Freie Universitat Berlin
Erasmus Program

10 ECTS

Telematics

Professor:
Prof. Dr. Ing. Jochen Schiller

Autor:
Filippo Ghirardini

Winter Semester 2024-2025

Contents

Telematics

Autor: Ghirardini Filippo

Winter Semester 2024-2025

1 Basics

1.1 Network composition

A network consists of three elements:

- **End systems**: can vary in size and usage
- **Intermediate systems**: these (e.g. routers) are the components that allow the network to work.
- **Links**: they connect the end systems and can be *optical*, *copper* or *wireless*. Even if wireless is becoming more and more important, cables are still fundamental (undersea cables, underground cables).

Question 1.1.1 (Why fiber optic?)**.** Because this medium has not reached it's maximum and still has a huge potential of **bandwidth**. Also, while copper cable start acting as an **antenna** (and a receiver), disturbing near copper cable, fiber optic doesn't have this problem. Furthermore, copper cables need amplifiers which increase **latency**.

Question 1.1.2 (Why copper?)**.** It's **cheaper** and **easier** to handle.

Question 1.1.3 (Why cables over wireless?)**.** Because of **stability** and **latency**. Usually the problem is tampered by buffers but obviously it doesn't work with interactive applications.

Question 1.1.4 (What are threats to cable?)**.**

1.2 Communication principles

There are two basics principles:

- **Synchronous**: joint action of sender and receiver. Requires **waiting** until all parties are ready (e.g. phone calls)
- **Asynchronous**: sender and receiver operate decoupled (e.g. SMS, email). Requires **buffering**.

Note 1.2.0.1. There is also **isochronous**, which means the messages are sent every predetermined amount of time.

1.2.1 Direction

Communication channels may allow traffic flow in different directions:

- **Simple duplex**: one direction
- **Half-duplex**: both directions in different moments
- **Full-duplex**: both directions at the same time

1.2.2 Distribution

The communication distribution can happen in different ways:

- **Unicast**: one to one
- **Broadcast**: one to all
- **Multicast**: one to a subset
- **Anycast**: one to the nearest, e.g. when requesting to a redundant database you don't care which one responds
- **Concast**: many to one, e.g. we collect sensor data and send it to one
- **Geocast**: one to a certain region

Note 1.2.2.1. Even if multicast would be easier and cheaper, companies usually go for unicast because they want to know who the clients are.

Note 1.2.2.2. Broadcast guarantees anonymity while multicast does not.

1.2.3 Topologies

The main topologies are:

- **Full mesh**: too expensive
- **Chain**: in cars and trains
- **Star**: ideal for switches
- **Partial mesh**: the best compromise
- **Tree**: not ideal for big networks since if you cut a side, you lose contact

1.3 Sharing

1.3.1 Cons

Sharing may create a lot of problems, like **bottlenecks**: links and intermediate nodes are shared between end systems. One solution may be to *reroute* or to start *dropping packets* (e.g. when streaming the resolution lowers down).

1.3.2 Pros

At the same time, sharing means more efficient (less expensive) mechanism to **exchange data** between different components of distributed systems and **minimize blocking** due to multiplexing.

1.3.3 How?

There are two possible ways of sharing:

- **Reservation**: you reserve in advance the resource so that it is guaranteed, e.g. remote surgery. When the peak demand and the flow duration varies, there are two options:
  1. *First Come First Served*
  2. Everyone gets $10Mbps$

  It is implemented with **circuit-switching**: establishes dynamically a dedicated communication channel. It has predictable performance and a simple and fast switching but it's inefficient for bursty traffic, complex to setup and not easily adaptive to failures.
- **On-demand**: when there is a resource available you take it (variable *delay*, **jitter**), e.g. email. It is implemented with **packet-switching**: splitting the resource in packets and multiplex them. Much more flexible but requires buffers, packets overhead and has unpredictable performances.

Observation 1.3.1. It all depends on the application. Each flow has a **peak rate** and an **average rate**.To decide if *reservation* works well for a specific case, we must look at the ratio $\frac{P}{A}$. If it's small then it works well, otherwise it's wasting resources.

1.4 Internet

Internet is a network of networks. It enables processes on different hosts to exchange data: it's a bit delivery system.
ISPs enable you to access and use Internet services: well defined and commonly required functions. There are two roles: **client** and **server** that can be on different machine (or not, like with P2P).

Definition 1.4.1 (Internet)**.** *The set of all reachable parties (IP addresses).*

1.5 Protocols, layer and standards

Definition 1.5.1 (Digital Data Communication)**.** *Processing and transport of digital data between interconnected computers.*

Definition 1.5.2 (Data)**.** ***Representation*** *of facts, concepts and statement in a formal way which is suitable for communication, interpretation and processing by human beings or technical means.*

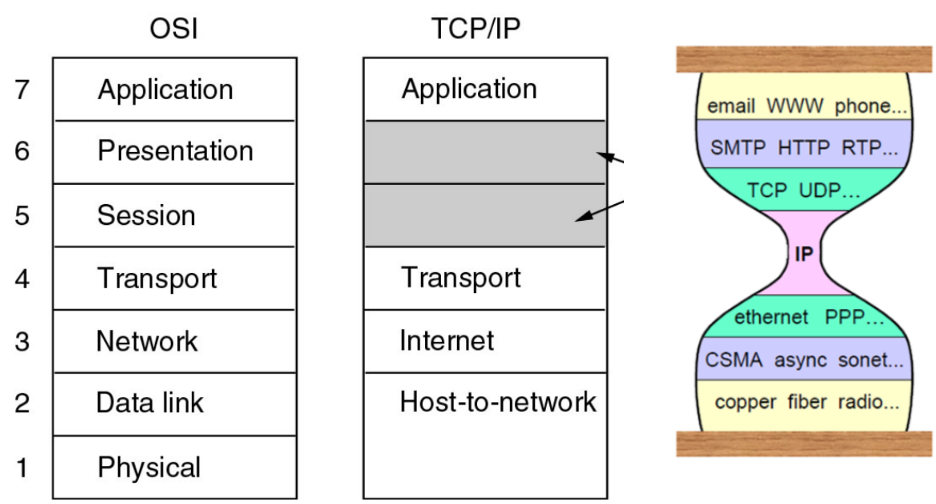
Note 1.5.0.1. **Information** is different from the data.

Definition 1.5.3 (Signal)**.** *A signal is the physical representation of data by spatial or timely variation of physical characteristics.*

In this context we need rules of communication for the different devices to communicate: we have heterogeneous computer architectures, network infrastructure and distributed application. A **protocol** is a conversational convention, consisting of syntax and semantic.

Definition 1.5.4 (Protocol)**.** *Protocols define format, order of messages sent and received among network entities, and actions taken on message transmission and receipt.*
A protocol is a set of ***unambiguous*** *specifications defining how processes communicate with one another through a connection (wire, radio etc.).*

To provide structure to the design of networks protocols, network designers organize protocols in **layers**. We use two models, either the ISO/OSI or the TCP/IP.



All the different layers need additional information, which is added via **headers** to the data payload via **encapsulation**. This could cause a lot of **overhead**.
All the protocols rely on the Internet Protocol. This maximizes **interoperability** and does not ensure anything, therefore no one has expectations.

1.6 Quality of service

To define the quality of communication we check:

- **Technical performance**: delay, jitter, throughput, data rate, etc.
- **Costs**
- **Reliability**: fault tolerance, system stability, immunity, availability
- **Security and Protection**: eavesdropping, authentication, denial of service, etc.

1.6.1 Latency

The main parameters we check are:

- **One-way delay**: measured in seconds

$$d_1 = t'_1 - t_1 \tag{1}$$

- **Round-trip-time**: measured in seconds

$$r_1 = t_2 - t_1 \tag{2}$$

  It should also integrate the processing time of the other device.

1.6.2 Stability

The main parameter that measures stability is the **Jitter**. It's measured in seconds and calculated using the delay:

$$d_i = t'_i - t_i \qquad j_i = d_{i+1} - d_i \tag{3}$$

1.6.3 Capacity

From a capacity perspective, we measure the **throughput** in $\frac{bit}{s}$ as follows:

$$T = \frac{\sum \mathrm{data}_i}{\Delta t} \tag{4}$$

The **goodput** instead, is the amount of **useful** throughput from a user perspective.

Note 1.6.3.1. **Bandwidth** is used for the description of the channel characteristics.

