DATABASES

Database

**Container used to store information** managed by the information system (efficient = no redundancy, effective = fast).

It contains files (saved on mass memory) logically organized to represent reality.

* **Small** db --> stored on a **file system**,
* **Large** db --> hosted on **pc clusters** or **cloud storage**.

Actions

* Search data,
* Retrieve data (generate reports),
* Edit/delete data…

Tables

Db are organized in **tables**, **lists of rows and columns**.

Record (row)

**Meaningful and consistent way to combine information about something**.

Field (column)

**Single item of information, one that appears in every record**. (Key field --> holds unique data that identifies the record)

Keys

An **attribute used to sort/identify data**; they give meaning to data.

* **Primary (PK)** --> **specific choice of a minimal set of fields that uniquely identifies a row** in a relation,
* **Alternate (AK)** --> candidate key to be primary, but isn’t,
* **Foreign (FK)** --> **refers to the PK of another table**.

Schema

**Structure** (in formal language) **of a db**, or, a **set of integrity constraints imposed on a db** (they ensure compatibility).

The **schema doesn’t vary in time**.

Instance

**Collection of information stored in the db at a certain moment/instance of time** (**snapshot** of the db).

The **instance does vary in time**.

Data models

Data structures to manage and interrogate dbs. **They classify databases**.

Non-Relational db models:

* **Hierarchical** --> data arranged in a **tree-like structure**. **Nodes** = entities, **Arcs** = relations. Rigidity: **redundancy**.
* **Flat-file** --> data organized into a **single table** (**spreadsheets**).
* **Network/reticular** --> organizes data in a **graph**. Access through **several paths**. **Complex**, **no redundancy**.
* **Object oriented** --> data stored as **objects belonging to classes**.

Relational db models:

Simple and effective (better represent reality), data in tables/files in fields and records.

SQL (Structured Query Language)

Language used for queries.

DBMS

It’s a **software** system that **stores and organizes data** other than **retrieving data for apps**. Main features:

Data normalization

**< risk, <** data duplication (**redundancy**) & **< chance of** anomalies or **inconsistencies**.

User-defined constraints

Users can define **constraints** (rules) to prevent accidental damage to the db by authorized users.

Security protocols

Protects the **integrity of db**, data and records with **encryption**, user **auth**entication and authorization.

Backups

**Copy of the db data in case of loss or corruption** by any mean. Used to reconstruct dbs.

Data structuring

DBMS allows users to define clear and **hierarchical structures** to organize information.

Abstraction layers

Simplified representation of a db in the form of written description of diagram. 3 levels:

External/application level

Exposed to users and devs. Describes data as it’s seen. Provides tools for db operations (modify and see data).

Logical level

Describes all the items of interest of the app and offers detailed descriptions about records of data.

Internal/physical level

Used to store data. Implements the logical level.

Type of information stored in a db

Ex: e-commerce app

* Customer data --> emails, passwords, preferences…
* Business data --> products colors, prices, ratings…
* Relationship data --> location of store with specific product…

A good db design

Principle guide db design:

1. **Redundancy** (duplicated data) **is bad** (> wasted space & > error chance)
2. **Correctness and completeness of information** is important. If **not**, = **misinformation** (unintentional, dis… = intentional).

A good db design, therefore:

* Divides info in subject-based tables (< redundancy),
* Gives access and information to join tables data together as needed,
* Ensures accuracy and integrity of info,
* Satisfies data processing and reporting needs.

Database application

**Program whose purpose is retrieving** (**& insert, update, delete…**) **information from a db**. It facilitates **simultaneous** **queries** from **multiple** **users** (since mid-1990s, common to build db apps with web interface).

Users POV

There are also dbs for homes & small businesses, like **Access, Oracle, SQL Server, MySQL**…

Accounting apps

Used for financial data. Record liabilities, inventory, transactions between customers and suppliers…

CRM apps

Manages relations of a business with clients + marketing, sales, support for clients… goal: > sales, < costs…

Web apps

Websites as db apps. Can incorporate db functions of accounting/CRM apps.

WEB

HTTP

**HyperText Transfer Protocol** is an **application protocol** used to **load webpages** using hypertext links. It’s the foundation of data communication on the WWW.

