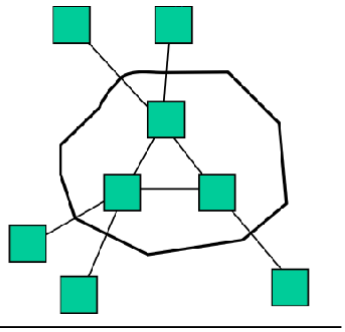


Mesh Topology

Topology used in the internet

+ fault tolerant, scalability

- complex, costly

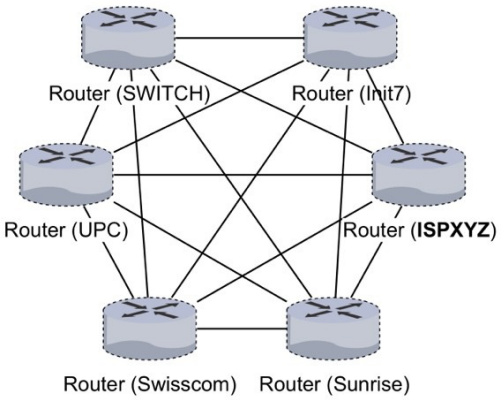


Full-Mesh Topology

Point-to-point links between all of the nodes from all nodes in the network, it guarantees that the connections are never busy.

+ highly fault-tolerant, high and consistent bandwidth (efficient), It provides multiple paths to the destination.

- complex, costly only limited scalability

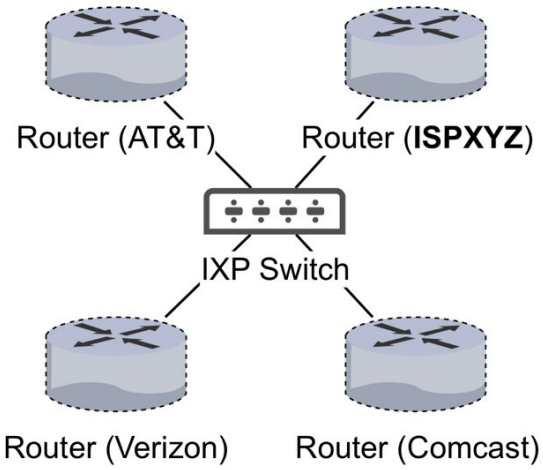


Star Topology

There is a centralized networking component and the others connected to it aside. It has the form of a star, this topology is present in-home routers or switches. all the routers are connected to a centralized switch. This switch acts as a middleware between the routers. Any router requesting for service or providing service has to first contact the switch for communication.

+ cost-effective, more scalable than bus

- Single point of failure (switch node)



Multiplexing Techniques

Multiplexing is a technique that allows network devices to communicate with each other without needing a dedicated connection between each device pair, but instead multiple signals or channels can be transmitted over a single physical medium at the same time, increasing the capacity and efficiency of the communication system.

Space-division multiplexing: type of multiplexing in which multiple communication channels or signals are transmitted over a physical medium, such as a coaxial cable or an optical fiber, by dividing the medium into different frequency bands or spatial paths.
Splitting a physical path and occupying the two-dimensional continuum of frequency and time (technique):
- Frequency-division multiplexing: each channel is assigned a frequency band, and multiple channels are combined together over a shared medium. The total bandwidth is divided into a set of frequency bands. Each of these bands is a carrier for a different signal that is generated and modulated by one of the sending devices. The quantity information is transmitted using a certain range of frequencies. FDM is a simple and effective method of multiplexing, but it requires a wide bandwidth to support multiple channels and is less efficient than some other multiplexing techniques (e.g., TDM)
Example: FM Radio.
- Time-division multiplexing: multiple signals are transmitted over the same channel in alternating time slots (transmission time is divided into time slots). It is commonly used in digital telephony to transmit multiple conversations across a common medium. The quantity information is transmitted using a certain range of time, it is hard to implement, the sender and the receiver

need to be synchronized. Each channel is assigned a unique time slot, and the channels are transmitted in a rotating sequence. TDM is efficient when the channels have uneven traffic, as it allows each channel to use the full bandwidth when needed.

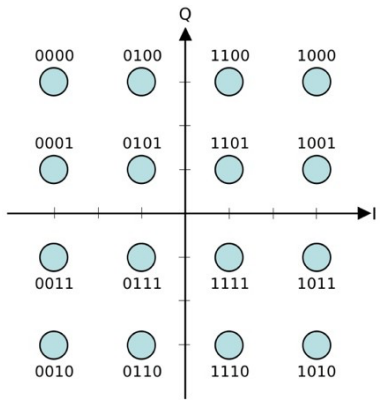
- Non-interfering (orthogonal) codes/Code-division multiplexing: multiple channels are transmitted over a single physical medium by assigning each channel a unique code. The channels are transmitted simultaneously and are distinguished by their codes at the receiving end. CDM is used in wireless communication systems and is more efficient than FDM when transmitting digital signals in a noisy environment.

Switching: Network-wide Resource Sharing
With 10 symbols we can send 20 bits (2 bits per symbol).

Modulation

Modulation refers to the transformation of a source signal into a different signal representation

Example: QAM-16



16 different symbols are possible à log(16) = 4 bits can be sent with each symbol

Data rate: r = Bs * log(n)

r data rate (bits/second)

Bs baud rate

n amount of symbols

Attention:

1 byte = 8 bit

Cables

Coaxial Cable

10-1’000 Mbit/s

200-1’000m

Fibre Cable

100-10’000 Mbit/s

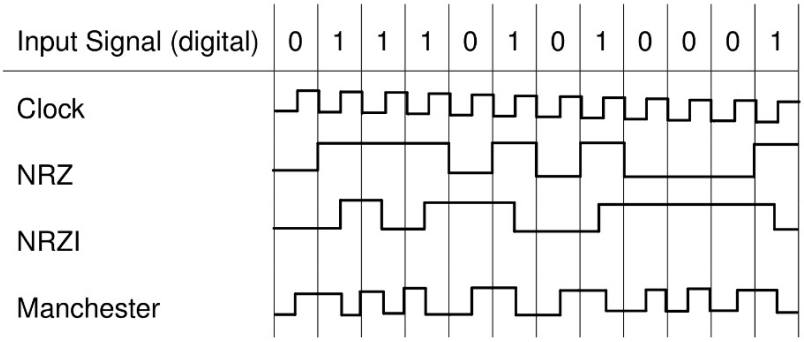
Single mode -> < 2 km

Multi-mode -> 10-100 km

E3: Reliable Transmission

Signal Encoding

Converting a given sequence of input signals into a specified format for the secured transmission of data



NRZ

NRZI

Manchester XOR between input signal and clock

Framing: point-to-point connection between two devices that consists of a wire in which data is transmitted as a stream of bits. It helps to simplify storage management and operations on the bit stream. We have to find a unit that processes at the link of sender/receiver -> frame. A frame has a well defined format, and it is different than a packet.

Layer-2 framing

Point-to-point connection between two devices that consists of a wire in which data is transmitted as a bit stream and takes place in layer 2. It is needed to simplify storage management and

operations on the bit stream (correction and detection of errors). Data (long bit streams) is broken up into frames that can easily be checked for error and corruption. It chops up a bit stream into a set of transmission units.

Approaches to provide leayer-2 framing:

- Sentinel-based framing
  type of framing that indicates a pre-defined start and ending pattern using sentinel characters. It marks the start and the end of a frame with a special marker.
  Need to escape the patterns for start and end if it occurs unintentionally in the original data unit
  à use special escape pattern that indicates, the next character is neither start nor end
- Counter-based framing
  define time length of a frame. Each frame is assigned a unique sequence number, which is used to identify the frame and ensure that it is transmitted and received correctly.

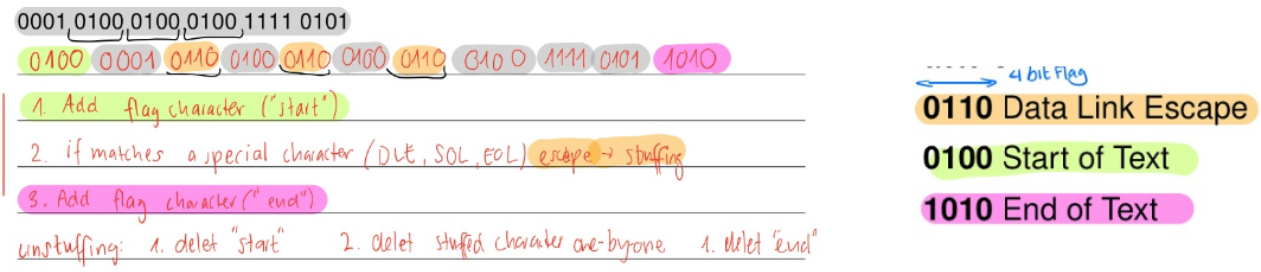
- Center-based framing
  need explicit information on number of bytes

- Clock-based framing
  a series of repetitive pulses are used to maintain a constant bit rate and keep the bits aligned in the data stream. It is used in the telephony environment and in synchronous transmission. This type of framing utilizes a pre-defined time, e.g. it looks every N seconds for a header.

General protocol functions for achieving reliability comprise:
- Sequence numbering: helps to distinguish between independent data units
- Acknowledgment: receiver has received or not what the sender sent
- Retransmission: the receiver has not received some data so the sender retransmits it
- Forward error correction
- Checksum calculation

Bit stuffing: insertion of extra charactes to recode data (after 5 0's you add a 1)

Character stuffing: first step is to add the start of the text character, then for each data character we add it to the message, and if it matches a special character (data link escape, start of text or end of text) we insert a data link escape afterwards. At the end of the text we add the end of the text character.



Sliding window mechanism: if the receiver only receives frames 1,3,4,5 the receiver acknowledges multiple times the last bit received (1) before the loss so that the sender retransmits 2,3,4,5. If he

received all frames (no frame loss or damage) then the receiver acknowledges ack(5), the highest bit received.

Cyclic Redundancy Checks are more powerful than simple (odd or even) Parity Checks because they have more bits and therefore provide more redundancy. CRC provide more information that can be used to detect errors (also burst errors).

Controls

Error Control

- detect, fix errors and order (error detection + error correction, either single or burst)
- retransmit frames
- detect and remove duplicates

Flow Control

- adapt sending rate to receiver (to avoid overflow)

Simple flow control: sender and receiver exchange two control messages besides data. Either stop, which means that the receiver is unable to cope with further incoming data, or continue, which allows the receiver to accept further data.