There is also the **Delay-Throughput-Produc** which measures how much data can be on the medium itself while traveling. E.g. with a connection of $1Mbps$ that has $200ms$ of delay we have

$$1Mbps \times 0.2s = 200kbit$$

2 DNS

2.1 Introduction

Names provide a level of abstraction from the IP address: for humans it's easier to remember. It also provides **load balancing** and easy **aliasing**.
The decision for DNS adding is handled by two organizations:

- **IETF**: how entries are entered and read from the phone book
- **ICANN**: how to decide *what* names should be entered in the phone book

To use naming you need two things:

- **Unique** names
- **Resolution** of names to locator (IP address) or other services

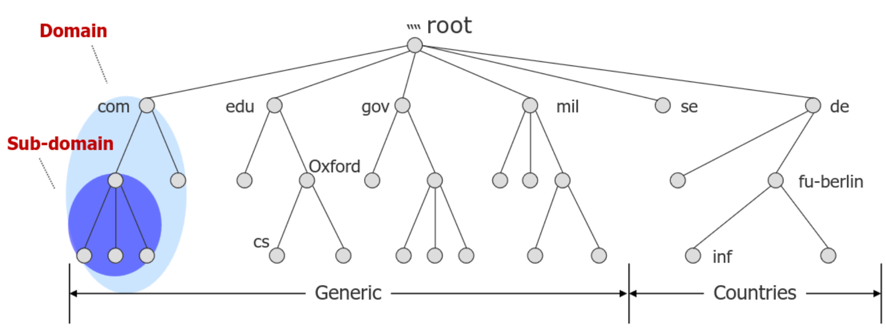
2.1.1 Scaling

To allow scaling, DNS uses **delegation** and **caching**. In particular for delegation, DNS adopts three intertwined **hierarchies**:

- **Name space**: hierarchy of names
- **Management**: hierarchy of authorities over names. Who owns which name part.
- **Infrastructure**: hierarchy of DNS server. Where is the mapping stored.

2.2 Namespace

DNS namespace is implemented as a tree structure: each node has a **label** which identifies it relatively to its parent node. Each node is **root** of a sub-tree (if it's not a leaf). In particular direct children of the root are called **Top Level Domains**. Each subtree represents a **domain** and each domain can be divided in **sub-domains**.



There is a limited number of TLDs: originally it was 7 plus one for each country. Now there are many more, even in non Latin alphabets.

2.2.1 Leafs

The name of a domain consists of a sequence of labels beginning with the root of the domain and going up to the root of the whole tree. Each label is separated by ".".
In the leaf nodes the IP addresses are associated with the names.
Furthermore, there could be **Domain Name Aliases**: pointers of one domain to another (Canonical Domain Name).

2.2.2 DNS Database

There are a few rules for the database:

- The **depth** of the tree is limited to 127
- Each label can have up to 63 characters
- The whole domain name has a maximum of 255 characters (even if the average is 10)
- A label of length 0 is reserved for the root

The full address to a host is the **Fully Qualified Domain Name**, which includes the leaf, each node and the root. The **Relative Domain Name** instead, is an incomplete domain name.

2.3 Management

The management of domain names also follows a hierarchy structure: **ICANN** manages the root domain and delegates someone (**DENIC** for Germany) to handle the *de* domain. They then delegate FUB to handle the *fu* domain. And so on.
This solution ensures that the names are unique.

2.4 Name servers and zones

2.4.1 Domains

Domains are administrative concepts managed by single organizations. The name of the domain corresponds to the name of the root node. They can delegate the responsibility for subdomains to other organizations but maintains the pointer to them to be able to forward requests.

2.4.2 Name servers and zones

On the other hand, name servers and zones are technical concepts. The name server is a process that maintains a database for a domain space. The part of the name space known to the server is called a **zone** and it's stored in a *zone file*. The name server may have multiple zones and has authority over them.

Primary Master It's a name server that must exist. Reads the data from a local file and has a database describing subdomains and computer in a selected zone.

Secondary Master It's optional and is a replication of the master for reliability reasons. It receives the data from another server which is authoritative.

2.5 Resolution

There are two types of Name Resolution:

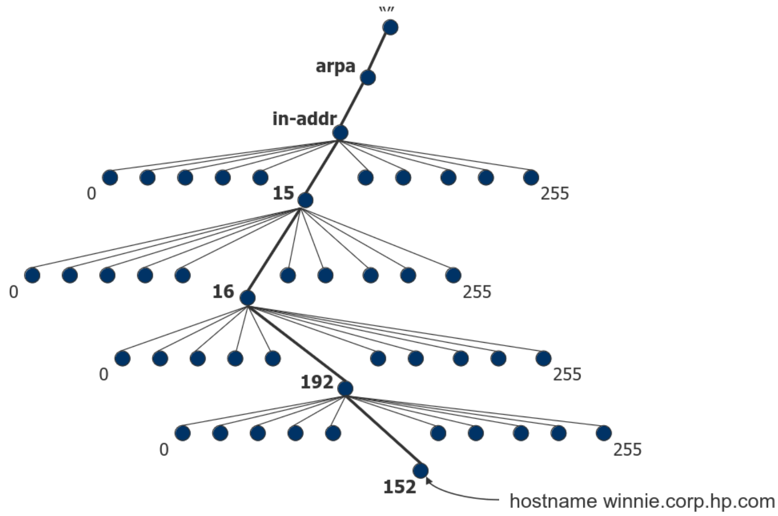
- **Recursive**: the name server replies either with the answer or with an error and it's responsible to contact the other nodes
- **Iterative**: a name server replies with the address of another one, it's the host duty to contact additional name servers for the answer

Question 2.5.1 (Why do root servers not support recursive solution?)**.** Using the recursive option implies that every intermediate needs to wait for all the others, depleting its resources.

Question 2.5.2 (How does this all contribute to scalability?)**.** We do not have **strong consistency** and

2.5.1 Reverse lookup

While mapping a name to a *global* IP address is simple, doing the other way round it's really difficult because we need to do a complete search of the tree.
Because of this, there is a special area in the database called **in-addr.arpa** that contains 256 sub domains, each one having 256 and so on.



Note 2.5.1.1. This is useful against **spoofing**: as an example if you get an email and you want to check if the sender is who claims to be, you can do a reverse lookup on the IP of the email server.

Question 2.5.3 (Why does reverse lookup not always work?)**.** Because the entries are not always present in the database.

2.6 Database entries

A **resource record** is the entry in the database to get the address or other information of a name. It's composed of a tuple:

```
(name, TTL, class, type, value)
```

TTL It's the Time To Live: after a certain amounts of seconds the record will be deleted from the cache and updated. With a shorter TTL you have a very updated cache while longer TTL means outdated caches but less requests for the server.

Type Indicates the type of data to be returned. **A** is the actual IPv4 address corresponding to the name (**AAAA** for IPv6).

Class Nowadays it's only **IN** but there were in the past other options for different networks with independent DNS zones.

Observation 2.6.1 (Load balancing)**.** DNS is very useful for load balancing: depending on the region when you ask for a DNS entry the answer will be the closest one. It can also be used for **evil purposes** (censorship, marketing).

2.6.1 Name Server

For each name server of a zone a **Name Server** record is created in the cache. E.g. when you want to visit *arnold.movie.edu* you may have in cache a NS entry for *movie.edu*.

2.6.2 CNAME

A **CN** record is an optional entry in the database that illustrates aliases on its canonical names.

2.6.3 Pointer

The **PTR** record provides information for the mapping of an address to names. If you do not have any entry for an IP address you then have to do a reverse lookup. Addresses should refer only to one name.

2.6.4 Mail Exchanger

The **MX** record serves for the controlling of the email routing. It specifies an email server responsible for a domain name with the option to indicate a preference if multiple servers are present (the smallest value is preferred).

2.7 Protocol

The resolver software triggers the resolution process and tries first for the cache. Then it sends the request to the local DNS server which is either static (resolv.conf) or dynamic (DHCP).
The protocol consists of a single packet used for inquiries and responses:

32 bits

Identification	Flags
Number of Questions	Number of Answers RR
Number of Authority RR	Number of Additional RR
Questions (variable number of RR)	
Answers (variable number of RR)	
Authority (variable number of RR)	
Additional information (variable number of RR)	

Identification 16*bits* for the mapping of an inquiry to a response.

Flags 16*bits* of various flags that indicate if the packet is a request or a response, if it's **authoritative** or not, if it's *iterative* or *recursive*.

Numbers These fields indicate the contained number of inquiries responses data records.

Questions Contains the names to be resolved.

Answers Resource records to the previous inquiry.

Authority Contains the ID of the passed responsible NS.

Additional information If the name searched is only an alias, the belonging resource record for the correct name is placed here.

The packet is sent through UDP on port 53 and the **reliability** is only implemented via repeating the requests. Also it is not protected.

2.8 Scalability

The scalability is achieved mainly with local caching of recent results. The cache can be in the network and also in the local client.
One of the main problem is how long should you keep the entries? You need to achieve both **consistency** and not doing too many requests. You also need to detect and flush the **stale entries**. You have to avoid **cache poisoning**: when a malicious person changes the value in the cache to redirect you to an evil software.

2.9 Extension

2.9.1 Dynamic DNS

The problem comes up when, as an example, you restart the router and your public IP address is changed (or maybe the ISP changes it every 24 hours). The DDNS allows you to tell the changed IP address.

2.9.2 Characters

The original DNS supports only ASCII, so there is an extension for UTF characters.

2.9.3 DNSSEC

The **security** is important because DNS is the most crucial indirection to access the data. Controlling DNS response implies controlling the discovery of the communication endpoints. It may be use in an evil way for political and economical reasons.