Markup language

System for structuring a document’s content in a way that is syntactically distinguishable from the text.

HTML

It’s the **standard markup language to create web pages**. Consists of several key components, including **tags** and **attributes**, character-based data types, character references and entity references.

HTML tags come in **pairs**, called the **opening** (start) and **closing** (end) **tags**. </>

HTTPS

It’s a **protocol for secure communication over a computer network** widely used on the internet. It consists of communications over **HTTP** in a connection **encrypted by TLS** (Transport Layer Security).

Main motivation: **auth** for visited websites & **privacy** and **integrity** of exchanged data.

TLS

**Transport Layer Security** is a **cryptographic protocol** that **provides** **communication** **security** **over** a pc **network**. Es:

* Connection is private thx to **symmetric cryptography** used to **encrypt data**,
* The **identity of** communicating **parties** can be **authenticated with** **public-key cryptography**,
* **Connection ensures integrity** because the messages include an **integrity check** with a message auth code.

URL (Uniform Resource Locator)

An **URL** is a **reference to a web resource** thatspecifiesits **location on a pc network & a mechanism for retrieving it**.

Every URL has:

* **Scheme** --> (identifies the) **protocol used to access the resource in internet** (HTTP or HTTPS).
* **Path** --> (identifies the) **resource that the web client wants to access**.
* **Host** --> (identifies the) **host that the web client wants to access**.

Query string

**String** (after the path in the URL) that **provides information that the resource can use for some purpose**, like: user data, port, query, fragments…

Internet

1) Internet as a Network Infrastructure

Hardware and software that implement the network conveying data:

* **Hw** --> **physical devices** (packet switches or routers, but hosts/*end-systems* too, because they are access devices).
* **Sw** --> **components** (TCP/IP protocols).

2) Internet as a Service Infrastructure

* **Services** --> apps for web browsing (email, chats, news, socials…).
* **Service providers** --> ISP (Internet Service Providers).

Infrastructure

Set of complementary elements and activities necessary to carry out a main activity.

**Internet as a Network Infrastructure**

Internet

**Pc network which interconnects millions of hosts**/end-systems (pcs, servers, phones, webcams, TVs…) **in the world**.

End-systems

Or **terminals**, because they are **at the edges of the network**; & **hosts** because they **host network apps & protocols**.

Hosts are interconnected through **links** (physical copper/fiber or wireless) and **packet switches**.

Transmissive media vs links

Transmissive media are **physical links on which a signal goes from 1 point to another of a network**. **When 2 hosts try to contact one another**, a **link between them is established on/through the transmissive media**.

The transmission media can hold **multiple links** (communication channels) if they are set on **different frequencies**. (ADSL on band 0-4 kHz, like for Public Switch Telephone Network; data transmission on band 25 kHz – 1.1 MHz).

Packet switches

**On** the **internet** **data** is **sent** **in** **packets**. **Packets** **contain** the **receiver’s address** (info to get to it).

**Packet switches choose the path to forward packets** so that they can reach their destination.

ISP

**Internet Service Providers**, they **offer** **services** **to enable hosts to access** the **internet** (like Telecom, FastWeb…).

Each ISP has a network of packet switches connected to each other and to other ISP’s switches through links.

ISPs are organized in a hierarchical Tier structure. The **set of standard internet protocols** is the **TCP/IP**.

Packet switch networks use the IP protocol and follow conventions defined by IPv4 and IPv6 protocols.

On the other hand, hosts also use protocols that define the structure of packets & control their sending & receiving.

RFC (Request for Comments)

In those is published **documentation** about the technologies, methods and protocols **for the internet** **by** the **IETF** (Internet Engineering Task Force). **RFCs** are **identified by numbers**.

STDs (Interned Standards)

**Some RFCs** officially **designated as STDs**, **identified by an STD number** (that **doesn’t change** even if the RFC number which it refers to changes). Some are **required** (**basic protocols for internet to work**), while others are **recommended**.