Congestion Control

- adapt sending rate to network utilization (to avoid congestion)

<u>odd-parity check</u>

way to detect errors

Cross sum of bits needs to be odd

If not fulfilled we know, there has to be an error à bits have flipped during transmission

If fulfilled we cannot guarantee that there was no error

Either no error occurred, or an even number of bit flipping has occurred

<u>Cyclic redundancy check</u>

More powerful way to detect errors

Based on polynomials and divisibility

Can detect more errors, if no error is detected probability is high that the transmission contains no error

<u>Forward error correction (FEC)</u>

Error detection and correction scheme

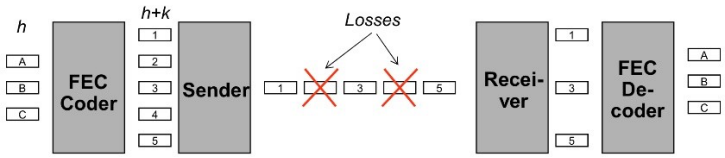
h initial data units to transmit

the FEC encoder generates h + k new units which contain redundancy

the new units are sent

on the transmission there might losses, so the receiver gets fewer units than sent

the FEC decoder can then use the units to recreate the original h data units



\+ delay is reduced in the case of an error or loss

\- additional bandwidth required for redundant data

Forward Error Correction (FEC) – Addition and Multiplication Tables

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

*	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

Sliding Window

Packets acknowledging scheme

Multiple packets are being sent by the sender

The receiver sends back acknowledgements for the highest received frame.

To indicate that a certain package is lost, the receiver acknowledges multiple times the highest consecutive frame it has received. This will make the sender re-send all the packets coming after the one that got acknowledged.

E4 - Shared Direct Links

To coordinate multiple independent senders and receivers using a shared medium for data exchange various alternatives have been discovered: distributed or centralized cooperation (somebody tells who talks first, does not work for large environments), allocation of medium on demand, constant frame lengths, variable frame lengths (e.g. reservation-based: token ring), etc.
Constant frame length: cell-based approaches (e.g. ATM).
Variable frame length: 1) Contention-based Access Methods: frames are transmitted in a time-based multiplexing scheme on a physical medium: at time t a single station must be allowed to send and if a collision occurs it must be detected. They have a co-ordination of accessing the passive medium without any centralized control. To coordinate the media without any centralized control there are 3 media states: free, busy or contented.
Reservation-based (e.g. Token Ring): Token ring network: local area network (LAN) topology that sends frames in one direction (upstream or downstream) throughout a specified number of locations by using a token (it is based on an alternating sharing concept). The token is a special bit pattern that rotates around the ring. This token has to be captured by the hosts before being

allowed to send data. Hosts then release the token after sending data and other hosts that wat to transmit data capture it. It prevents collisions.

Important features of access algorithms: performance, fairness (= average/maximum time to access the medium (should be evenly distributed), fairness depends on configuration and traffic).

ALOHA protocol: multiple access protocol (with variable frame length) for transmission of data via a shared network channel. It is the simplest multi access control mechanism, where no state check of the medium is required and there is no coordination. It contains a frame with a checksum, for which error correction is possible. Sends packets whenever ready, receiver acknowledges "good" frames, and the sender retransmits some if no acknowledgment is immediately returned. Using this protocol, several data streams originating from multiple nodes are transferred through a multi-point transmission channel.
Vulnerability for ALOHA needs to be as smallest as possible (ideally 1 bit only) so that we stop transmitting asap when a collision occurs.
Slotted ALOHA: it reduces the number of collisions and doubles the capacity of ALOHA. The shared channel is divided into a number of time intervals called slots. Sending permissions are only allowed for slot starts, which means that the collisions are restricted to the beginning of a second. In ALOHA devices transmit whenever they want (transmission is ready), in Slotted ALOHA they have to wait for (the beginning of) an available slot to send data.
Carrier Sensing Multiple Access (CSMA): when a station has frames to transmit, CSMA detects the presence of the carrier signal from other signal from other nodes connected to a shared channel -> signal channel and devices wait until it is free to send data. If a carrier signal is detected, it means that a transmission is in progress.
From the physical layer we need the guarantee that collisions are detectable in order to be able to use CSMA/CD in a sensible way. While sending data, there is a need to see if there is another transmission in progress.
CSMA/CD also sends and monitor, if there is a collision it waits. CSMA if effective before a collision, CSMA/CD is effective after a collision. When a collision occurs, in CSMA/CD the transmission is topped and a jam signal is sent by the stations and then the station waits for a random time before retransmission. CSMA/CD has an increased throughput for many attempts.

Knowing the host address and the destination address is enough to decide if the host will accept the frame (we don't need to know the host's internal state).
10- Gigabit Ethernet transmits 10 times faster than the Gigabit Ethernet. Gigabit Ethernet works on half-duplex and full-duplex mode, whereas 10-Gigabit Ethernet only operates in full-duplex mode. In 10-Gigabit Ethernet there is no more CSMA/CD, it consists of a collisions free star topology.
Ethernet is a shared medium.
Ethernet MAC addresses:

Unique addresses:

a) 48-bit unicast address assigned to each physical adapter (example: 8:0:2b:e4:b1:2)
b) Broadcast address (all 1s). The highest address in range is the broadcast address.

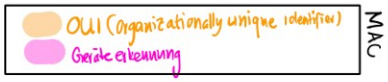
c) Multicast address (first bit is 1)

Adapter receives always all frames, but only accepts:

a) Frames addressed to its own unicast address
b) Frames addressed to its own broadcast address
c) Frames addressed to any multicast address it has been programmed to accept

Ethernet MAC addresses (shared medium)

- unique addresses
  - unicast address -> 48-bit assigned to each physical adapter
    14:10:9f:e3:d2:cd
  - broadcast address
    ff:ff:ff:ff:ff:ff (all 1s)
  - multicast address
    01:00:00:00:00:00 (fist bit is 1)



last fast Ethernet technology

Fast Ethernet	Gigabit Ethernet	10-Gigabit Ethernet
CSMA/CD + half duplex	CSMA/CD + full duplex	Full duplex only → no Ethernet anymore
10x faster than normal Ethernet	Leveraged Fibre Channel PMDs	New optical PMDs
8B/6T encoding	Reused 8B/10B coding	New coding schemes 64B/66B
not more than 4 hubs between any station in network	easy/cheap upgrade reliable cable length limited up to 100m	collision-free frame compatibility star topology point-to-point
support LAN to 412m	Support LAN to 5 km	Support LAN to 40 km

- Ethernet like IPv6, does know any broadcast addresses
- any MAC address **receives always all frams**
- any MAC address **recieves** from host, when there is a **physical connection (cable) between the two hosts**
- every host **accept** braodcast frames
- Multicast frames are **not stateless**
- MAC address **accept multicast frames if part of group**
- MAC address **accept unicast frames if host is destination** (same OUI)
- MAc address **accept all frames if programmed to accept anything**

E5 - Routing and Switching

Virtual Switching vs Packet Switching

	Virtual Switching	**Packet (Datagram) Switching**
Transmission	Connection-Oriented	Connectionless
Delays	Connection Setup + Forwarding Delay	Only Forwarding Delay
Addressing	Circuit ID	Source, Destination
Path	Fixed ordering	Dynamic -> ordering might not be the same

- It is not guaranteed that a new circuit can be opened at any time. If there are too many already established connections a new one might be denied. -> Once a circuit is established a certain bandwidth is guaranteed.
- Message switching follows a store-and-forward switching operation -> messages are received in full before being forwarded
- Message switching is not similar to circuit switching in terms of addressing. Message switching the address includes: Source, destination and the message length. While the Circuit Switching doesn't include a header because the exact path is determined and established beforehand
- Packet switching doesn't follow connection-oriented transmission since there is no setup phase

- In a circuit switched network there are never more established connections than the network can support -> otherwise trunk allocation denied
- In a circuit switched network you don't have ordering issues because there is a stable, given transmission path where the bits are sent sequentially on a fixed bandwidth
- It is possible to ensure ordering of the packets sent on a packet switched network, implementing a virtual switched network where there is a sequence ID. This is used by TCP.

Packet switching advantage vs Circuit Switching: **More flexible** -> allows for **more data types** like voice, video, email and it is more efficient -> **don't always need to have connection up** and running

When the **delay** of a message is **most important** (e.g. door sensor once a day message) then choose **message** or **packet switching** since no setup delay

If the **network must guarantee a certain number of connections** (e.g. emergency purposes) then **circuit** or **virtual circuit switching** is the best. Since then you can deny connection allocation for random calls if the network is almost at max capacity to ensure that emergency calls can still go through.

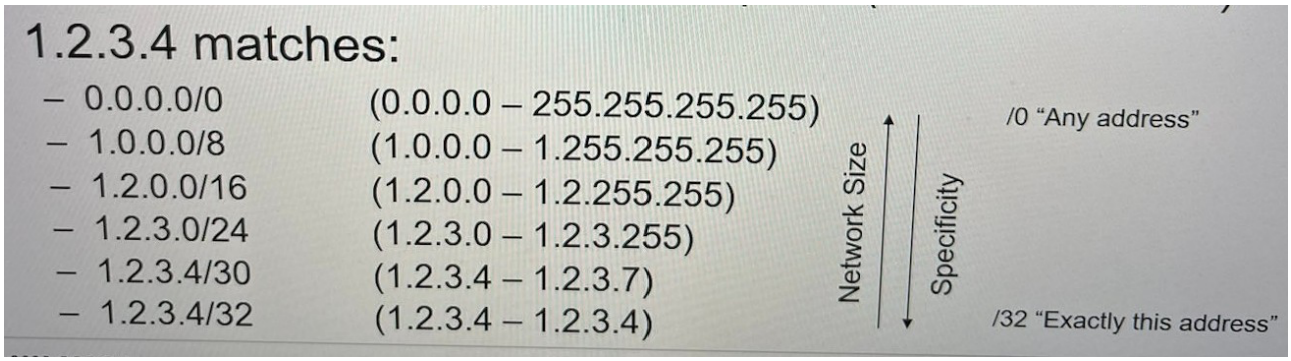
Device	**OSI-Layer**	Comment
Small Switch which you can connect using Ethernet cable	Data Link Layer (L2)	Switch = Powerful transparent bridge with high port density
Regenerating/Amplifying **signal device**	Physical Layer (L1)	
Device which **extends signal from fiber link to copper-based** link using **same L2** protocol	L1	Because Layer 2 has to be identical
Hardware Firewall with allowlist (looks at **MAC address**)	L2	The MAC addresses are used to differentiate who is allowed and who isn't
Hardware Firewall which inspects IP Addresses of packets (because MAC address can be changed)	L3	Because IP addresses (L3) are used to differentiate
Device which **extends signal from fiber link to copper-based** link **using different L2** protocol	L2	2-Port Bridge/Multiport ->Bridge which interconnects two or multiple networks
Hub vs Switch	Hub: L1, Switch: L2	Hub is a repeater with multiple ports that repeats signal on all links Switch examines Layer 2 frames (MAC addresses) and sends them to most appropriate link

Key property a spanning tree algorithm establishes for a network: It ensures that the network doesn't have any loops -> makes the network loop free -> prevents "broadcast storms" (infinite packet replication)

Network Type	**Number of Devices** that can be addresses	**Subnet mask**	**Lowest** possible **address** and **highest** possible **address**
1.2.3.0/24	32 bits – 24 bits = 8 bits --> 2^8 addresses = 256 2 "Unusable": Network address: 1.2.3.0 and Broadcast address: 1.2.3.255 Usable 256-2=254	11111111.11111111.11111111.00000000 → 255.255.255.0 Needed to determine if an IP address belongs to the network Make an AND between Subnet Mask and IP address	Lowest: 1.2.3.0 Highest: 1.2.3.255
4.5.0/16	32 bits -16 bits = 16 bits →2^16 addresses = 65536 65536 – 2 (useless) = 65534	11111111.11111111.00000000.00000000 →255.255.0.0	Lowest: 4.5.0.0 Highest: 4.5.255.255

In a routing table there is a Destination, Gateway and Interface.