3 Email

3.1 Introduction

Email is an example of an application that works on the different layers. It's **asynchronous**, **decentralized** (to improve *scaling*), **client-server** and based on simple ASCII text.

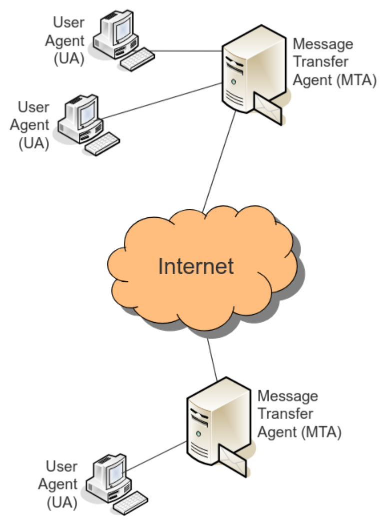
3.1.1 Motivation

Email was the first *killer application*. It started in the 1980's with a simple terminal interface, evolving in the 1990's with the *web-mail* and then *mobile email*. Today social network are trying to swallow the email concept.

3.2 Architecture

There are two actors involved:

- **User Agent**, with email clients. Runs on the computer of the user and is intermittently on. E.g. Thunderbird or Outlook
- **Message Transfer Agent**: the email servers. They run on a remote machine and stores and forwards on behalf of the User Agents. It's always on



3.3 Message

Message are viewed as having an **envelope**, the fundamental part for the delivery, and a **content**. The latter contains a **header** with a certain coding and a **body** consisting of simple characters.

3.3.1 Envelope

The envelope is created by the **MTA** or the **MSA** and includes all the information for transporting the message. Some information are redundant with the header (like the sending and receiving address) but there are some differences, like when you send a Blind Carbon Copy message.

Note 3.3.1.1. You can't know if the sender is correct. This can be used for evil purposes. The only way to avoid that is by encrypting or signing the email.

3.3.2 Content

Header It contains characters with the following syntax:

```
<key>:<value>
```

Body It's the content of the email. It's separated from the header by a blank line.

Since originally the content could only contain 7*bit* ASCII, **Multipurpose Internet Mail Extensions** was invented to extend the classical format. It adds additional headers, content types and sub-types:

- **MIME-version**
- **content-description**: string that describes the content of the message
- **content-id**: identifier for the content
- **content-transfer-encoding**: selected coding for the content (BASE64, ASCII)
- **content-type**: specifies the type of the body in the format *type/subtype*, e.g. text, image, audio, video, etc..

3.4 Protocols

3.4.1 SMTP

Simple Mail Transport Protocol delivers the mail to the final inbox. It can't ensure that the message arrives to the final user because it expects the receiver to be always online.
It uses **reliable data transfer** based on TCP on port 25 and it's *best effort*. It provides **little security**: no encryption, optional authentication on port 587 to reach MSA but nothing between MTAs.
The protocol follows these steps:

1. **Write** an email, the client formats it and sends it to it's own mail server
2. The mail server sets up a connection with the receiver's server and **sends** a copy of the email
3. The **receiver**'s server creates the header of the email and places the message in the inbox

Graylisting A first attempt to block spam. If a combination of IP address of the sender, their email and the receiver's one is seen for the first time, the message is discarded and an error is returned. From the second time on, the message goes through. This is based on the idea that scammers won't send the email twice.

3.4.2 POP3

This protocol pulls emails from the server over a connection on port 110. It's text based and allows basic functionalities such as *logging*, *copying locally* and *deleting* from the server.
It works in two phases:

1. **Authorization** phase: *user*name and *pass*word for authentication, either successful or not
2. **Transaction** phase: a *list* of the messages and their sizes is provided, then via *retr* its possible to retrieve a message using the number of the list and with *dele* to delete an email.

Note 3.4.2.1. POP3 is heavily limited due to problems with multiple users handling and always-on connections.

3.4.3 IMAP

This protocol works on port 143. In this case the emails remains on the server and may be cached by the client. All the actions are performed on the server. Ideal when you need to access it from different locations.

3.4.4 HTTP

The **webmail** allows the user to interact with emails via WEB. E.g. Gmail or Outlook.

4 HTTP

HTTP is a protocol that allows the user to request a **resource** (e.g. HTML page) that is on a server. They may contain references to other resources, therefore creating a *web*, called **World Wide Web**.

4.1 History

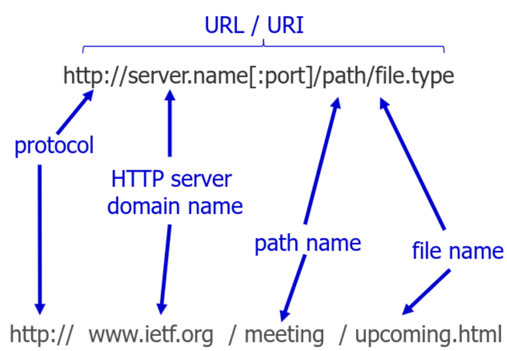
The first idea came in 1945 by Vannever Bush, with his **Memtex**, a desk containing different information categorized accordingly: **hypertex** context was born. Then in 1962 Doug Engelbart started to work on its actual implementation and by 1989 Tim Berners Lee connected that with TCP/IP and DNS protocols, effectively creating the WWW.
Today it gives access to **intelinked documents** distributed across several computers in the world, using the internet as exchange.

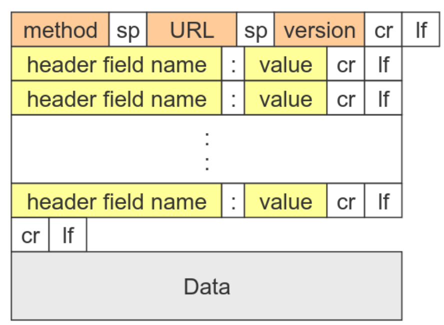
4.2 Communication

4.2.1 HTTP/1

The standard way of communicating is between a **client** (e.g. Firefox, Chrome) and a **web server** (e.g. Apache, Nginx).
The communication is handled by the HTTP protocol (usually on port 80). It's a text based **request/response** protocol.
It was **stateless** until version 2. This means that the server maintains no information about previous requests and thus the specification of the context is needed every time.

Request HTTP requests follow the **REST** API principle, allowing for performance, scalability, simplicity, modifiability, portability and reliability. The resources are retrieved via a **URL**



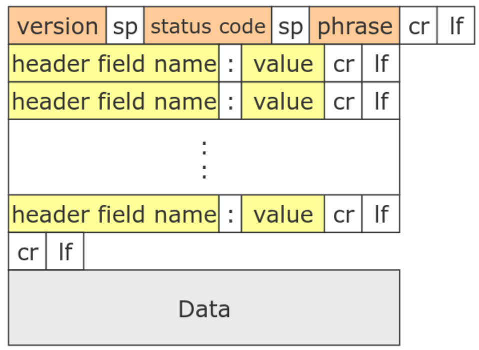


The specified commands that can be used with a URL are:

- **GET**: load a web page
- **HEAD**: load only the header of the web page, used for *debugging*
- **PUT**: store a page on the web server
- **POST**: append something to the request passed to the web server
- **DELETE**: delete a web page

Response The HTTP response contains the protocol used, the header lines and the **status code** , that can be:

- **1xx**: only for information
- **2xx**: successful inquiry
- **3xx**: further activities are necessary
- **4xx**: client error (syntax)
- **5xx**: server error



4.2.2 Web Sockets

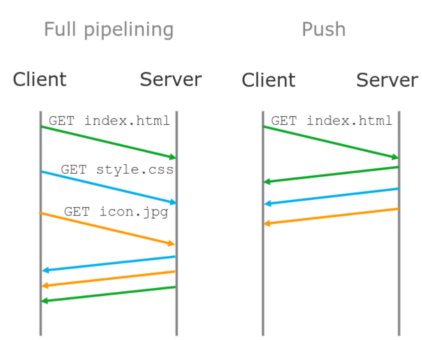
HTTP first problem is that the client needs to poll explicitly for content, causing a huge **overhead**. Thus Web Sockets were created: they allow a full duplex communication between the server and the client without the need of HTTP. It uses the same ports and it's set up using an HTTP request to "upgrade", which is then answered with a "switching protocol" response.

4.2.3 WebRTC

HTTP second problems is to enable communications between multiple browsers without creating a web server for each one of them. WebRTC implements a P2P communication that provides functions to establish media and data exchange, e.g. for videoconferencing.

4.2.4 HTTP/2

The second version of HTTP is **binary** instead of text based. It is fully **multiplexed**, associating requests and response and allowing a bi-directional stream. Therefore it can use only one connection while still granting **parallelism**. Furthermore it uses **header compression** to reduce overhead and allows server to push responses into client caches, reducing the number of requests to render web pages.



4.2.5 HTTP/3

This version uses **QUIC** protocol over UDP instead of TCP and TLS for security, avoiding *head of line* blocking.