**Internet as a Service Infrastructure**

**Internet** can also be described as an **infrastructure that offers app-type services** (like email, chat, VoIP, streaming, games, P2P, auth, ecommerce…). Applications offered as network services are called **distributed**, because they **involve multiple** **hosts** that exchange data. App is a program that runs on hosts, written in C, C#, Java…

So, the IaaS is made of apps and the network including all hosts running the software.

Client/Server Paradigm

Network apps = multiple sw running on different hosts that exchange data. **Data exchange** **takes place according to** **the Client/Server Paradigm** --> this says: **a client requires to communicate with a server, which can respond or not**.

APIs (Application Programming Interfaces)

Programming languages use **APIs** that provides tools for client/server tasks, such as C#’s *System.Net.Sockets* package.

Communication between client & server is always ruled by a protocol that may involve:

* **Pull** **protocol** --> (Ex: **HTTP**, used to ask & get web pages from servers) Client’s **request** **to** **receive** **data** **from** the **server**.
* **Push** **protocol** --> (Ex: **SMTP**, used to ask & send emails) Client’s **request** **to** **send** **data** **to** the **server**.

Any activity in internet involving multiple remote hosts is ruled by the TCP/IP protocol.

Communication models

Client/Server model

* **App divided** in **client program** and **server program**,
* **Client protocol** associated with **client program** in a **client host**,
* **Server protocol** associated with **server program** in a **server host** (generally ISP),
* ISP’s servers make up the infrastructure of internet services… (web apps & Dropbox for pc)? **ISP servers run programs**.

P2P model

* **App = single program**,
* **App** associated with **both client** and **server protocols**,
* **Client protocol** **of** **a** **host** **requires** **communication** **to** **server protocol** **of** **another host**,
* **Hosts run P2P apps** (like BitTorrent).

Hybrid model (like Skype)

* **Infrastructure server program** running **on ISP servers**,
* **P2P program** running **on** user **hosts**,
* **Phases**: 1) user contact server, 2) server bring the contact to another user, 3) contact set, involves now only clients.

Services

Divided in 2 categories:

Web Services

Apps that **user accesses via web apps and through a browser** (socials, Dropbox web, webmail, YouTube…).

Internet Services

**Apps** (client/server or P2P) that the **user can access through a specific program** (not browser) (WhatsApp desktop…).

TYPES OF COMMUNICATION NETWORKS

Circuit switching (ex: telephone network)

Physicalinfrastructure made of **circuit switches** and **transmission media**, **before** the **communication**, a **connection** is **established** **between** the **transmitter** **and** the **receiver** **at** a **physical** **level** (***virtual circuit***); it **stays reserved**.

Network layer protocols provide a **connection-oriented** and **reliable** service.

Packet switching (ex: internet)

The **data** is **organized in packets** and travels in an infrastructure made of **transmission media** and **packet switching** **nodes** (**PSN**). **Packet switches choose the** packet’s path considering: **info about the receiver** and **network state**. Network resources are provided **on demand** to individual packets.

Network layer protocols provide a **connectionless** and **unreliable** service.

**Types of service**

Protocols are designed to work in 2 ways:

Connection-oriented

Before communicating, a handshake is performed: the transmitter network protocol exchanges control packets with the receiver network protocol. This is done because:

* **Transmitter** --> checks if the **receiver is available**,
* **Receiver** --> **prepares** it **to receive** packets,
* (**Both** --> to agree on any parameters).

Once the **handshake** is **done**, the **communication begins**; at the **end** of which, follows a phase of demolition (connection termination) that frees the resources used for communicating.

“*Connection oriented*” as it occurs between **logical** **layers** with information saved in memory and protocols; **not** **involving** any aspect of the **physical** layer. The connection is **virtual**.

Connectionless

**No handshakes** are performed, when the transmitter has data to send, it sends it.

Reliability

Protocols can ensure reliable data delivery or not give guarantees about data delivery. The service can be:

Reliable

Ensures that **data** is **delivered** **correctly**, **without duplicates**, **lossless** and in the **right order**. If a **packet** is **corrupted**, it’s **retransmitted** by the transmitter.

Unreliable

The **service doesn’t guarantee anything**. **Corrupted packets** will be simply **discarded** by the receiver.