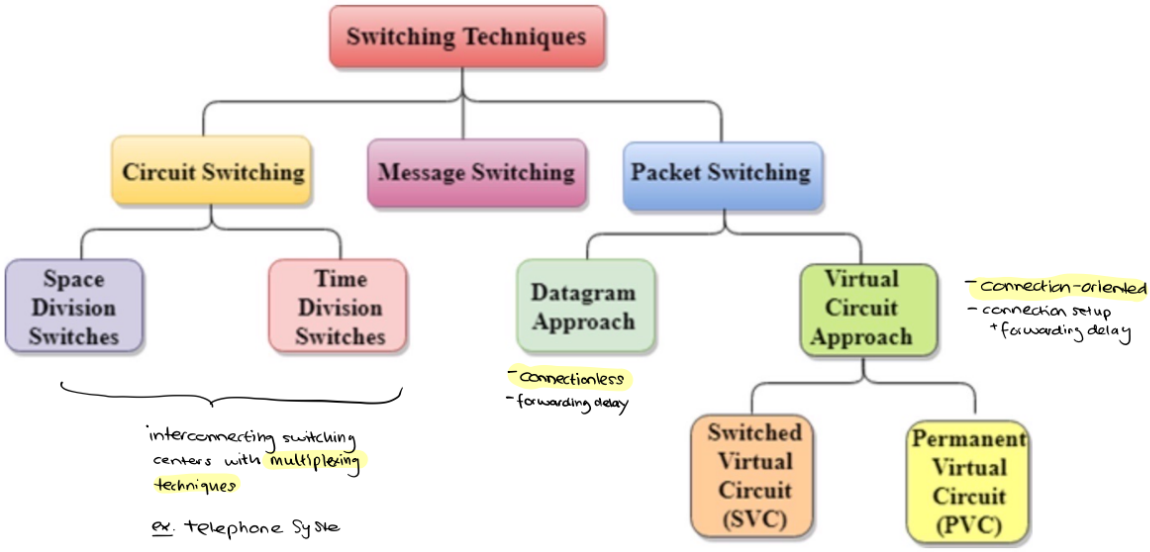
- The **Destination** is where the incoming IP packet wants to go. First the router checks for all matches this is done by doing an AND with the subnet mask of each destination of the routing table -> if the result from that equals the destination address in the router it is a match. The default destination is always a match since its address range goes from 0.0.0.0 – 255.255.255.255
- The **Gateway** is the IP address where the packet is sent next it has to be in the same address range as the destination: e.g. 1.2.3.1, or link#3 -> the destination is directly attached so there is not other router in between
- The **Interface** is where physically on the router the packet is sent: ethernet0, ethernet1…
- Now the **best match** needs to be **determined**. The best match is the one with the longest prefix == smallest network for example 1.2.3.4/30 is more specific than 1.2.3.0/24 as seen in the picture below
- It is usually **not good** to have **multiple default destinations** because then the destination is chosen randomly which can lead to errors and performance issues.



E6 - Routing and End-to-end protocols

Routing: process of path selection in a network.

Switching: process of forwarding packets from one host to another and helps deciding the best route for data transmission if there are multiple paths in a network.



Circuit Switching: it is the dominant switching technology for the public telephone network. The transmission/propagation delay in this circuit is determined by physical transmission delay. All bits are sent sequentially, and the bandwidth is fixed. The sequence of the package sent is unchanged, fully digital switching is possible. management of end-to-end circuits with fixed resources. It is related to the phone where the full path is blocked and the call can't be interrupted, but only the signal can be disturbed. Circuit switching is connection oriented. Data is transferred through 3 different phases. Transmission of data done only by the source. In circuit switching there is a dedicated communication channel through the network and fixed resources (e.g., route, bandwidth) which has its advantages but the network bandwidth might also be unused. In Circuit Switching, clients are not guaranteed that they can open a new circuit at any time. The quality-of-service guarantees require fixed allocations (trunk allocation may fail). Circuit switching does not have a header (path is established beforehand with signaling). There are no transmission units. In a circuit switched network, there are never more stablished connections than the network can support (trunk allocation would be denied if network reaches capacity). Network is never overloaded (congestion is avoided). In a circuit switched network there are no ordering issues (stable transmission paths, all bits are sent sequentially in order).

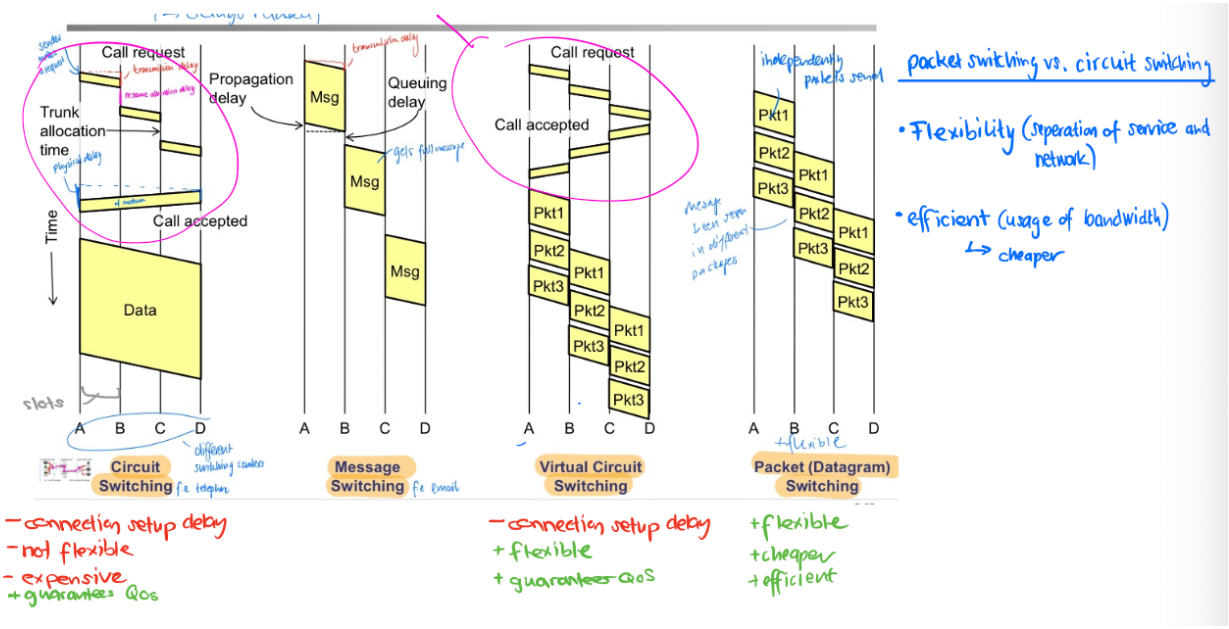
Message Switching: it is implemented on top of packet switching or circuit switching networks. Messages are received in full before forwarded. The content of a message is a header including

the source, destination and length of the message. Then also the message text is included in the content of a message. Message-switching follows a store-and-forward switching operation. It has a header which includes source, destination and length information. Management of messages, similar to e-mail (can stay on uzh server for undetermined amount of time). Also has connection-setup delay.

Virtual Circuit (VC) Switching: enables transmissions over packet-switched networks. The path established between two devices appears as a dedicated physical link. Virtual Circuits are set-up

through signaling protocol: all packets follow the same path and contain a virtual circuit ID, the path context is available in all intermediate nodes. Connection-oriented, delay occurs in connection setup and forwarding delay. Addressing is based on the Circuit ID. Virtual Circuits can be implemented on top of a packet-switched network.

Packet (Datagram) Switching: Data is divided into packets and packets are routed individually (datagrams and virtual circuits, for the same connection we can send multiple messages only with different paths). Content of packets: header including switching and length information, switching based source/destination address or virtual circuit ID, packet content. There is no connection setup, we don't know how fast the message is sent (if the network is busy there is a waiting time). Packet switching is connectionless (there is no connection set-up phase). The transfers of data take place directly. Transmission of data done by the source and the intermediate routers. The quality of services guarantees may be lost. Packet switching has forwarding delay. Addresses source and destination, and path is dynamic.



Fiber optical carriers/mediums work better/faster than copper cables over long ranges but in short ranges the copper cables are faster than the fiber optical ones in terms of signal propagation delay.
To avoid connection-setup delay choose between packet switching or message switching.
Queueing delay is present in all different switching methods.
Quality of service guarantees can be easily implemented for circuit switching or virtual circuit switching (e.g. deny trunk allocation if <10 slots).

Layer	Device
Application	Application gateway, *e.g.,* mail forwarder, firewall
Transport	Transport gateway, firewall
Network	Router
Data link (MAC)	Bridge, switch
Physical	Repeater, hub

Medium Access Control (MAC) addresses: address that identifies a device to other devices on the same local network. Any port of communicating devices is equipped with a unique identifier (address) assigned to all network interfaces. The MAC address is only valid within the interface and it is often assigned by the manufacturer.

Minimum Spanning Tree: consists of a set of links connecting all vertices (without any cycles), where each link/edge has a cost (weighted graph), and the sum of the link costs is minimized. The graph is weighted, connected and either directed or undirected. For a graph, more than one MST might exist.
Shortest path tree: set of links connecting all vertices from the root (starting node) to any other node in the graph is the shortest path. The shortest path is the path with the minimum cost. The shortest path tree is not necessarily a minimum spanning tree.

Dijkstra's Algorithm, procedure:

a) Apply infinite cost to every node apart from the starting node/vertex
b) Keep track of the starting node
c) Follow the edges to reachable (adjacent) nodes, and if the cost to reach a node is lower update its cost. Repeat till are nodes are visited.