4.3 Cookies

The main problem with HTTP is that it's **stateless**, this meaning that after every request/reply the web server forgets everything. While this is not a problem for simple browsing, it means that we cannot store user content to personalize the experience.
The solution is the **cookies**: tags stored in the web browser and set by the server so that they can be sent again to allow the latter to identify the client.

4.3.1 Structure

Cookies are stored as name-value pairs defined by the server. They can have optional parameters such as an **expiry date**.

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=497793521	15-10-18 17:00	Yes
joes-store.com	/	Cart=1-00501;1-07031	11-10-18 12:00	No
aportal.com	/	Prefs=Stk:SUNW+ORCL	31-12-18 17:30	No
sneaky.com	/	UserID=2344537333744	31-12-18 18:00	No

4.3.2 Pros and cons

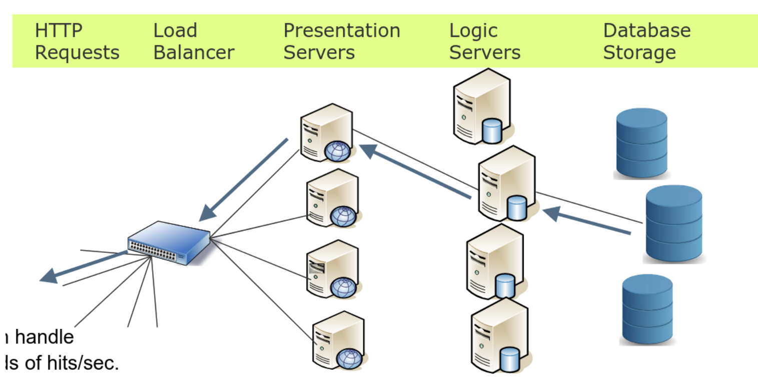
They enable **authorization**, shopping carts, recommendations and **user session state** (e.g. for web mail). The biggest problem is about **privacy**: cookies are identified by **Etags**, an opaque identifier for a specific version of a resource, and can be used to track users.

4.4 Proxy

A **proxy** is an intermediate cache between multiple clients and a server. The main goal is to have a more **efficient** page loading, improving **scalability**. It temporarily stores the pages loaded by the browsers: if the client requests it and it hasn't changed yet, it's loaded from the proxy, otherwise a new request to the server is made and the cache is updated.
It can also enable support for protocols such as FTP or Telnet without the need for a new browser implementation.
It can also work as a **firewall**.

4.5 Scaling

To handle huge loads (top 1000 websites) we use **3-tier** architecture, which separates the web server in **presentation servers**, **logic servers** and **database storage**.



If, instead, we want to deal with medium and small web servers, we usually virtualize a lot of them on a single machine and we do the multiplexing with the server URL field in HTTP.

4.6 DNS

It's possible to use DNS over HTTP by sending a request (either *GET* or *POST*) to the DNS server. It improves privacy and security but the user looses control.

5 SNMP

The Simple Network Management Protocol allows the management of devices and services using a simple datagram service with a **client-server** architecture:

- **Agent**: process continuously running on each managed node collecting information
- **Manager**: process running intermittently on a management workstation that requires information about the devices in the network

There may also be a **proxy agent** that integrates non-SNMP capable systems

5.1 Motivation

If we have problems in the network we need a **management tool** to be able to correctly identify them and their causes. In particular the tool needs to manage:

- **Performance**: measure and analyze network performance to provide good network service
- **Configuration**: monitor or modify configuration settings or HW or SW elements
- **Accounting**: measure network utilization parameters per user or group of users
- **Fault**: detect, log, notify users and automatically fix problems while running
- **Security**: control access to network resources according to local guidelines to avoid sabotage and unauthorized access to sensitive information

Observation 5.1.1. It should be achieved **remotely** over the existing network, meaning a protocol above IP.

5.1.1 Goals

The main goals of network management are:

- **Monitoring** HW equipment
- **Statistics** of network usage
- Remote **diagnostics**
- **Protected** and **safe** networking
- **Efficient** internetworking
- Simple model of **network status**
- Gather data for **network planning**

5.2 Overview

A network management framework is based on three building blocks:

- **SNMP**: defines **format of messages** exchanged by management systems and agents and **basic operations**
- **SMI**: Structure of Management Information specifies how the monitored information is structured, defining the objects that SNMP protocol accesses over the network
- **MIB**: Management Information Base describes the concrete managed objects and is an open concept for data storage. It may be public (RFC) or proprietary.

5.2.1 History

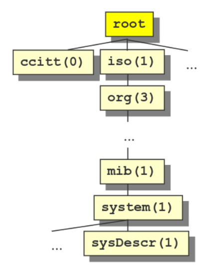
There have been three major versions of SNMP during history:

- **SNMPv1**, 1998, was designed originally as an interim solution but became the standard. It had very weak security model with complex bulk requests but was simpler than CMIS/CMIP
- **SNMPv2**, which was then split in
  - *SNMPv2u*: user-based security
  - *SNMPv2**: user-based security and additional features
  - *SNMPv2c*: without security but with *GetBulk* operation
- **SNMPv3**, the current one, that provides an advanced security model: now each message has security parameters, integrity and authentication.

5.3 Managed objects

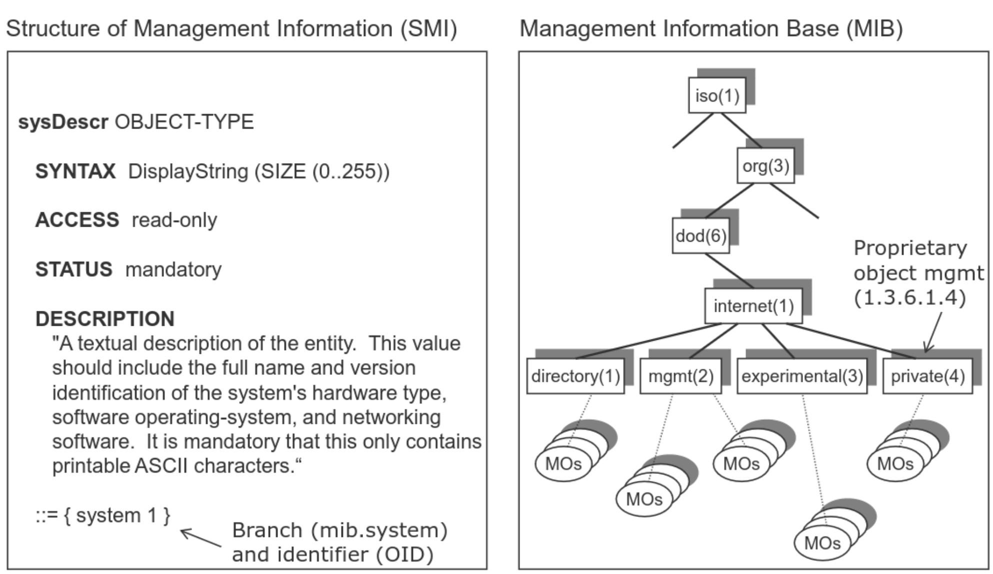
An *agent* monitors the network resources that are abstracted as **managed objects**. Each one has a **unique ID** and a **name** and models various property of the resource. Its standard components are:

- **Common prefix + Unique name**: e.g. *iso.org.dod.internet.mgmt.mib.system.sysDescr*
- **Syntax**: simple data types *integer*, *string* and *array*
- **Access rights**: *read-only* or *read-write*
- **Status**: *mandatory* or *optional*



MIB The Management Information Base is a distributed virtual database that hosts the collection of managed objects that belong to the same context. It defines the capabilities of the device that can be managed.
In particular **MIB-2** defines the generics for all manageable internet devices. Its prefix is *iso(1).org(3).dod(6).internet(1).mgmt(2).mib2(1)*. Some examples are *batteryAgingNotification* with OID *1.3.6.1.2.1.233.0.5* and *sysLocation* with *1.3.6.1.2.1.1.6*.
Another specific MIB is **RMON** for Remote Monitoring, it has more advanced functions such as gathering of statistics, alarms, events, on-board evaluations.

MIT Each managed object has a unique position in the Management Information Tree, thus providing a unique reference.



5.3.1 Encoding

Since different systems use different data representation, it's necessary to recode data while maintaining its meaning. To do that the message is first composed in ASN.1 syntax and then transferred using BER.