COMMUNICATION

Modes

Unicast --> Connection between 2 hosts (phone call),

Multicast --> Data transmitted simultaneously to a group of hosts (videoconference),

Broadcast --> Data transmitted from a host to all the other hosts of the network (TV channel),

Anycast --> Data directed to the nearest node (fewest hops away) of a target group.

Directions

Simplex --> Data flows only in 1 direction,

Half-duplex --> Both directions, but alternatively (walkie-talkie),

Duplex --> Both directions at the same time (phone call).

Symmetric

Dataflows are equal.

Asymmetric

Dataflows are different.

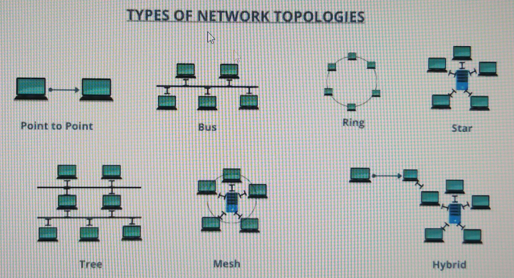
Channels

Asynchronous --> Every message is made of a single char. 1 char transmitted at a time. 1 bit of start and 1 bit of stop.

Synchronous --> Every message is made of a block of bytes.

LAN

Topologies (location, how far they can get/reach)



Typologies (based on features: PAN, LAN…)

* PAN --> personal (Bluetooth),
* LAN --> local,
* MAN --> metropolitan (WiMAX = Wireless MAN),
* WAN --> wide (national/international = internet. Typical topology = mesh).

NETWORK AND PROTOCOL STACKS

Networks are complex and in order to communicate, devices must follow network protocols organized in stacks.

Network architectures

ISO/OSI (International Standard Organization / Open System Interconnection)

(Doesn’t correspond to real network) it’s a system defined by **ISO**, specifically **Zimmermann** in **1980** as 1st step to **standardize** international **protocols** for the various levels. Concerns the interconnection of open systems, as defined by the EU Commission.

TCP/IP

Is the architecture over which **Internet is based nowadays**, defined by **Cherf** and **Kahn** in **1974**. Its goal was to give an organic structure to the already existing protocols (the 1st ARPAnet).

Hybrid model layers

Physical (ISO/OSI)

Handles the **transmission of bits** with the purpose of **transmitting them from 1 end of a physical link to the other**. Service offered at highest level to ensure that each bit at 1 is received with that value. This layer:

* Establishes the signals for the link (electrical, optic, wireless) to represent 1 and 0,
* How long a bit lasts,
* If transmission is full duplex or not,
* How to establish and end a connection,
* Shape and size of the connectors,
* Type of transmission mediums/means.

**Hubs** operate at physical level.

Datalink (ISO/OSI)

**Transfers frames from a node to another** (services and protocols for this are various, like WiFi or Frame Relay).

**Switches** and **bridges** operate at this level.

Network (TCP/IP)

**Transfers datagrams from a host to another**, **handling** also the **path to reach the destination**. The IP (Internet Protocol) is used in **IPv4** and **IPv6** versions, but also the **ICMP**.

**Routers** operate in this level.

Transport (TCP/IP)

(Only in hosts), 2 POVs:

* **Transmitting** host --> handle **data from** **application layer and transfer it to the network layer**,
* **Destination** host --> handle **datagrams from layer 3 and deliver the data to the correct application protocol**.

Used protocols: **TCP** and **UDP**.

Application (TCP/IP)

(Only in hosts), **interfaces a program to the physical network**, allowing apps to communicate with other ones. Many protocols in this layer: HTTP/S for web pages, SMTP/POP3/IMAP for emails, FTP for file transfer, DNS for IP addresses…

HISTORY OF THE INTERNET

Internet

Internet is a global network made by pc networks + physical cables (but also wireless connections like 3G/4G/5G).

There are 2 reasons for the birth of the internet:

* The necessity of a system in which info could be shared more easily (because computers in the 1960s were large & immobile, sharing info was only possible with a magnetic tape brought to the receiver (by travelling or by mail)).
* The heating up of the Cold War, because the US Defense needed to distribute info so it could still be disseminated after a nuclear attack.