Distance Vector Routing: simplifies the routing process by assuming the cost of every link is one unit. Therefore, the efficiency of transmission can be measured by the number of links to reach the destination. In Distance vector routing, the cost is based on hop count. The distance-vector routing (DVR) is designed to periodically update the routing data in the network model based on the Bellman-Ford algorithm. Every node in distance vectors has only its local perspective available (only sees its neighborhood). Distance vectors suffer from count-to-infinity problem: it is a routing loop and occurs when an interface goes down or two routers send updates to each other at the same time (misleading routing information into the routing table). The solution is that each router needs to know the entire topology, but it uses more memory (costly) and has redundant information. Distance vector routing protocols react fast to good news but slow to bad news.
To solve this problem the entire topology would need to be know -> Link State Routing.
Link State Routing: each router distributes/shares knowledge of its neighborhood (local view) to every other router in the network. Each router maintains a complete view of the topology. Advantages of link state routing: fast convergence in case of changes in the network (a single link-state update is sufficient), doesn't suffer from the count-to-infinity problem (when there are loops), it has a high stability since every router has a complete view of the network.

An autonomous System (AS) is a single routing domain and corresponds to an administrative domain showing a unique number (ID). There are three different types of Autonomous Systems:

a) Stub AS: connection to only one other AS (for local traffic)
b) Multi-homed AS: connection to more than one other AS (refuses carriage of transit traffic)
c) Transit AS: connection to more than one other AS (for local and transit traffic)

Examples of AS: university, company, or backbone network (part of network which interconnects networks).

Transport address in TCP/IP: consists of an IP address and port number, an end host X and a service Y, it is globally unique. Each port number within a particular device identifies a particular software process. A transport address addresses an application-layer process running in an end-host.
User Datagram Protocol (UDP): connection-less, unreliable and unordered datagram service. Message-based (containing source, destination and data), including multicasts, simple multiplexing and no flow control. Endpoints are identified by ports (servers have well-known ports). It has optional checksum (error control but no error correction). In UDP there is no guarantee of ordering. No retransmission and no acknowledgments. Used for DNS, streaming. Delay is more important than every bit/segment being correct (focus on delay).
Checksum is a value that represents the number of bits in a transmission message and is used to detect errors within data transmissions.
Transmission Control Protocol (TCP): connection-oriented, reliable, connection between two sockets (endpoint for sending or receiving data in a computer network , more specifically endpoint of a communication link between two application ports) in full duplex mode.
TCP has error control, flow control and congestion control. The data transfer is byte-stream driven: sending process writes some number of bytes, TCP breaks into segments (TCP packets) and sends via IP, receiving process reads some number of bytes. Has retransmission, acknowledgments and ordering guarantees. Used in HTTPS, FTP. Every small error could lead to a very big lost (focus on errors).
TCP Header: consists of source and destination port (part of the transport layer address), sequence number and acknowledgment, tcp header length, flags, advertised window (sending window of flow control), urgent pointer (pointer to important data).
TCP Connection Setup:

a) Opening of a socket:
    1) Active mode: request of TCP connection from a specific socket
    2) Passive mode: user informs TCP being able to accept an incoming connection (listen/accept), it announces its willingness to be open.
b) Functional view: 3-way-handshake and exchange of initial sequence numbers (avoids conflicts with old connections that have used the same port).

	TCP	UDP
Connection	Connection-oriented	Connection-less
Retransmission	✓	×
Flow Control	✓	×
Acknowledgements	✓	×
Ordering Guarantees	✓	×
Use Cases	HTTPS, FTP "If one bit is wrong – everything is useless"	DNS, Streaming "Delay is more important than every bit/segment being correct"

Congestion control: mechanism that controls the entry of data packets into a network, and it is used to avoid overloading and congesting it. It adapts the sending rate by dropping packets. Its goal is to keep the sender from overriding the network.
TCP congestion control: addresses the queuing situation in routers, congestion leads to retransmissions in the transport layer (this leads to an increase in the congestion), congestion control needs to adapt sending rate(s).
Approaches for TCP congestion control:

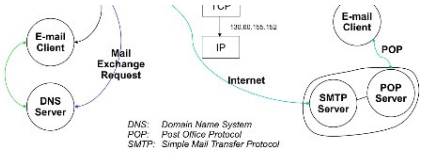
a) Implicit Mechanisms: increase of sending credit after receipt of an ACK, decrease of sending credit upon lack of an ACK.
b) Explicit Mechanisms: explicit congestion notification.

Window-based congestion control, 2 Phases:
Window-based Congestion Control 1 : the window defines the number of packets allowed to be sent upon receipt of an ACK. Advantage: small overhead due to ACKs. Drawbacks: bursty nature, loss of ACKs. Small start (small window size), then exponential growth until given threshold. If congestion is starting to happen, restart with slow start.

Window-based congestion control 2: utilizes sliding window control, advantages: not bursty and loss of ACKs is bearable. Drawbacks: large overhead due to many ACKs and there is interdependency between error control and window size.
Additive increase: Linear growth (e.g. increase window size by 1). Multiplicative decrease: e.g. half the window, and adapt the threshold after packet loss.
If congestion is starting to happen, restart with slow start (phase 1).
Window-based congestion control is implemented on end-hosts, not on network.
If congestion is starting to happen, restart with slow start (phase 1).
Window-based congestion control is implemented on end-hosts, not on network.

Link Layer Protocols	Transport Protocols
Entities are connected via a single point-to-point link	Entities do not reside in neighbor systems
Mostly fixed and small Round Trip Times (RTT)	Mainly variable and much larger RTTs, leading to a larger bandwidth-delay product
No change of packet sequence→order stays the same	Ordered delivery not guaranteed
Link limits sending rate	Possible bottleneck may exist on any intermediate link

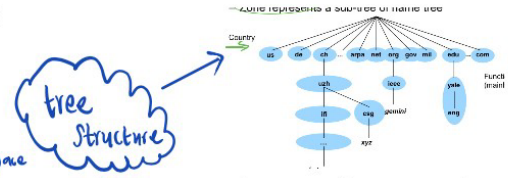
E7 – Applications and Mechanisms



- Simple Mail Transfer Protocol (**SMTP**):
  - exchange of messages (ASCII only) between client & server
  - Usecase: Electronic Mail
    - international exchange of electronic messages
    - support of asynchronous sender and reciever
    - store-and-forward switching (Packet/Message switching)
    - Protocols: SMTP , DNS
  - max. message size 64'000 Byte
  - Content:
    - Envelope - addressing based on DNS (name@acm.org)
    - Header - additional field (subject, CC)
    - Body - contains message
  - uses direct TCP connenctions -> makes it so reliable
  - SMTP does not have a built-in method for authenticating an email address
    - „from" - filed is not authenticated
    - Solution for security: DMARC, SMTP AUTHentication
  - Extension: **MIME** (exchanging different kinds of data - vidoe,audio,images,special characters)
    - Encoding, content-type, boundaries
  - Client operation: POP3, IMAP, HTTP

- Internet Mail Access Protocol (**IMAP**):
  - accessing e-mails by using a connection set-up

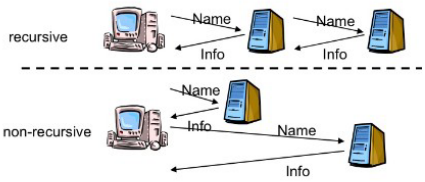
- Domain Name System(**DNS**):
  - DNS resolves name authority and the implementation of a distributed system to map names onto IP addresses -> makes IP addresses human readable
  - DNS supports IPv4 and IPv6 but does not assume TCP/IP (underlying networe x IP-based)
  - addresses are fixed length -> easy for computers
  - names are variable length -> easy for humans
  - ***name spaces*** -> defines set of possible names
    - **flat name space**
    - **hierarchical name space** → better than flat name space



      - separates domains by zones, which are represented as a sub-ree of a name tree
      - every **zone** maintains:
        - a subset of its domain it belongs to, means every NS knows those IP adresses of all NS in the directly attached sub-domains
        - 1 NS (name server)
        - at least 2 secondary NS

each → a delegation is responsible for a limited part of it

      - hosts need to know the IP addresses of the top-level name servers
  - ***name resolution*** -> starts at tree root, maps names onto information
    - **recursive DNS lookup**
      - one DNS server communicates with serveral other DNS servers to resolve IP address for client
      - \- higer performance demand on each name server
    - **iterative DNS lookup**
      - client communicats directly with each DNS server
      - \- caching is only client-side, if request same name name, resolution has to be done again



  - DNS packet format
    - uses simple *UDP* since it is efficent (no connection set-up or reliability required)
    - Requests -> contains DNS name and type of request
    - DNS protocol allows multiple queries

		12	variable	variable	variable	variable	[byte]
IP	UDP	DNS-Header	Requests	Reply RRs	Authority-RRs	Additional RRs	

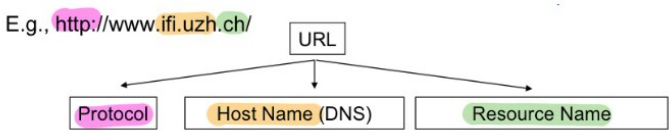
  - Extension:
    - **DNSSEC** (DNS Security)
      - adds layers on top of DNS to make it verifiable
      - the protocol **ensures integrity of DNS,** with cryptographic authentication of data
      - an intermediary can read but not modify DNS requests
    - **DoT** (DNS over TLS)
      - DNS is added over TLS (Transport Layer Security) and **ensures confidentiality**
      - **TLS:**
        - located above transport layer (L4) because it runs on top of TCP (on L4)
        - responsible for ensuring that functional data is being transferred (integrity)
        - TLS uses handshake (session are set up)
    - **DoH** (DNS over HTTP)
      - DNS is added over HTTP and **provides confidentiality,** and **avoids port-based detection**

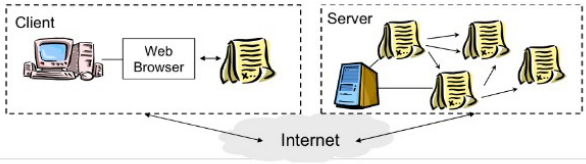
HTTP:

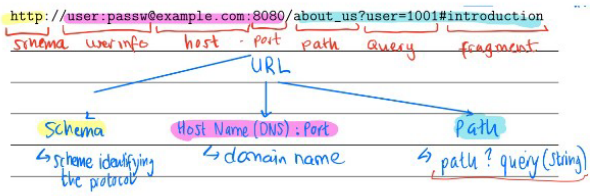
Request (Status) Line	General Header	Request (Response) Header	Entity Header	Entity Body

ASCII-based application with default port 80

- World Wide Web (**WWW**)
  - acts as an information platform and ensures simpe world-wide exchange of documents
  - command-line based
  - *client/server-based architeture* -> hyperlink enable navigation by addressing a **web page** uniquely
  - **„web page“** = **URL** (Uniform resource Locator)
    - determins the resource‘s location as a name