ASN.1 Abstract Syntax Notation One is an ISO standardized language for representation-independent specification of data types. It's used in SNMP to describe managed objects. It consists of:

- **Elementary** types, such as *boolean*, *integer*, *bitstring*, ...
- **Structured** types:
  - **Sequence**: ordered list of data types
  - **Set**: unordered set of data types
  - **Sequence Of**: like *array* in C
  - **Set Of**: unordered set of elements from the same data type
  - **Choice**: like *union* in C

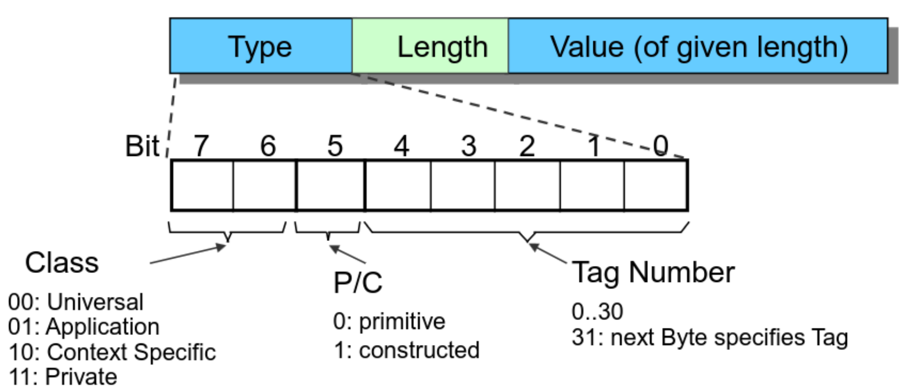
Example 5.3.1. Some types defined by ASN.1:

```
- INTEGER
  - signed 32-bit int
- OCTET STRING
- OBJECT IDENTIFIER (OID)
```

and some defined by SMI:

```
- IpAddress
  - OCTET STRING of size 4, in network byte order
- Counter
  - unsigned 32-bit int (rolls over)
- Gauge
  - unsigned 32-bit int (will top out and stay there)
- TimeTicks
  - unsigned 32-bit int (rolls over after 497 days)
- Opaque
  - used to create new data types not in SNMPv1
- DateAndTime, DisplayString, MacAddress, PhysAddress, TimeInterval, TimeStamp,
   TruthValue, VariablePointer
  - textual conventions used as types
```

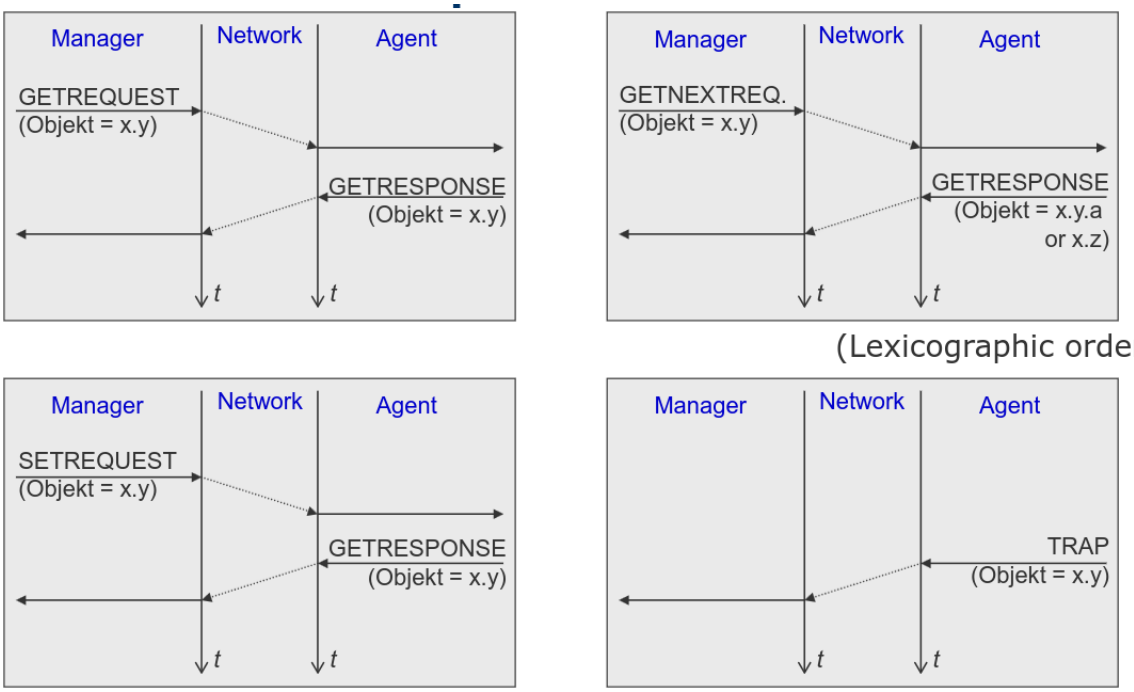
BER ASN.1 content is then converted into smaller binary data that follows the Basic Encoding Rules, like source code is converted to machine code.



5.4 Operations

SNMP has five main operations:

- **GetRequest**, **GetNextRequest**, **SetRequest** and **GetResponse** that are initiated by the SNMP manager
- **Trap** that allows an agent to push information to the manager
- **GetBulkRequest** is a series of *GetNextRequest*



SNMP works on a well-known UDP port: 161 for everything except of *Trap* that is on 162.

5.4.1 Packet format

The packet is divided in:

Version	Community	Command dependent PDU part

- **Version**: version number of SNMP
- **Community**: string used for authentication, transmitted in plain text
- Command dependent**PDU**:

PDU-Type	Request ID	Error Status	Error Index	Object name 1	Object value 1	…	Object name k	Object value k

  - **PDU-Type**
  - **Request Id**: identification of pending request
  - **Error Status**, based on the result of the query
  - **Error Index**: reference to the variable binding pair that caused the failure
  - Variable length list of pairs of (Object Name, Object Value)

0	noError
1	tooBig
2	noSuchName
3	badValue
4	readOnly
5	genErr

5.5 Tools

There are different tools:

- **Command line** like *snmpget*, *snmpnext*, etc that allows the generation and decoding of SNMP data units and sometimes also support for MIB files
- **MIB Browser**
- **Management Platforms** like *CiscoWorks* and **Nagios** (open source)

All of the tools need to do the **discovery** and **polling** of the network topology through ICMP, SNMP and HTTP, among diffent devices with common interfaces but also vendor specific functionalities.

5.6 NETCONF

SNMP is used for monitoring only and therefore a tool for configuration is needed. NETCONF is based on XML and messages are exchanged through a secure transport protocol. The included operations are:

- **get**: retrieve running configuration and device state information
- **get-config**: retrieve all or of a specified configuration datastore
- **edit-config**: edit a configuration datastore by creating, deleting, merging or replacing content
- **copy-config**: copy an entire configuration datastore to another configuration datastore
- **delete-config**: delete a configuration datastore
- **lock**: lock an entire configuration datastore of a device
- **unlock**: release a configuration datastore lock previously obtained with the *lock* operation
- **close-session**: request graceful termination of a NETCONF session
- **kill-session**: force the termination of a NETCONF session

An extension of NETCONF is **RESTCONF**, which allows accessing data defined in **YANG**. YANG (Yet Another Next Generation) is a data modeling language for the definition of the data. It can be used for configuration data, status of devices, events or notification. It's protocol independent but can be converted into XML or JSON. It's **modular** and represents data structures as XML tree, with many data types.

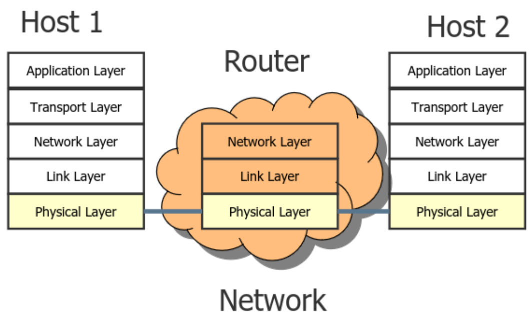
5.6.1 NETCONF vs SNMP

Let's analyze the main differences:

- **Security**: NETCONF offers more granular and flexible access control mechanism, it runs over SSH or TLS (RESTCONF on HTTPS), while SNMP lacks proper encryption and authentication
- **Structure**: NETCONF uses structured data models to define the configuration and operational state of network devices, a clear and standardized way that reduces misconfigurations. On the other hand, SNMP is less structured and less intuitive (tree structure), often requiring complex OIDs and more prone to errors
- **Functionality**: SNMP can only retrieve device data while NETCONF can also modify or replace configurations

6 Physical

In the network we have the **hosts**, end systems hosting user applications, and the **routers**, intermediate systems providing network connectivity.



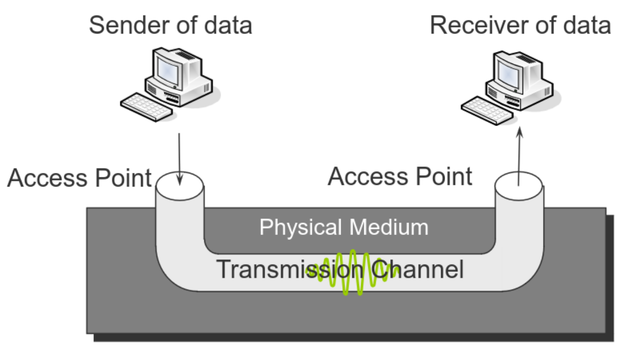
The hosts have the full stack implementation while the routers only up to the Network Layer.