Since 1960s pcs were fault-prone, large mainframe systems with terminals, in those years, *distributed adaptive message block switching* was developed (small groups of data sent along different paths to get to destination) as well as several standards later, like NCP (Network Control Program), telnet and FTP (File Transfer Protocol).

ARPANET

This led to the formation in 1969 of the 1st packet-switching WAN: the ARPANET (*Advanced Research Projects Agency Network*). Its main use was for academic & research purposes. Many of today’s protocols were made for the ARPANET, such as the TCP/IP, that enabled communication between different networks (in 1977). ARPANET changed to it on **1st January 1983**, hence the **birth** of the **Internet**. After the **2000s** the presence of internet was **widespread** in everyday life, from companies to houses.

INTERNET APPLICATIONS NOWDAYS

Nowadays, internet has various applications in every person’s daily life, such as:

Social networking

Those sites are the most popular use of the internet and people use them to connect with friends, family and others. 3 billion people use social networks; and while each platform has the same premise (), each offers unique features.

E-commerce

People use internet for online shopping: the customers purchase goods from home and compare prices between various sellers while the companies sell the items.

Online banking

With internet, banking can be made online, without going to a bank branch. It’s also easier to transfer money, pay bills & manage finances. In addition, banks also offer credit card statements and loan applications through internet.

Education & info retrieval

Internet has made education more accessible with educational websites, tutorials and online courses. People can also find every kind of info: geography, weather, news or info from one’s company’s database.

Leisure

With internet, new sectors and industries developed rapidly: gaming, music, streaming or just casual browsing.

WWW, WEBSITES & BROWSER

The WWW

The WWW (or World Wide Web) was developed by the scientist Tim Berners-Lee in 1989 while employed at CERN in Ginevra. It’s an internet-based hypermedia system for global sharing of resources, identified by URLs (Uniform Resource Locators) and interlinked by hypertext links. WWW of public domain on 30 April 1993.

Hypertexts

Documents on the internet containing links to other data (& that allow users to navigate through the document’s areas). The documents are formatted with HTML and contain embedded hyperlinks.

Hypermedia

Hypertexts that contain (or link) other types of media such as images, sounds, videos…

Website

Multiple web pages with a common theme and/or a common domain name make up a website.

Browsers

Software used to access and view websites (Chrome, Firefox, Edge, Safari, Internet Explorer… Lynx, Netscape, Mosaic early). Its primary function is to render HTML; in other words, process elements and render them in the browser window.

Early browsers supported bookmarks, URLs saved to a special list, now called favorites.

Rendering engines

Browser compatibility issues are still a problem since different browsers use different rendering engines (Chromium, Webkit, Gecko…); thus, websites could appear different on different browsers / rendering engines. Steps:

1) The rendering engine receives the HTML (+ CSS) in blocks,

2) It creates the render tree with the elements style (where, how and in which order elements appear on the page),

3) It then defines the size and position of the render trees,

4) Finally, it takes all the info and transports it to the UI rendering it.

These steps could be carried out differently depending on the browser, so it’s essential for quality assurance that every website is tested on multiple browsers.

SEARCH ENGINES

Search engine

Website that helps to search for other websites. It’s a software that creates indexes of databases based on titles, keywords or contents of files. It has an interface that allows the typing of what we are looking for into a blank field; then gives a list of results of the search.

Spiders

Search engines use multiple spiders (or crawlers), bots that search lots of pages and generate lots of data. So:

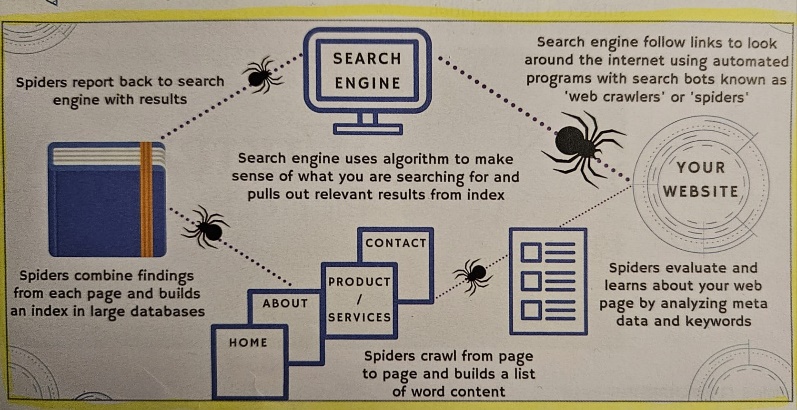
1) They search lots of pages and propagate via links in those,

2) They take note of the words in the pages, of their position and they give them a certain weight/importance,

3) Then they organize the words in lists and build indexes of them based on their weight,

4) Finally, they store all the data in a db for the users to access in the form of a list of pages.