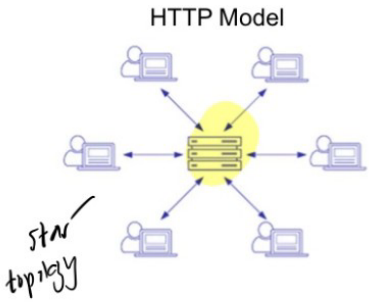





```
<scheme>:<scheme-specific-part>
    ftp://<user>:<password>@<host>:<cwd1>..
          <cwdN>/<name>;type=<typecode>
   http://<host>:<port>/<path>?<searchpart>
  mailto:<rfc822-addr-spec>
    nntp://<host>:<port>/
          <newsgroup-name>/<article-number>
  telnet://<user>:<password>@<host>:<port>
    file://<host>/path
```

- %20 == space

  - **HTTP**
    - Backbone protocol of the Internet -> centralized
    - HTTP uses TCP connection
    - why did HTTP had a rapid growth?
      - simplicity
      - stateless
      - infeasible to block port 80, 443
      - text-based with support for binary data
      - established encrypting
    - simplicity of HTTP protocol made it very rapid growth
    - ASCII- base application protocol
    - Response: protocol veresion, ersponse status code, (optional: header, message body)
    - Extension: **HTTPS** (more secured version of HTTP)



  - **HTML** (Hypertext Markup Language)
    - text options:
      - start <…>, end </…>
      - paragraph <p>
      - new line
      - header <hn>
      - italic <i>, bold <b>, emphasized ,
    - alternative text style:
      - CSS, JavaScript, Browser Settings

```
<html>
 <head>
 <title> Document Title </title>
 </head>
 <body>
 <p>This is a HTML document.</p>
 </body>
</html>
```

other application protocols

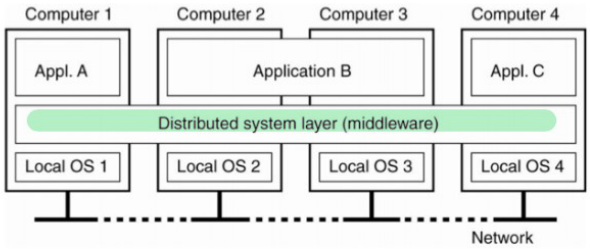
- **FTP** - transferring files
- **SSH** - securely accesing remote systems
- **FTP** - transferring files
- **SNMP** - retriving device information for network management *****

E8 – Distributed Systems – Introduction

Distributed System =a collection of independent computers that appears to its users as a single coherent system (=Transparency)

Characteristics:

- Autonomous components
- Users think it's a single system (=Transparaency)
- Communication between components mostly hidden from users
- Users can interact in a consistent and uniform way, regardless of where and when interaction takes place -> makes resources easily accessible
- Reliability & High Availability (Failure Handling), Openness
- Placed in disttributed system layer-> Sometimes called **middleware**
- Consequences: global clock , cuncurrency independent failures



- provides similar interface to all participants

Key challanges in DS:

1. Transparency (Transparent)
   - Single view of the system
   - Hide numerous details
2. Heterogeneity (Heterogenität)
   - Networks
   - Computers (HW)
   - Operating systems
   - Programming languages
   - Developers
3. Failure Handling (Fehlerbehebung)
   - Detecting
   - Masking
   - Tolerating
   - Recovery
   - Redundancy (Redundanz)
4. Openness (Offenheit)
   - Extensibility -> how to develop further and further
   - Publication of interfaces
5. Scalability (Skalierbarkeit)
   - Controlling the cost of resources
   - Controlling the performance
   - Preventing resources from running out replication technique: caching
   - Avoiding performance bottlenecks (Engpass)
6. Security (Sicherheit)
   - Secrecy
   - Authentication
   - Authorization
   - Non-repudiation

1. Transparency

Transparency	Description	Example
Access Transparency	Enables local/remote resources to be accessed using identical operations	Different data representations (4 or 1 decimals) Different conventions (Linux: case-sensitive, Windows: not)
Location Transparency	Enables resources to be accessed without knowledge of their physical location or network location	Access to Web page without knwoing it's IP
Concurrency Transparency	Enables several processes to operate concurrently (gleichzeitig) using shared resources without interference between them	
Replication Transparency	Enables multiple instances of resources to be used to increase reliability and performance	Gmail, Facebook maintain their data transparently Replicated for increased availability and improve access time
Failure Transparency	Allows user/application to complete task despite the failure of hardware or software components	
Mobility Transparency	Allows movement of resources and clients within a system without affecting the operation	
Performance Transparency	Allows system to be reconfigured to improve performance as load varies	
Scaling Transparency	Allows system /application to expand in scale without change to the system/application structure	

2. Heterogeneity

-> Hetrogeneity addresses transparently differences in:

- Performance
- Capabilities
- Network connectivity

3. Failure Handling

-> ensure **high reliability and availability**

Class of Failure	Affects	Description
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process' s incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behavior: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

4. Oppeness

- Goal: allow components to be easily added or replaced **(Extensibility)**
- well defined interface
  - clearly ducumented (publicly)
  - standardized
  - well designed
  - using API (Application programming interfaces)
- use of open IDL
  - boosting **interoperability, portability and extensibility**

5. Scalability

- Scalable with respect to:
  - Size: Easily add more users and resources
  - Geography: Users and resources may lie far apart
  - Administration: Easy to manage if it spans many administrative organizations
- Problem: most systems suffer from performance loss when scaling
- Solution: storing data in a distributed manner might reduce overall data volume communicated

Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

- Techniques:
  - Distribution of responsibilities -> split data on multiple computers
    - ex. DNS -> divide into zones
  - Hide communication latencies -> avoid waiting for responses
    - ex. differenciate between server and client ceck
  - Apply replication techniques -> copy data onto multiple servers
    - ex. Caching

6. Security

- secure systems against attacks
  - viruses, worms, DDoS, cybercrimes
- Countermeasures
  - Authentication, authorization, encryption, non-repudiation
- „weakest link“ principle -> security measures are defined by a risk assessment

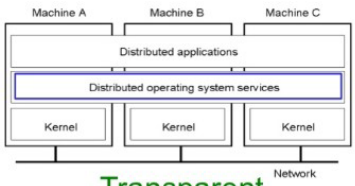
Hardware Architectures

- **Bus-based multiprocessor**
  - all processors share same bus to access memory
  - using caches to prevent bus saturation -Y consistency problem
  - \+ simple, cheap
  - \- not scalable
- **Switch-based multiprocessor**
  - concurrent memory access by multiple processors
  - \+ increased concurrency -> more speedy
  - \- delay, expensive, fast crosspoint switches

Software Architectures

System	*Description*	*Main Goal*
Distributed Operating System (DOS)	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
Network OS (NOS)	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

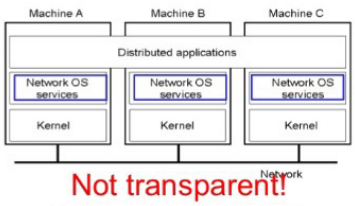
Distributed OS



Transparent

Non-autonomous computers!

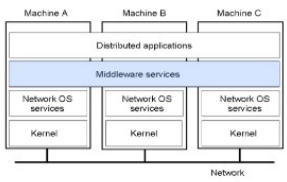
Network OS



Not transparent!

Autonomous computers

Middleware



Combination of the advantages, while omitting the drawbacks.

Middleware

- provides a similar interface to all users in a open system
  - ensures interoperability
- Openness, transparency, (scalability)

	Distributed OS	**Networ OS**	**Middleware**
Transparency	yes	no	yes
Openness	closed	open	open
Scalability	no	yes	varies
Same Os on all nodes	yes	no	no
Resource management	Global, central	Per node	Per node

What are advantages of Network Operating Systems?

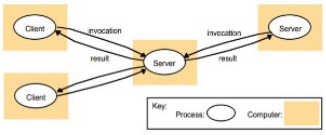
- [✓] Centralized servers are stable
- [✗] Cheap implementation
- [✓] Remote access
- [✗] Low maintenance

What are disadvantages of Network Operating Systems?

- [✓] Costly
- [✓] High maintenance
- [✓] Hard to integrate
- [✓] Dependency for most operations

Network Interactions

- **Client/Server**
  - request-respone pattern (client send request, server responses)
  - common protocol needed to enable communication
    - interoperability between two nodes
  - Client: User interface/ application
  - Server: Application/ Database



- **3-Tier network application**
  - client/server software architecture pattern with multiple levels
    - 1. user interface level
    - 2. processing level
    - 3. data level (storage and access)
  - modules developed and maintained independently (separated platforms)

- **Cluster of servers**
  - set of cooperating interconnected computers
  - high availability, high performance, mass storage, richer application services
  - goal: reduce downtime and outages -> allwoing another server to take over in event of outage
  - ex. Google

- **Multi-tiered architecture**
  - scaled more complex than 3.tier architechture
  - depending on functioons that will be performed

- **Web proxy server**
- **Code mobility**
- **Thin clients**
- **Overlay networks**

Intranet	Internet
Private network	Public network
Limited users	Unlimited users
More Secure	Less Secure
Mostly LAN	LAN, WAN, MAN etc.
Has owner	Does't have an owner

Examples for DS:

- **Client-Server Communication**
- **Itranet**
- **Internet**
- **Automation Networs**
  - focus on reliability
  - 3 layers
- **Personal Networks**
- **Peer-to-Peer Systems**
- **Cloud Computing**
  - implemented on cluster computers
  - on-demand delivery of computing services
  - set of Internet-based application service, storage service and computational services

E9 – Naming

Distributed System: a collection of independent computers that appears to its user as a single coherent system, interconnected devices feel as one to the users.

Consequences:

a) Concurrency: multiple tasks which start, run, and complete in overlapping time periods, in no specific order
b) No global clock (common, shared timing source among the participants.)
c) Independent failures

Examples of distributed systems: intranet, internet, personal networks, automation networks, peer-to-peer networks, client-server communications, cloud computing.
Client-server communication: uses a request-response pattern, client sends a request, then the server sends a response. Communication is always initiated by client and served by server. A common protocol for clients and servers is needed for communication to ensure interoperability, to communicate.
Intranet is a private computer network which shares information, collaboration tools, operational systems, computing services withing an organization (no access to outsiders). Internet is a public network accessible to anyone. The internet protocol suite enables intranet connectivity.

Intranet	Internet
Private network	Public network
Limited users	Unlimited users
More Secure	Less Secure
Mostly LAN	LAN, WAN, MAN etc.
Has owner	Does't have an owner

Parallelism: multiple tasks run at the same time, in parallel.
Middleware: software that enables one or more kinds of communication or connectivity between two or more applications in a distributed system. A middleware provides similar interfaces to all participants. It facilitates the interoperability of varying software framework applications. It provides a standard-based data exchange so two applications can connect without communicating directly. It converts network/process failures into programming-level exceptions. A middleware also provides uniform computational models. Examples of Middlewares: Java remote method invocation, CORBA with remote object invocation.
Key challenges in distributed systems:

1) Transparency: single view of the system, hide numerous details
2) Heterogeneity: there are lots of different networks, computers, operating systems, programming languages and developers
3) Failure Handling: detecting, masking (ignoring), tolerating, recovery, redundancy. A failure can be partial (some components fail while others continue to function) but also particularly difficult to handle. Failure handling consists of the set of actions taken by systems in order to circumvent failures.
4) Openness: extensibility, publication of interfaces. Distributed system cannot be accessed if they are closed. Openness determines if a system is extensible and re-implementable. It requires key interfaces to be published. The goal of openness is to allow components to be easily added or replaced. Open distributed systems must have a uniform communication

mechanism. They are constructed from heterogeneous hardware7software/vendors, but each component must conform to the published standard and be carefully tested and verified.