6.1 Signals

Definition 6.1.1 (Signal)**.** *A signal is the physical representation of the data. It can be:*

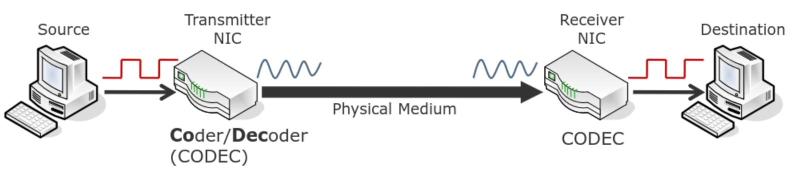
- ***Analogue****: sequence of continuous values*
- ***Digital****: sequence of discrete values*

Data is converted to signal which is then sent over the **transmission channel**, which is composed by **access points** and a **physical medium** (e.g. copper).



Since computers deal with digital signals, transmitting one bit via at a time a given medium, they need:

- **Quantization**: convert from digital signal to analog signal and vice versa
- **Sampling**: must rely on periodical measurements of the physical medium



Observation 6.1.1. We have different challenges during the transmission of data:

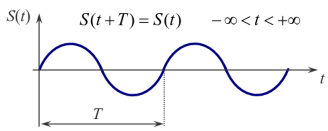
- **Internal**: **collision** and **synchronization**
- **External**: **noise**

6.1.1 Periodic signal

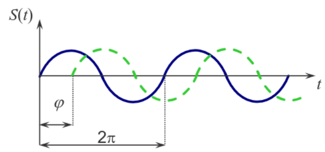
Periodic signals are the simplest signals. They take the following parameters:

- **Period** T
- **Frequency** $f = \frac{1}{T}$
- **Amplitude** $S(t)$
- **Phase** φ

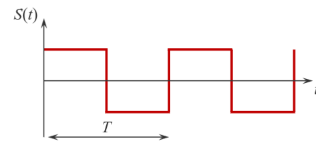
Example 6.1.1. Some examples of signals:



(a) Sine wave



(b) Phase



(c) Square wave

Fourier Analysis Like the composition of functions, it's possible to **compose** signals, generating new ones. In fact, per the **Fourier Analysis**, any period function can be constructed as the sum of a number of sines and cosines, resulting in a **Fourier Series**.

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t) \tag{5}$$

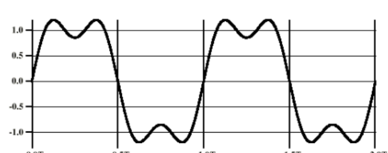
Fourier Transform is a mathematical transformation used to transform signals between time domain and frequency domain. Exists also in two dimensional space.

Distortion If all Fourier components were equally diminished the resulting signal would be reduced in amplitude but not distorted. Unfortunately all transmission facilities diminish different components by different amounts, introducing **distortion**.

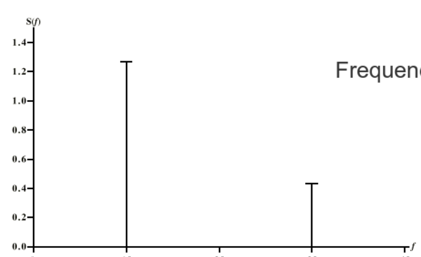
Frequency domain The frequency domain is described by:

- **Spectrum**: the range of frequencies a signals consists in
- **Bandwidth**: width of the *spectrum*. In theory many signals have infinite bandwidth. The **effective bandwidth** is the narrow band of frequencies where most of the energy is contained.

Example 6.1.2. In the following signal we have a **spectrum** from f to $3f$ and a **bandwidth** of $2f$.



(d) Time domain



(e) Frequency domain

6.1.2 Bandwidth

We have two possible digital signal:

- **Binary**: two possible values, 0 and 1
- **Multilevel**: more than two possible values (e.g. ternary, quaternary)

Definition 6.1.2 (Symbol rate). *Number of physical signaling events per unit of time on the transmission medium. The unit of measure is a **baud**.*

Definition 6.1.3 (Data rate). *Rate of bits decoded from symbol rate per unit of time. The unit of measure is $\frac{bit}{s}$. There are two cases:*

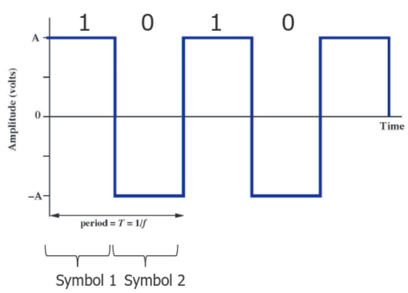
- ***Binary** signals with frequency v, each signaling event codes one bit*

$$Data\ rate = v$$

- ***Multilevel** signals with n possible values*

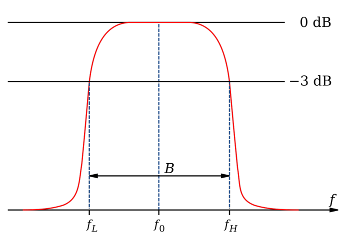
$$Data\ rate = v \cdot \log_2(n)$$

Example 6.1.3. In this image we have a square wave with a negative (0) and a positive (1) pulse



The duration of a symbol is $\frac{1}{2} \cdot T = \frac{1}{2f}$, hence the **symbol rate** (which in this case is equal to the **data rate**) is $2f$ bits per second.

Definition 6.1.4 (Bandwidth). *The bandwidth of the medium is the highest f_H minus lowest f_L frequency which can be transmitted over this medium (in Hz). f_0 is the center frequency.*



A **square wave** is composed of an infinite number of **harmonics**, where the amplitude of the kth one is $\frac{1}{k}$.

$$s(t) = A \cdot \frac{4}{\pi} \cdot \sum_{k=1, k\ \text{odd}}^{\infty} \frac{1}{k} \sin(2\pi k f t) \tag{6}$$

While theoretically we would need an infinite bandwidth to get a square wave, the medium limits the number of harmonics.

6.2 Transmission

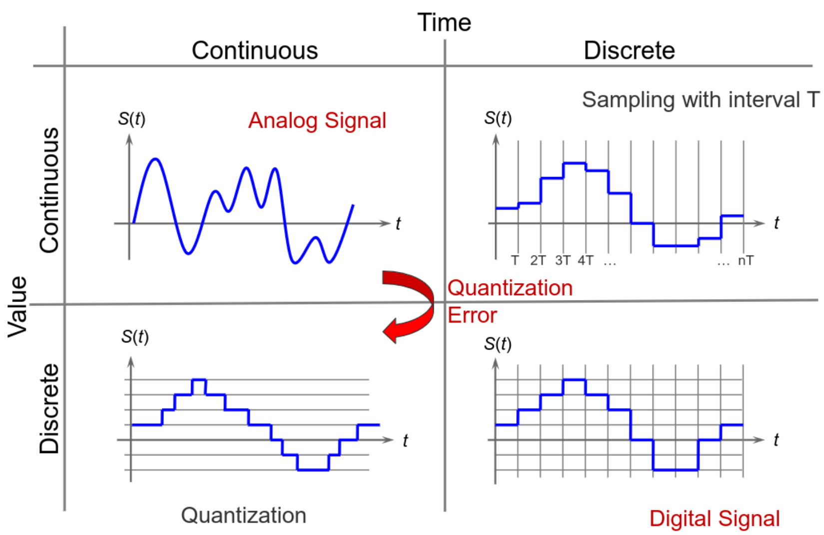
The fundamental problem of communication consists in reproducing on one side exactly or approximated a message selected on the other side. The ideal transmission would be a square wave while the actual one is an approximation.
To get from an analog to a digital signal, there are two main operations:

- **Quantization**: from a continuous value we get a discrete one with an error
- **Sampling**: periodical measures at a given rate with an interval T

  Theorem 6.2.1 (Nyquist Sampling Theorem)**.** To allow the reconstruction of the original analog signal it is sufficient that the sampling frequency f_s is such that

  $$f_s > 2W \tag{7}$$

  where W is the bandwidth in Hz.



To get an analog signal from a digital one we perform the **coding** operation, where the quantization intervals are assigned to a binary code and then transmitted.

6.2.1 Channel capacity

To define the **capacity** of an analog physical channel, we need to consider that there is always some **noise**. This means that the capacity will be **finite** and will depend on several parameters, usually:

- **Additive**: the noise value is added to the signal and that's what the receiver gets
- **White noise**: independent, random noise values with constant spectral density
- **Gaussian**: probability distribution of the amplitude of random noise values

While on the analog signals noise will cause degradation of the quality, on digital ones it causes **bit errors**. It is possible to reduce the effect of noise by boosting signal amplitude, but requires energy and causes more interferences.
Transmission impairments essentially come from:

- Signal **attenuation** and **attenuation distortion**
- **Delay distortion**
- **Noise**
  - *Thermal* noise
  - *Intermodulation* noise
  - *Crosstalk*
  - *Impulse* noise

Definition 6.2.1 (BER)**.** *Bit Error Rate is a metric for bit errors. It depends on the* ***environment****, on the* ***communication medium*** *and the* ***length*** *of the transmission line (higher frequencies attenuated stronger than lower, different frequencies have different speed).*

$$BER = \frac{Number\ of\ erroneous\ bits}{Number\ of\ transmitted\ bits} \tag{8}$$

Theorem 6.2.2 (Shannon Theorem)**.** A channel with W bandwidth, P average signal power and N average noise power has a maximum data rate of:

$$\text{maximum data rate} = W \cdot \log_2(1 + \underbrace{\frac{P}{N}}_{\text{SNR}^1}) \tag{9}$$

Even if we don't consider noise, throughput is still **finite** due to:

- **Quantization** at the transmitter and **discrete levels** of the signals
- **Sampling** at the receiver

While Shannon Theorem gives us an upper bound for the data rate, we can use Nyquist theorem to calculate the amount of discrete signals needed to achieve that.