Many algorithms decide the order in which the pages will appear (for example advertised pages are at the top).



WIKIS

Wikis

A wiki is a server program that allows users to collaborate in order to create a website and its content. Wikis provide a simplified interface, so it isn’t necessary to know HTML (advanced wikis have mechanisms to accept or reject other’s changes).

Wikis are based on collaborative trust, in fact there can be multiple contributors that edit and review a website. Anyone can change the contents of a wiki easily once they read the Text Formatting Rules.

The 1st wiki was Wiki Wiki Web; all other wikis descend from it. The best and most notorious example of a wiki website is Wikipedia, now the 5th most visited website of all. Launched on 15th January 2001, it was a complementary project for Nupedia (a free online English encyclopedia) with just an English version. As of today, Wikipedia counts 290 languages, over 40 million articles and of those 40k high-quality (In 2005, 42 tested with an accuracy similar to that of Encyclopedia Britannica; however, there is still criticism concerning the manipulation / authenticity of some articles on controversial topics).

EMAIL

Email

Email is a way of sending messages (& documents) from a pc to another. The messages are often text-based and sent from a pc connected to Internet (but also from a WAP or from a mobile phone). You can email using either:

Email client

Aka MUA (Mail User Agent) is a client/server software on the user’s pc. They send/receive emails to/from a mail server by communicating with an MTA (Transfer) which sends the messages to the relative receiver via some MDA (Delivery).

Advantage: emails are stored locally, so this implies:

* Faster access compared to a web interface,
* Offline work: ability to read downloaded messages and to write emails that will be sent when online again.

Famous: Outlook & Thunderbird. Emails have to be downloaded to be available, so user mailboxes are accessed in 2 ways:

* **POP** (*Post Office Protocol*) --> Simple protocol that allows to **download** **emails** from an **inbox** **to** a **device** (**1 device** only),
* **IMAP** (*Internet Message Access Protocol*) --> Allows user to **manage emails** **while** **stored** **on** the **server** (**multiple** devices).

Webmail

Cloud service provider accessed via browser, where emails are operated from a website, so they aren’t downloaded locally & the user doesn’t have to install fancy software on his pc (but no download or writing in offline mode).

Advantage: the user to access them everywhere with an Internet connection. Most famous webmails: Gmail & Hotmail.

Process: open browser, access email website, login with username & password, receive and send emails. When an email is sent its stored in a mailbox on the pc system of the recipient’s ISP, so when the receiver connects to the internet and opens his account, the new message will be visible and downloadable. Structure of an email address:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| gil.dis@bulz.co.uk | gil.dis = **username** | bulz = **domain name** / **service provider** | co = **domain type** | uk = **country code** |

Which is better?

Generally, webmail is better because of ease of access, setup and time; but, for people who travel in regions with 0 to low internet or for those who want max privacy/security, the email client seems a much better option.

CLOUD COMPUTING

Cloud Computing

Is the on-demand availability & delivery of resources (computing, storage capacity…) as services to end users. Cloud computing shares resources on-demand using a pay-as-you-go model.

Large clouds often distribute functions over multiple locations, called datacenters.

Characteristics

* **On-demand self-service** --> resources are required automatically when needed without human interaction,
* **Broad network access** --> resources are available over internet and accessed through client platforms,
* **Resource pooling** --> the provider’s **resources** are **pooled** to serve **multiple consumers** (dynamically assigned to each),
* **Rapid elasticity** --> capabilities are elastically provided and released (sometimes automatically).
* **Measured service** --> cloud systems **auto control/optimize** **resource use** by **measuring**/**monitoring** user’s **activity**.