5) Scalability: controlling the cost of resources, controlling the performance, preventing the resources from running out, avoiding performance bottlenecks. Scalability is the ability of a system to continue working fluently when its size increases. Most systems suffer from performance loss when scaling. Problematic dimensions when scaling are the server performance and the network bandwidth; a solution might be to store data in a distributed manner to reduce overall data volume communicated.
6) Security: secrecy, authentication, authorization.

Advantages of distributed systems: cheap implementation and remote access.
Disadvantages of distributed systems: high maintenance, hard to integrate, dependency for most operations.
Transparency: a set of computers appears as a single computer to all applications and users. All details are hidden and dealt with transparently.
There are 8 different types of transparency:

1) Access transparency: enables local and remote resources to be accessed using identical operations
2) Location transparency: enables resources to be accessed without knowledge of their physical or network location
3) Concurrency transparency: enables several processes to operate concurrently using shared resources without interference between them
4) Replication transparency: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by the users or application programmers
5) Failure transparency: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components
6) Mobility transparency: allows the movement of resources and clients within a system without affecting the operation of users or programs
7) Performance transparency: allows the system to be reconfigured to improve performance as loads vary
8) Scaling transparency: allows the system and applications to expand in scale without change to the system structure or the application algorithms

System	*Description*	*Main Goal*
Distributed Operating System (DOS)	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
Network OS (NOS)	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

Distributed systems should be failure-transparent. A failure on components should not be fatal (or ideally, detectable) to the applications. Example: a failure in a bank server should not block you from withdrawing money.
To operate on an entity, access to it is essential. Each entity should have an access point, usually called address. Addresses of an entity change due to relocation (new IP address) or to reorganization (a Web server may be moved to a new host). An entity may have more than one access point (replication).
Naming provides an abstraction useful for location independence, relocation of entities, allowing a single reference to a set of alternative access points, offering human-friendly names. Names do not generally indicate where an object is located.

Three naming types:

1) Hierarchical naming: used by DNS, Unix File System
2) Flat Naming: used by distributed hash tables
3) Attribute-based naming

Advantages of **naming** in distributed systems:

1. With naming you are independent of the location
2. You have a single reference to the system
3. It's human friendly

Flat Naming:

- Common in centralized systems (with central device, e.g. star topology)
- Very complicated in decentralized systems (no central device, e.g. mesh topology).
- Used for addressing space in homogenous way (e.g. memory addressing)
- There is no explicit relationship between names in the namespace
- Approaches to locate mobile entities used in flat naming: Broadcasting, Multicasting and Forwarding Pointers

Attribute-based Naming:

- LDAP (Lightweight Directory Access Protocol) -> protocol for storing and accessing directory information in a distributed system. It allows resources to be identified and located by their attributes, such as their name or role.
- Each attribute called Relative Distinguished Name (RDN) -> A sequence of those names/attributes result in a globally unique name
- More easily searchable using queries than for example using the domain name system -> e.g. country = CH, City = ZH …

Hierarchical Naming:

- Used for a Domain Name System (DNS) server -> Translates domain names into IP addresses (used by servers)

Home-Based Approach: A host has a **home** which interacts with the clients who don't know that the host moved to a new location. Like this you don't have to inform everyone of your new location but only the people who actually want to talk to you

1. The client sends packet to the hosts previous home location
2. The home provides the client with the hosts new location so in the future the client can send the packets directly to the host's new location
3. The home tunnels the packet to the host's current location
4. From now on the client knows the host's new location and sends the packets there directly

Each cost has a fixed home location (e.g.. an IP address). When a host moves to a new network, it receives a new (temporary) IP address (mobile IP), the temporary IP address is stored at the home location.

Chord Distributed Hash Table -> used for routing

- Each node only connects to its direct neighbours
- Instead of key -> value there now is: name -> address
- Each node has an ID and each has a small routing table (finger table) which are pointers to other nodes
- To find a node holding the value for a key, find the largest node ID <= key in routing table or if none exist: values are stored in the node with the next higher ID in the routing table.
- A higher node stores all the keys below it upto the next lowest node
- The lookup is clockwise

In this example the address range is from 0 upto 2^8

To determine Node 28s finger table, one needs to apply this formula

i	Calculation (Position)	Closest node
0	28 + 2^0 = 29	33 (key 29 is stored in node 33)
1	28 + 2^1=30	33
2	28 + 2^2 = 32	33
3	28 + 2^3 = 36	89
4	28 + 2^4 = 44	89
5	28 + 2^5 = 60	89
6	28 + 2^6 = 92	110
7	28 + 2^7 = 156	157

In a finger table, the position (calculation) corresponds to $n + 2^i$, where $0<=i<=m-1$, and n is the value of the node for which we are building the finger table.
When starting from a node and wanting to go to the node which holds its key -> always go to the closest one

Example: key = 21 (is stored in Node 28)

Node 33 (starting node) -> Node 222 (closest to 21 in Node 33s finger table) -> Node 13 (it checks if it is keys 21 predecessor -> which it is) -> forwards it to the correct Node 28

7. Considering the circular Chord DHT node structure in 2, with 2^m possible integer valued addresses between $[0, 2^m)$, where $m = 8$.

   (a) In Figure 2 below, what would be Node 28's Finger Table?

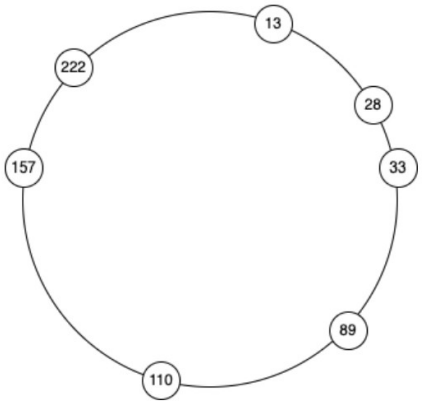


Figure 2: Chord Node Structure

E10 - Distributed File Systems

Examples of **file system modules**:

- Directory module -> translates file names to file IDs
- File Module -> translates file IDs to particular files
- Device module -> Disk Input/Output and buffering
- Access control module -> Checks permission for operating requests
- File access module -> Reads or writes file data or attributes

Distributed File System Requirements:

a) Transparency: 8 types
b) Concurrent file updates
c) File replication: process of generating copies of files
d) Hardware and software heterogeneity
e) Fault tolerance
f) Consistency: every node has the same view of data at a given time irrespective of which client has updated the data
g) Security
h) Efficiency

Unique file identifiers (UFID) are used to refer to file in all requests for service operations.

Network File System (NFS) concerns:

a) Auto-installer: mount on access of any non-mounted file.
b) Server caching
c) Client caching

d) Security: quite fragile, better with access control systems (Kerberos)

Valid operations of a **Network File System** (NFS):

- lookup(dirfh, name)
- mkdir(dirfh,name,attr)
- symlink(newdirfh,newname,string)
- Uses Automounter to improve performance and scalability->otherwise large number of unused entries in mount tables

Interplanetary File System (IPFS) characteristics:

- **Decentralized** vs centralized (https)
- **Content-based** vs location based (https)
- **Files** identified **by** a **hash** vs host identified by IP address (https)
- **Higher availability** because data decentralized across multiple peers vs data stored in a single host -> lower availability (https)
- Uses Distributed hash tables (DHT)

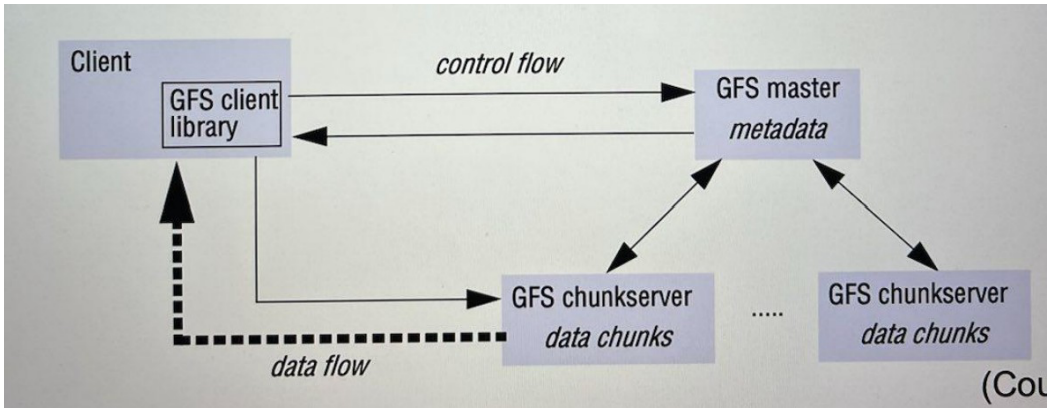
Distributed Garbage Collector (DGC) -> Makes sure that if a object is no longer referenced the object will be deleted

Unique File Identifiers (UFIDs) characteristics :

- Unique because you only want to access a unique file and not mutliples/the wrong one
- Uniqueness ensured through a 48 bit integer ID -> 32 bit IP address of host who creates it and a 16-bit integer of the creation date/time

Google File System uses the following **components** in its structure:

- Server chunks -> Data flows from server to client
- Master (for coordination) receives instructions from client and forwards the necessary action to the serverchunks
- Client -> Gives instructions to master and receives files from serverchunks



- **Transparency** requirements **essential** for **distributed Systems**:
  Replication transparency -> create multiple copies without affecting applications using the data, copies must have same contents, same operation must always yield the same result

- **Concurrency** transparency -> several processes operating on and using shared resources simultaneously without interfering between them
- **Migration** transparency -> Migrate resources/data objects/process without affecting operations of users/programs
- **Distribution** transparency -> Distributed systems look similar/identical to conventional programming -> when using it should feel like you are using just one single computer

Communication mechanisms which can be used for **remote file systems**:

- LAN
- WAN
- Peer-to-peer link

E11 – Synchronization in Distributed Systems

Computer Network vs Distributed System: In a distributed system, the computers and devices are typically connected by a computer network, but the key difference is that a distributed system is designed to coordinate and communicate in order to perform a specific task or service, whereas a computer network is simply a means of connecting devices and enabling communication between them.
Algorithms that can be used to coordinate multiple tasks: Ricart-Agrawala algorithm, Bully algorithm.
How to ensure synchronization?

a) Agreement on total ordering of events

What is the problem if all devices share the same clock? Synchronization issues
Could we have an absolute time synchronization? Does not avoid synchronization problem, it provides inaccuracy and clock drifts (difference between actual time and clock time).
Two drawbacks of causal communications when they are includes in the middleware layer:

1) Some causality might not be captured in the middleware. External communication can mess up the causality of the middleware (e.g., Alice informs bob about a message and bob takes another action that depends on the information got from Alice, this causality is not captured by the middleware).
2) Potential causality is captured: non-related messages that happen in an order are considered dependent: this makes the system more complex than necessary.