Theorem 6.2.3 (Nyquist theorem)**.** Given a channel of W bandwidth and n discrete levels of the signal, the maximum data rate is

$$\text{maximum data rate} = 2W \cdot \log_2(n) \tag{10}$$

6.3 Data encoding

To transmit individual bits there are two options:

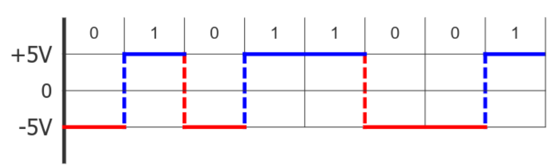
- **Baseband**: the original data is transmitted "as is" over the medium, requiring **data encoding**
- **Broadband**: the data is transmitted by **modulating** it onto a **carrier** analog signal

A data encoding technique needs:

- **Robustness**: tolerance to distortion
- **Efficiency**: high transmission rate, achieved using coded words (binary, ternary, quaternary)
- **Synchronization** with receiver: less opportunities for out-of-sync. Achieved by frequent changes of voltage level regarding to a fixed cycle. Needs to avoid direct current: positive and negative signals should alternatively arise. Bipolar/Unipolar encoding.

6.3.1 NRZ

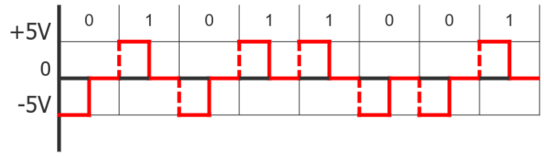
Non Return to Zero is a simple approach where 1 is coded with a positive voltage ($+5V$) and 0 as a negative one ($-5V$). It's very **simple** and the smaller the clock pulse, the higher the data rate. It's prone to **loss of synchronization** and has **direct current** during long sequences of the same bit.



[1] Signal to Noise Ratio

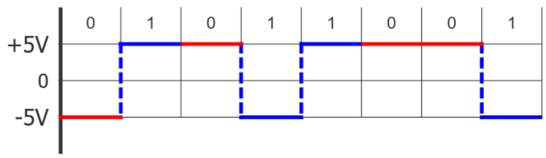
6.3.2 RZ

Return to zero works on the same principle of NRZ but after each bit the signal goes back to zero first. This way, the signal is **self-clocking** and there is no direct current. It needs twice the bandwidth.



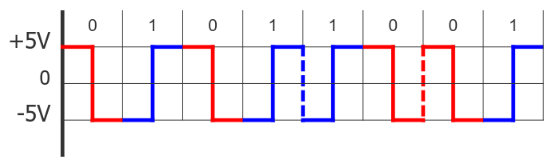
6.3.3 Differential NRZ

Similar principle of NRZ but 1 is encoded as a **voltage level change** and 0 as a missing change, thus having the disadvantages of NRZ only for a sequence of zeros.



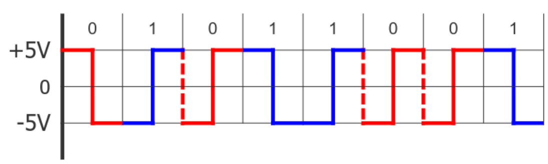
6.3.4 Manchester code

With each code element the clock pulse is transferred: voltage level change occurs in the middle of each bit. 0 is encoded as voltage level change from positive ($+5V$) to negative ($-5V$) while 1 as voltage level change from negative ($-5V$) to positive ($+5V$).
Clock synchronization happens for each bit and the end of the transmission is easily recognizable. The capacity used is only half.



6.3.5 Differential Manchester code

A variant of the Manchester code. 0 is encoded as a voltage level change while 1 as a missing one.



6.3.6 4B/5B

To improve the efficiency of the Manchester code, this techniques codes a hexadecimal character of four bits in five, avoiding long zero blocks. Uses the same principle of the differential NRZ. Allows some combinations for control information. The transmission provides clocking.
It's used in the USB and FastEthernet context and in GigabitEthernet with different variants (8B/10B, 64B/66B, . . .).

Name	4b	5b	Description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
I	-NONE-	11111	Idle
J	-NONE-	11000	Start #1
K	-NONE-	10001	Start #2
T	-NONE-	01101	End
R	-NONE-	00111	Reset
H	-NONE-	00100	Halt

6.4 Modulation

To transmit with **broadband** we need to modulate the signal, that being **shaping** a carrier frequency via the baseband signal.

$$s(t) = A \cdot \sin(2 \cdot \pi f t + \varphi) \tag{11}$$

6.4.1 ASK

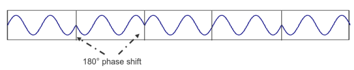
Modulation of the **amplitude** A. It's easy to realize and doesn't need much bandwidth but it's not robust against distortions. Often used in optical transmissions.

6.4.2 FSK

Modulation of the **frequency** f. It was the first used in data transmission using phone lines. Needs a lot of bandwidth and it's a waste of frequencies.

6.4.3 PSK

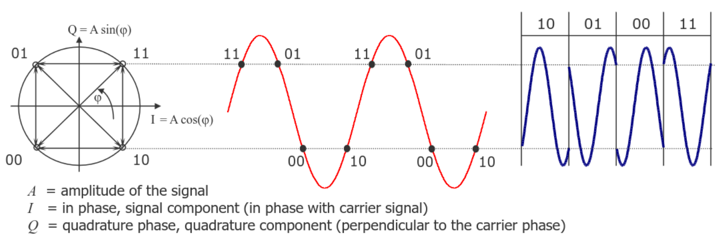
Modulation of the **phase** φ. It has a complex demodulation process but it's robust against disturbances.



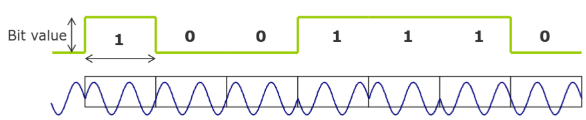
Also called **Binary** PSK.

6.4.4 PSK variants

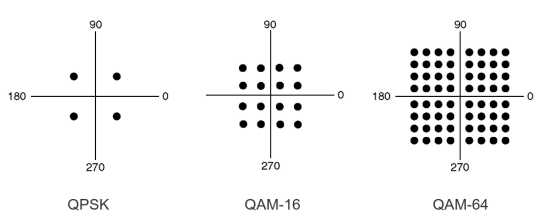
Quadrature Phase Shifting Key (also 2B1Q) allows the shifting between 4 phases, allowing for 4 states and thus 2 bits at a time (doubling the data rate).



Differential BPSK works with two different phases like PSK but it shifts only if 1 is the next bit.



Quadrature Amplitude Modulation is a combination of ASK and QPSK. $n > 2$ bit can be transferred at the same time. Bit error rate increases with n but is still less than similar techniques. Used frequently in wireless communication.

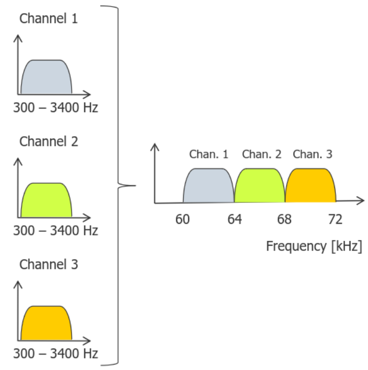


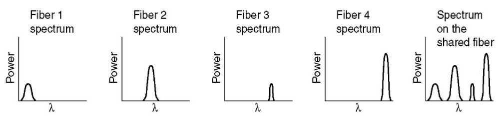
6.5 Multiplexing

Since lines are expensive it's important to share the resources. Multiplexing is a technique that provides simultaneous transmission over a single medium.

6.5.1 FDM

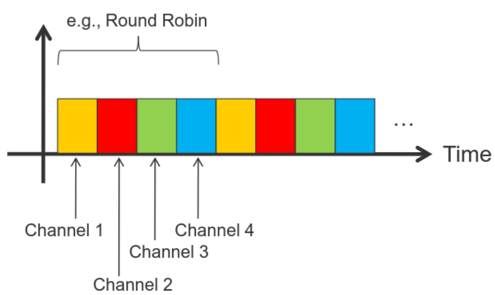
Frequency Division Multiplexing divides the frequency spectrum in frequency **bands**, which are used exclusively and simultaneously. When used for optical transmission is called **Wavelength Division Multiplexing**.