IaaS

Refers to **physical/virtual machines** (run as **guests** by the **hypervisor**) offered **on-demand** (so > capabilities + < invest on hw).

Clients are able to **deploy** & **run** **sw** (like OS and apps). The client doesn’t manage the cloud infrastructure, but **controls**:

* OS,
* Storage,
* Deployed applications,
* (Maybe limited control of networking components like firewalls…)

Why use IaaS?

So, you **won't have to buy & manage** the **hw**. [IaaS is billed on a **utility computing basis** (cost = n° of resources allocated)].

PaaS

Refers to a pc platform including: OS, programming environment, db & web server. With this devs can deploy and run sw on a cloud platform without having to buy & manage the underlying hw & sw. The client doesn’t manage the cloud infrastructure + network, servers, OS or storage, but **controls**:

* Deployed applications,
* Data,
* (Maybe configuration settings for the application-hosting environment).

SaaS

Apps are installed & running in the cloud, & are accessible from various client devices through a browser/interface.

The client doesn’t manage the cloud infrastructure or the platform (network, servers, OS, storage…), but **controls**:

* The used application (not its capabilities, but its **results**/**outputs**),
* (Maybe limited user-specific configuration settings).

With SaaS the client doesn’t need to install the application; and SaaS is often referred as on-demand service and usually priced on a pay-per-use basis or with a subscription fee.

Public Cloud Computing

A **multi-tenant environment** in which **servers** **shares** the same **resources** as others tenants in cloud (usually pay-as-you-go).

Cloud services are “**public**” when delivered over the public internet & can be free or paid (with subscription).

Few differences between public & private cloud, but **security concerns** increase as services are shared by multiple users.

With a public cloud, all sw and hw and other infrastructure are owned and managed by the cloud provider

In a public cloud you share the same hw, storage & network with other cloud "tenants" & access service with browser.

Public cloud deployments often provide: webmail, online office apps, storage, testing and dev environments.

Advantages

* Low costs --> no need to buy hw or sw, you pay only for the service that you use,
* No maintenance --> service providers provide maintenance,
* Near unlimited scalability --> on-demand resources are available to meet your business needs,
* High reliability --> vast network of services ensures against failures.

Private Cloud Computing

Is a single-tenant environment that:

* Is dedicated to a single client/company (so it’s customizable & maintained on a private network),
* Is hosted & managed internally (on-site) or by an external 3rd-party (outsourced),
* (Hw is dedicated, so client always pays a fixed amount).

Private clouds are often user by government agencies, financial institutions or other companies which need enhanced control over their environment

Advantages

* More **flexibility** --> the organization can customize its cloud environment,
* More **control** --> higher privacy, because resources are not shared (outside of the company),
* More **scalability** --> more scalability than on-premise (in sede) infrastructures.

Hybrid Cloud Computing

A cloud that combines on-premise infrastructure (private cloud) with a public cloud; allowing data sharing between the 2.

Hybrid cloud is evolving to include edge workloads. By moving workloads to the edges (closer to the devices where data resides) devices spend less time communicating = < latency and ability to operate reliably in extended offline periods.

Benefits

Hybrid cloud **benefits**: > flexibility, > deployment options, security, compliance, > value from existing infrastructure.

In case of overflows (too much info to process), hybrid cloud seamlessly scales up the on-premises infrastructure with the public cloud to handle the situation (without giving third party datacenters access to data).

Hence companies can run certain workloads in the cloud while keeping highly sensitive data in their own datacenter.

Advantages

* Control --> org can maintain a private infrastructure for sensitive data & workloads that need low latency,
* Flexibility --> additional resources in the public cloud can be used when needed,
* Cost-effectiveness --> since public cloud can be scaled up, only the needed extra computing power is paid,
* Ease --> the migration isn’t overwhelming because it can be done gradually over time.

Public, Private or Hybrid?

There's no one type of cloud computing that's right for everyone.

Different cloud computing models, types and services have evolved to meet the rapidly changing tech needs of orgs.

SOCIAL NETWORKS

Social networks

Social networks are online communities of people who use websites to share interests/content and interact with others eventually fostering online communication. To be part of 1 you need to sign up, thus give personal details.