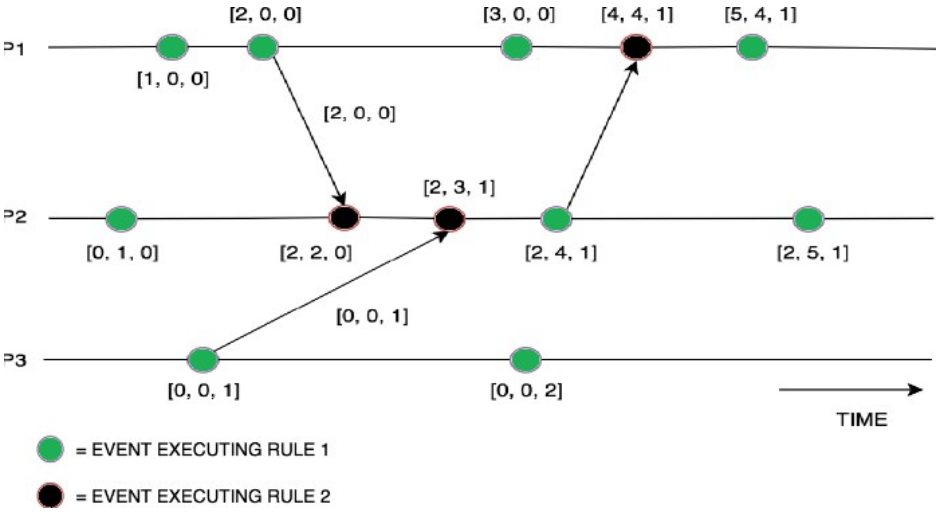
Causality: principle that an event or action will produce a specific effect.
Mutual exclusion property of synchronization: states that there can only ever be one process in the critical section at any given point of time. This term has been coined by Dijkstra.
To guarantee mutual exclusion, these properties must be satisfied:

a) Safety: only one process can be executed in the critical section
b) Liveness: no deadlocks or livelocks should occur
c) Ordering: requests are handled in order of appearance

Logical clocks: absolute time synchronization is not always needed; we need to ensure that the order in which events happen is preserved across all computers. All computers should agree on a total ordering of events. In a system using logical clocks, each node or process maintains its own logical clock, which is incremented each time an event occurs at that node or process. When a node or process sends a message to another node or process, it includes the current value of its logical clock as part of the message. The receiving node or process updates its own logical clock based on the value of the sender's logical clock and the current value of its own clock. Lamport's timestamps give a total ordering of events but:

- Notion of causality (dependencies between events) is lost
- Total ordering is often too strict
- Adjustment of clocks is needed not retain the total order of events

In a system using vector clocks, each node or process maintains a vector of clock values, one for each node or process in the system. When an event occurs at a node or process, the value of the clock for that node or process is incremented, and the vector is updated to reflect the new clock values. When a node or process sends a message to another node or process, it includes its current vector of clock values as part of the message. The receiving node or process updates its own vector of clock values based on the values in the sender's vector and its own current vector.
Vector clocks are generally more accurate and reliable than logical clocks, as they take into account the interactions between multiple nodes or processes in the system.



The centralized approach is an alternative to guarantee mutual exclusion:

- Advantages: simple implementation, no starvation (no process has to wait forever), guarantees mutual exclusion, fair (order maintained with queue), only takes 3 messages per use of resource (request, grant, release).
- Disadvantages: single point of failure (coordinator), processes cannot distinguish between busy resource or dead coordinator, in large systems a single coordinator may become a performance bottleneck.

In the centralized approach one process is elected as the coordinator. Process 1 asks coordinator for permission to access, permission is granted. Process 2 then asks permission to access same

resource, coordinator does not reply. Process 1 is then released and sends a message to the coordinator, which then replies to process 2.

Algorithms for coordinator election:

- Bully algorithm, Ring election algorithm

Election Algorithms: algorithms that choose a process from a group of processors to act as a coordinator. If the coordinator process crashes due to some reasons, then a new coordinator is elected on other processor. Election algorithm assumes that every active process in the system has a unique priority number. The process with highest priority will be chosen as a new coordinator. Hence, when a coordinator fails, this algorithm elects that active process which has highest priority number. Then this number is sent to every active process in the distributed system.

There are two election algorithms for two different configurations of a distributed system:

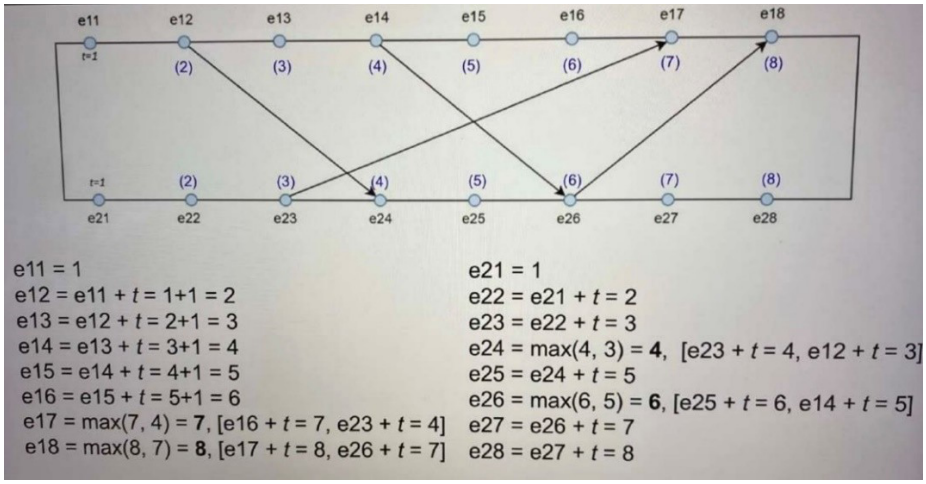
1) The Bully Algorithm – This algorithm applies to system where every process can send a message to every other process in the system.
   Algorithm: Suppose process P sends a message to the coordinator.
   a) If the coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.
   b) Now process P sends an election message to every process with higher priority number than his.
   c) It waits for OK responses, if no one responds for time interval T then process P elects itself as a coordinator.
   d) Then it sends a message to all lower priority number processes that it is elected as their new coordinator.
   e) However, if an answer is received within time T from any other process Q,
      (I) Process P again waits for time interval T' to receive another message from Q that it has been elected as coordinator.
      (II) If Q doesn't respond within time interval T' then it is assumed to have failed and algorithm is restarted.
2) The Token Ring Algorithm – This algorithm applies to systems organized as a ring. In this algorithm we assume that the link between the process are unidirectional and every process can message to the process on its right only.

Algorithm :

1. If process P1 detects a coordinator failure, it creates new active list which is empty initially. It sends election message to its neighbour on right and adds number 1 to its active list.
2. If process P2 receives message elect from processes on left, it responds in 3 ways:
   - (I) If message received does not contain 1 in active list, then P1 adds 2 to its active list and forwards the message.
   - (II) If this is the first election message it has received or sent, P1 creates new active list with numbers 1 and 2. It then sends election message 1 followed by 2.
   - (III) If Process P1 receives its own election message 1 then active list for P1 now contains numbers of all the active processes in the system. Now Process P1 detects highest priority number from list and elects it as the new coordinator.

Distributed approaches:

1) Lamport timestamps: can be used to capture causality in a distributed system, but they are not a complete solution for determining causality. This is because Lamport timestamps only provide a partial ordering of events, which means that they do not necessarily capture all of the relationships between events.



2) Ricart-Agrawala algorithm: it is not a leader election algorithm but instead guarantees the mutual exclusion property and works by having each node or process in the system request permission to access a shared resource, and then granting permission to only one node or process at a time. When a node or process wants to access the shared resource, it sends a request message to all of the other nodes or processes in the system. If the requesting node or process has the highest priority (lowest timestamp) among all the nodes or processes that have requested access to the resource, it is granted permission to access the resource. If two or more nodes or processes have the same highest priority, they must wait until all the other nodes or processes have either granted or denied permission before they can be granted permission to access the resource. Nodes in the Ricart and Agrawala algorithm use logical clocks: all events are in total order.

Algorithm	Messages per entry	Waiting time to enter critical state	Problems
Centralized	3	2	Crash of coordinator
Distributed	2*(n-1)	2*(n-1)	Crash of any node
Token ring	1 to infinite	0 to n-1	Lost token, crash of any node in ring

Multicasts messages are useful in distributed systems, because multiple receivers can receive the same message. Characteristics of multicast messages:

a) Fault tolerance based on replicated servers: even when some of the members fail, clients can still be served.
b) Discovering services in spontaneous working
c) Propagation of event notification. Everybody is informed and therefore information is not lost.

Problems of Basic Multicast algorithm: implosion of acknowledgments, unreliable.

For a reliable multicast algorithm, the following properties must hold:

a) Integrity: a working process p in group g delivers a message m at most once, and m is multicast by some working process.
b) Agreement: if a working process delivers m then all the other working processes in group g will deliver m too.