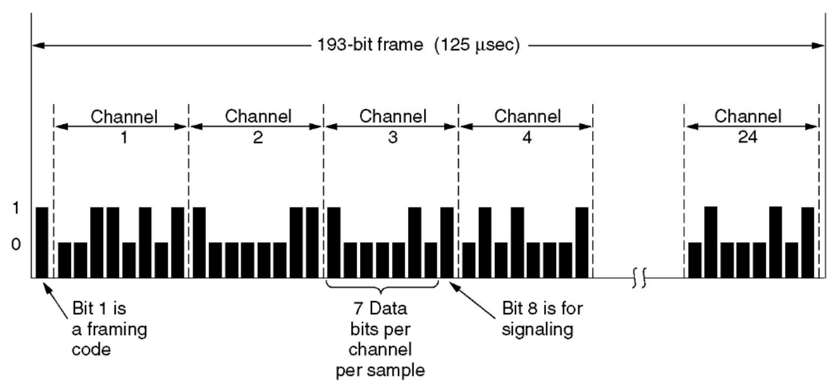


6.5.2 TDM

Time Division Multiplexing divides time into slots of fixed or variable length. Each timeslot represents one sub-channel.



A classical TDM is the T1, having 24 channels in parallel with 8bit per channel (one is control), a 193 bit frame that lasts for 125μsec for 1.554Mbit/s.

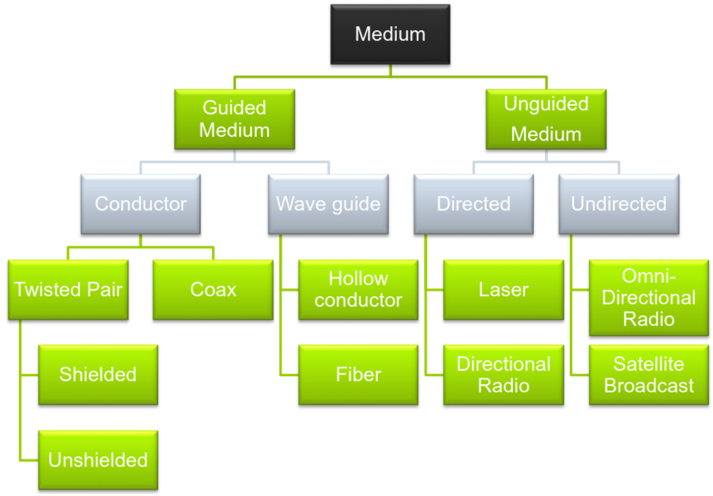


It's possible to multiplex T1 into higher carriers to get T2, T3, ...

Note 6.5.2.1. Multiplexing techniques plus algorithm that control how to do it result in **Multiple Access** technologies (e.g. FDMA, TDMA) on layer 2.

6.6 Physical media

There are many different **medias**.



And different type of **networks**, classified over the size of their scope.

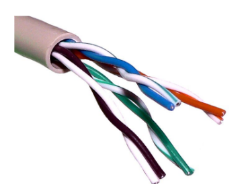
Name	Scope	Example
Body/Personal	1mt	Body
Local	10mt-100mt	Room, Building
Metropolitan	1Km-10Km	Campus, Town
Wide	100Km-1000Km	Country, Continent
Internet	10000Km	Planet

6.6.1 Electromagnetic waves

Electromagnetic waves are used to transmit the signal over cables and wireless. In a vacuum they travel at the speed of light c but in copper or fiber they slow down at about $\frac{2}{3}$ of c.
The **fundamental relationship** between wavelength λ, frequency f and c:

$$\lambda \cdot f = c \tag{12}$$

6.6.2 Guided medium

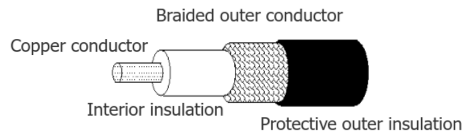


Twisted pair This medium transmits data through **electrical signals**. Electromagnetic signals from the environment can disturb it, thus it's necessary to have **insulation** and **twisting** (also done at different rates to reduce **crosstalk**). It's cheap and simple and it's used both for digital and analog signals. They have a bit-error rate of $\approx 10^{-5}$.
They are divided in:

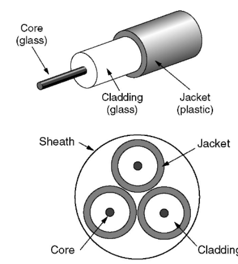
- **Unshielded** (U)
- **Foil shielded** (F)
- **Screen shielded** (S)

The **shield** can be individual, overall or both. They are also divided in **categories** based on their shielding level and maximum speed.

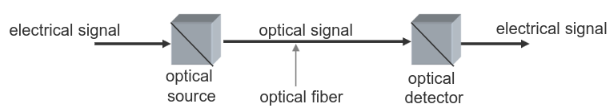
Coaxial It's a copper cable with braided outer conductor to reduce disturbances and interior insulation between them. Has a bit-rate error of $\approx 10^{-9}$, has higher data rates over longer distances compared to the twisted pair and has a better signal quality.



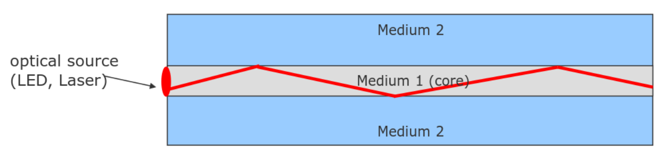
Optical Fiber Huge capacity with nearly unlimited data rate. It's insensitive to electromagnetic disturbances. Has a good signal-to-noise ratio. It's smaller and lighter and has a bit-error rate of $\approx 10^{-12}$.
The structure of an optical transmission system has:



- **Light source**: converts electrical into optical signals, 1 is a light pulse and 0 is no light pulse
- **Transmission medium**, the optical fiber
- **Detector**: converts optical into electrical signals

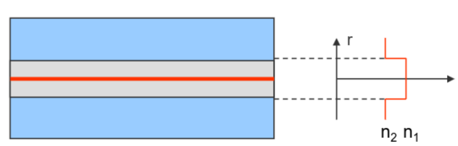


The cable has a **core** made of optical glass (super thin), an internal **glass cladding** and a protective **plastic covering**. The transmission takes place in the core, which has a high **refractive index** (refraction effect relatively to vacuum), where a ray of light is reflected between the two mediums.

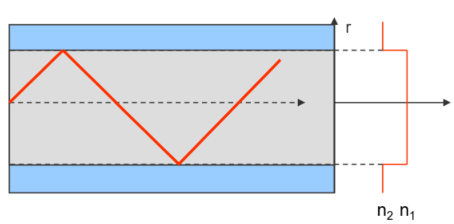


There are three types of fiber cables:

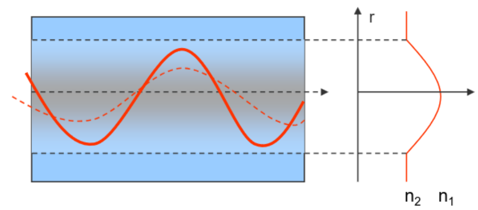
- **Single mode**: with a diameter core of $8 - 10\mu m$, all rays can go in only one direction and therefore having no dispersion (homogeneous signal delay). Expensive due to small core.



- **Simple multimode** (step index): core diameter of $50\mu m$, uses different wavelengths with different delay signals and therefore has a high dispersion



- **Multimode** with gradient index: same as the simple multimode but the refracting index changes continuously, hence having a low dispersion



Radiation sources Radiation sources can be of two types:

- **LED**: cheap and reliable, has a broad wavelength spectrum thus a **high dispersion** (small range) and a **low capacity**
- **Laser**: **expensive** but with **high capacity** and a small wavelength spectrum and thus **high range**. They are sensitive to **temperatures**.

On the other side of the signals there are **photodiodes**.

Attenuation The ray of light is increasingly weakened along the medium due to **absorption** and **impurities** in it.

Dispersion Rays of light spread in the medium at different speeds, refractive index in the medium is not constant.

6.6.3 Home connection

There are multiple solutions to connect homes to the internet:

- Existing **phone lines** local loop (DSL). It was the first approach. It needed a MoDem (Modulator and Demodulator) to convert digital data into analogs and viceversa. Today uses the whole spectrum of the copper cable and modulates through **Discrete Multi-Tone** or **Carrierless Amplitude Phase**. Calls are now over IP. The main standards are VDSL and VDSL2
- Existing **cable TV** network connections, using coaxial and upstream multiplexing through **Cable Modem Termination System**. Uses the **DOCSIS** standard.
- Deployed **cellular networks**
- Existing **powerline connections**
- **Satellite** communication
- **Microwave** links
- **Fiber-to-the-home**

Discrete Multi-Tone Modulation Uses multiple carriers where each channel uses a suitable optimal modulation method (QAM). Channels in high frequencies have a lower quality over long distances.