The success of socials changed the internet from a tool for archiving/transmitting data to a tool that facilitates relationships and collaboration.

Common characteristics

* Free & easy to access,
* Anyone can provide or edit content,
* Always updated,

Still, writing personal info or comments could raise privacy/identity issues.

Categories (general)

* Public social networks --> free access for everybody interested,
* Customer social networks --> access granted by companies to customers,
* Employee social networks --> access restricted to staff members.

Categories (by type of service provided)

* Networking (LinkedIn, Facebook, WhatsApp…),
* Microblogging (Twitter, Instagram…),
* Collaboration (Yahoo Answers, StackOverflow…),
* Image sharing (Pinterest…),
* Video/Audio sharing (YouTube, Spotify…),
* Live streaming (Twitch…),
* Written/Visual publishing (SlideShare, Docsity…).

Now

Nowadays socials are extremely important for businesses for them to grow and to gain returns on investments in different ways, such as with advertising or with personnel search on LinkedIn.

BLOGS & FORUMS

Blogs

A **blog** is a frequently updated website that contains dated entries arranged in reverse chronological order so that the most recent appears first (in other words an online diary).

Structure: Date & time --> Category --> Title --> Tags --> Body --> Call to Action --> Comments.

**Business usage**

Blogs are used for promotional campaigns & to measure customer satisfaction too. Business blogs’ common entries:

* How-to post --> Entry with a step-by-step description of how a product/service works,
* List post --> Used to launch different products on the market in 1 go,
* Review post --> Offers informed opinion about a particular product or service.

Forums

A **forum** (aka message board / newsgroup) is an online discussion site were people hold conversations with post messages. (It’s different from a chat room because texts are longer than 1 line of text and at least temporarily archived).

Forums has a hierarchical structure & contain sub-forums each containing several topics. Within a topic, each new discussion is called a thread, & readers can reply to it.

In some forums must login to post messages, but on most, you don't have to login to read messages. Forums can be basic (text-only) or more advanced (with multimedia support).

FILE SHARING

File Sharing (FS)

Public or private sharing of files & folders through a network (sharing also outside the network with removable media). With FS several people can use, create, modify and delete data depending on the permissions they have.

FS can also be referred to the sharing of medias on websites. A FS system has levels of access to read, view, edit… files.

FTP

With the internet the File Transfer Protocol has become widely used to access shared files in a server (authentication with username and password, but many FTP sites offer public FS or anonymous file download).

OS file sharing

Features available with OS FS (but specific options vary by OS):

* Specifying user permissions for shared folders,
* Imposing storage limits to the amount of data that users can store/share,
* (Windows Server offers a feature where admins can specify what types of data can be stored and where).

Nearly any modern OS allows FS (on Windows with SMB & sharing in ReFS, *Resilient File System*).

Internet File sharing

There are several types of internet FS, the most common are:

P2P FS  
A consumer-level tech where each pc acts as a client in a larger FS network. When 1 download something, the P2P 3rd-party sw speeds up the download process, allowing it to come from different sources (this is possible by the segmentation of shared files into smaller pieces by the 3rd-party sw).

File hosting services  
Websites which host a broad selection of online materials to enable users to download them.

Enterprise file sync & share services  
Popular for smart working because enables: cloud storage + access to its files from everywhere.  
Files created offline in the storage, will be automatically synchronized the next time that the creator goes online.  
The storage stores data in a centralized location to allow its security and backup. Any data shared between end user’s devices & the server, has to be put in a special encrypted folder (vault) to prevent its loss.

Portal websites  
Allows users to share files and folders with co-workers also outside of the organization via a shared link. These portals provide real-time FS through a browser from everywhere and with every device online.

How FS Works

To organize files in an enterprise FS tool, admins create folders and grants access privileges to users or groups; then users can access the files and download them from their storage (physical or cloud).

Accessing files via another pc works with P2P FS because each pc is client in a network; on the other hand, with server-based FS access to files can be restricted. Users edit files, the changes will be synchronized between all users.

Issues

* Even if there are legitimate uses of P2P, it has gained a reputation for being used to share **pirated** media.
* Websites that share files online sometimes hide spywares/adwares in their files (then installed on the user’s pc).