E12 – Coordination

Atomicity means **all commit or all abort** -> guarantees that operation is completed for all participants or for none -> state is always consistent

In **Distributed Systems** major aspects of **atomicity** are:

- **Serializability** -> order of executions doesn't matter -> result is always the same
- **Recoverability** -> Each operation executes completely or not at all

Difficulties:

- Communication System Errors
- Communication Protocols unreliabilities
- Distributed System Hardware failures
- Application misbehaviours
- →all of them lead to problems that the state among all systems have to be consisten

Desirable Properties of atomic commit:

- If one aborts no one commits
- If there is a failure -> A and B cannot commit -> abort everything

A **Timeout** can occur in the Two-Phase-Commit Protocol (2PC) in the following cases:

- The Transaction Coordinator (TC) is running but doesn't receive the message it expects
- A participant crashed
- The network is down/The network dropped messages

A **Reboot** can occur in the Two-Phase-Commit Protocol (2PC) in the following cases:

- The TC crashes -> it needs to reboot
- Participants crash -> they need to reboot

To **prevent crash** and **reboot** problems:

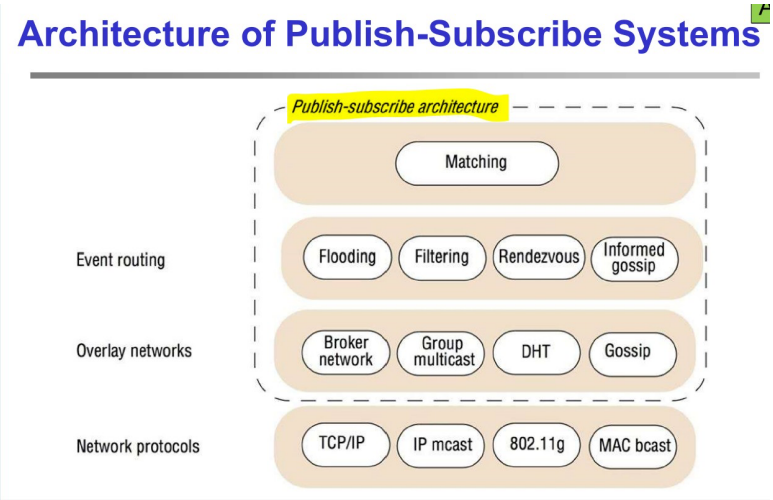
- Store the current state in non-volatile memory -> memory which isn't erased when there is no power -> floppy disk
- The order of store and send needs to be saved correctly
- Messages need to be written on disk before sending them -> everyone needs to do this including TC

Different Views of a Distributed System:

- **Synchronous** (time-couple) and **Named** -> communications are directed at defined receiver which must be running at the same time -> e.g. Remote Procedure Call (RPC), Remote Method Invocation (RMI)
- **Synchronous** (time-coupled) and **Unnamed** -> Unknown name of receiver which must exist at the same time -> e.g. Multicast
- **Asynchronous** (time-uncoupled) and **Named** -> Communications directed at defined receiver which exists at some time -> e.g. E-Mail
- **Asynchronous** (time-uncoupled) and **Unnamed** -> Unknown name of receiver who should exist at some time -> e.g. Tuple Spaces (distributed computing), Zookeeper (service for maintaining config information, naming,…)

Examples of **Publish-Subscriber** (asynchronous/named) paradigms:

- (Event) **Type-Based** subscription -> organize events in hierarchy and only get the ones of a particular type
- **Content Based** subscription -> filter events based on content
- **Channel-based** subscription -> get all events of the subscribed channel



The **Information Bus (TIB)/Rendezvous** -> **Messaging middleware** between multiple services ->provides **high-speed data distribution**, no single point of failure -> fully distributed (peer-to-peer architecture)

General Multicast Approach used for reliable communications -> **Receivers send NACK** (Not acknowledged) -> **retransmission for complaining receivers** -> more reliable this way -> everyone who needs it gets message

Jini System -> type of distributed system -> turn network into flexible, easily administered tool with which services can be added/removed dynamically to build such a system you need:

- Components that provide infrastructure to build such a system
- Programming model/framework
- Services that can be integrated into the Jini system which offer functionality to other members of the network

E13 – Distributed Systems in Use

Crawler: type of software program that has the task to locate and retrieve the contents of the Web and pass the contents onto the indexing subsystem. It downloads a page, checks the page for links and then downloads the other pages recursively.

URL resolver: piece of software that takes an URL as input and converts it into the IP address that corresponds to that URL. This is necessary because while humans can easily remember and understand URL, computers deal exclusively with IP addresses. It converts relative URLs into absolute URLS and into docIDs.

Indexer: software component that produces an index for the contents of the Web. It uncompresses, parses, gets and stores words with docID, gets links and stores them.

Doc Index: store information about docs, like ID, URL, status.

PageRank: the more pages link to a page the more important that page is. The more often a word occurs the more relevant it is.

After a failure occurs during the execution of a MapReduce, the master assigns the tasks and monitors the progress, it reschedules the failing tasks.

Scenarios suitable for MapReduce:

a) a cluster with willingness to use cloud services
b) Working with large datasets (big data scenario)

Multimedia system categories: web-based multimedia, Video-on-Demand (VoD), Video-on-reservation (VoR)

Quality of Service (QoS) management for multimedia systems enables multitasking in an operating system, multiplexing of data streams on one network access, using concurrent physical resources, and resource allocation to meet all data stream requirements on a certain machine.

Traffic shaping is used to optimize or guarantee performance, improve latency, or increase usable bandwidth for some kind of packets by delaying other kinds.

Algorithms that can be used for traffic shaping: leaky bucket, token bucket.

RFC 1363 standardized the flow specification as a data structure used by hosts to request special services on the intranet. For compatibility reasons, specifications of flows are standardized in Eleven 16 bit name-value pairs. The protocol version consists of Bandwidth, Delay and Loss.

	Protocol version
Bandwidth:	Maximum transmission unit
	Token bucket rate
	Token bucket size
	Maximum transmission rate
Delay:	Minimum delay noticed
	Maximum delay variation
Loss:	Loss sensitivity
	Burst loss sensitivity
	Loss interval
	Quality of guarantee

Scaling possibilities that can be made to improve the data stream if QoS cannot be met (adaptations on scaling can help):

- Temporal scaling: reduction of amount of audio/video frames delivered at an interval of time to lower the resolution of the video stream -> reduction of audio/video resolution.
- Spatial scaling: reduction of pixels per image.
- Frequency scaling: modification of compression per image.
- Amplitude scaling: reduction of color depth per image.

Data parallelism refers to a technique in which multiple copies of the same operation/task are performed simultaneously on different data. This is typically achieved by dividing the data into smaller chunks and distributing the chunks among the available processors or computers. Each processor or computer then performs the operation on its own chunk of data, and the results are combined at the end.
Task Parallelism: Each oval is a server (e.g. url server, crawler, indexer).
Task parallelism, on the other hand, refers to a technique in which different operations are performed concurrently on different data items. This is typically achieved by dividing the task into smaller sub-tasks and assigning each sub-task to a separate processor or computer.

Data Parallelisms	Task Parallelisms
1. Same task are performed on different subsets of same data.	1. Different task are performed on the same or different data.
2. Synchronous computation is performed.	2. Asynchronous computation is performed.
3. As there is only one execution thread operating on all sets of data, so the speedup is more.	3. As each processor will execute a different thread or process on the same or different set of data, so speedup is less.
4. Amount of parallelization is proportional to the input size.	4. Amount of parallelization is proportional to the number of independent tasks is performed.
5. It is designed for optimum load balance on multiprocessor system.	5. Here, load balancing depends upon on the e availability of the hardware and scheduling algorithms like static and dynamic scheduling.

MapReduce: The MapReduce model is a programming model for efficient distributed computing and it consists of two main functions: "map" and "reduce". The map function takes a set of key-value pairs as input and produces a set of intermediate key-value pairs as output. The reduce function takes the intermediate key-value pairs produced by the map function as input and produces a set of output key-value pairs. In a MapReduce program, the input data is divided into chunks, and the map function is applied to each chunk in parallel on different machines. The intermediate key-value pairs produced by the map function are then sorted by key and passed to the reduce function, which is also applied in parallel on different machines. The output of the reduce function is the final result of the MapReduce program.

Partitioning function: function that determines which intermediate key-value pairs produced by the map function will be sent to which reduce function for further processing.

The partitioning function is applied to each intermediate key-value pair after the map function has been applied and before the reduce function is applied. Its purpose is to divide the intermediate key-value pairs into a set of partitions, with each partition containing a subset of the key-value pairs.

The partitioning function is typically implemented as a hash function that takes the key as input and maps it to a specific partition based on the value of the hash. The number of partitions is typically equal to the number of reduce functions, so that each reduce function receives a distinct set of intermediate key-value pairs.
Namespace: set of signs (names) that are used to identify and refer to objects of various kinds.

E14 – Security

Security Classifications

Vulnerability

A quality, property or characteristics of a system that provides an opportunity for misuse

Threat

Any potentially malicious occurrence

Has an undesirable effect on the assets and resources of an IT system

Risk

Normal risk management: risk = likelihood x impact

Applied to IT security: risk = threat x vulnerability

Security Areas

Organizational Security

Access rights, key management

Technical Security

Security mechanisms, algorithms

User Behaviour

Passwords, external and internal attacks

à all these aspects lead to Information system security

Authentication Mechanisms

Ownership

Smart card, physical device etc.

Knowledge

Password, account etc.

Biometrics

Finger print, iris scan, face scan etc.

Location or context

Proficiency

Signing etc.

Cyberattacks

Phishing

(distributed) Denial of Service

Generate as much possible work for an infrastructure to prevent the processing of normal tasks. Distributed if the sources of attack come from a big number of different nodes.

Easvesdropping

Someone who is not part of the communication is listening to a communication channel

à threat for confidentiality

Use encryption to prevent eavesdropping

Security Pillars

Authentication

Ensure that partners involved in communications really are the people they claim to be

Authorization

Access rights

Integrity

Protection against the modification of a message along a transmission path

Privacy

Message is not read by somebody other than intended

Confidentiality

Only authorized receiver can interpret the message

Non-repudiation

Neither the sender nor the receiver can deny that the communication has taken place

Two Factor Authentication

To get authenticated, one needs to pass two different schemes in combination

Example: Login at an insurance requires Password and SMS Pin sent to your mobile phone

à defence in depth

Hashing Functions

One-way functions working with modulo operations

the idea is that, given an input the output can be computed very easily, but the other way around its almost impossible

The cryptographic strength of a hash function is depending on the size of the hash output, size and quality of the key.

Rainbow tables can be used to find the plaintext of passwords even if they are hashed.

They are really big dictionaries with frequently used passwords linked with their corresponding hash value. One can search for a given hash and the rainbow table outputs the password in plaintext if the entry exists, therefore was precalculated.

A defence strategy is the process of salting.

Encryption

From plain text to cipher text (=encrypted text)

Basic principle: Message P

Cryptofunction K(x)

Ciphertext C

à C = K(P) for encryption

à P = K© for decryption

Symmetric cryptography

Entities share a private key, the encryption and decryption is done with this single shared key

+ short key, only few calculations

- key exchange is a major vulnerability

Asymmetric cryptography

Every entity owns a pair of public and private keys

The public key is shared and accessible by anyone, while the private key is known by the owner entity only.

If B wants to send a Message to A, the public key of a is looked up.

B uses this public key to encrypt its message and sends it to A.

Now A can use the private key to decrypt the message.

+ no sharing needed

- longer keys, more calculations

In practice the symmetric encryption alone is not secure enough, either asymmetric encryption or a combination of the two are used.

Digital Signatures

The idea is to verify that a certain message is really coming from the person that is stated in this message and that the content did not change.

Every entity has a pair of public and private keys.

A wants to send a message to B.

The signature is created by giving the whole message as input to the private key and added to the message. The message is sent to B.

B uses the public key of A and fills in the signature, as result the message should be the output. If this is the case, the verification was successful, and B knows that the message was not changed during transmission and that A is the real sender.

Other concepts

Tunneling

Embedding an IP packet into a second IP packet