Basic Knowledge:

Angle brackets <>　　ampersand(&)　　Modulus operator(%)

The Science of secrets:

Cryptography: (creating secrets) the study of mathematical techniques to providing aspects of information security services.

Cryptanalysis: (breaking secrets) the study of mathematical techniques for attempting to defeat information security services.

Cryptology: the study of **cryptography** and **cryptanalysis**

Terminology:
Plaintext > Encryption > Ciphertext > Decryption > Plaintext

Fundamental Goals:
Confidentiality — no eavesdropping
Integrity — no unauthorized modifications
Authenticity — no spoofing or faking
Non-repudiation — no disclaiming of authorship

Brute force: try all the possibilities, Simple, exhaustive key search can be effective.

Symmetric Key Cryptography:
Algorithms use a single key for encryption and decryption.
Besides in person, not safe to share a key over internet.

Asymmetric Cryptography: aka: Public Key Cryptography

DATA Bridging:

A data bridge is **a process that connects two or more stable, predefined data stores for a limited time or on an ongoing basis**. Data bridges are used in a variety of applications such as: Converting data from an old system to a new system; Interfacing data from one system to another.

Backdoor:

A backdoor means bypassing regular authentication and/or authorization to access resources. Backdoors may be secretly added information technology by organizations or individuals in order to gain access to systems and data. Backdoor can also be an open and documented feature of information technology. In either case, they can potentially represent an information security vulnerability. Common examples:

- **Hardware**: can be hardwired into devices.
- **Operating systems:** often include remote administration tools that allow administrators and vendors to support users. These are typically secured but are an attractive target for malicious use because they are designed to gain control of a machine remotely.
- **Applications:** applications may have backdoors secretly installed to achieve malicious objectives. Backdoors are also openly installed for administrative and integration purposes. When you install software, you are placing trust in a vendor that they haven't added insecure or malicious backdoors that will leave you vulnerable.

Key Points:

- Transport Layer Security(TLS) - certificate - Certificate Authority(CA) Green lock icon indicates secure TLS connection to the user.

- Key compromise and Revocation: After a secret key has been compromised, the owner needs to revoke the certificate from CA. A compromised key attack is the use of a key that an attacker has stolen to gain access to a secured transmission. The key allows the attacker to decrypt the data that is being sent. The sender and receiver are usually not aware of the attack.

- System must validate passwords provided by users; thus passwords must be stored somewhere. Passwords should never be stored in plain text. Otherwise, if the attacker can compromise it, they can directly log-in as the user including root/administrator.

Authentication: the process of verifying an actor's identity; compare the user entered identity with the stored identity; typically parameterized as a **username** and a **secret.** A secret is *unforgeable*(cannot be faked), *unguessable*, and *revocable*(if been stolen/compromised, get rid of it, and set a new one.

Secrets Type - Something I know:

Threat Model on storing passwords:
When you build an App and choose passwords as user authentication, you need to securely store those passwords in somewhere which is difficult challenge.

Attacker's goal:
Get stored user's username and password which are stored in a file:
    On Linux system: /etc/shadow
    On Windows: c:\windows\system32\config\sam

Never store passwords in plain text, other wise once the attacker get in these files, game over.

1: Store hashed versions of passwords.

Cryptographic hash function:
Deterministic: hash(A) = hash(A)
High entropy: hash values of a tiny bit different inputs are wildly different.
Collision resistant: given a hash values, it should be extremely difficult to get the input that produce the hash value.

Dictionary attacks: Precompute the hash value of dictionary words and compare it with the files they steal. People often choose common words for passwords, so 60-70% of hashed passwords can be cracked in < 24 hours.

2: Stored salted passwords
Salt is a unique and random string that is not secret and stored in plain text in the same file with username, passwords, and hashing AlGO named by numbers: 1: MD5; 2: Blowfish; 2a: eksBlowfish;
5: SHA-256; 6: SHA-512
hash(salt + password) = hash value.

Now, even the attacker precompute the english words, nothing matches with the file they steal. It forces the attacker to brute-force each english word with the salt individually. This makes the attacker's life significantly difficult, but did not eliminate the problem.

Today's computer is very fast, and GPU computing with thousands of CPU cores, the attacker can perform parallel attack. Renting GPU from cloud is cheap $0.9 / hour.

3: Key stretching: hash the password multiple time 10 - 100.

4: Use hash functions that are designed to be slow; it takes additional parameter that is known as work factor which increases the time complexity of the calculation. It may take 100 milliseconds to compute. The extra time is nothing for users but a bottleneck for attackers.

Password Recovery/Reset: (fundamentally challenging)
Remember that company only stores “username salt hash-value”.
Their should be no way to recover your password if you ask them.
If it’s easy for you to recover the password it’s easy for the attacker to attack. Therefore, they don’t serve recovery, but offer reset; and reset is the weakest link.

Solution 1:
Knowledge-based Authentication(KBA) like your friend’s name in elementary school: it’s supposed be secret, but nowadays, they are all semi-public. You could store random string only you know.

Solution 2:
Account-based Reset: send you a link or a code to your email or phone number.

In practice: use layered authentication meaning use all of the above.
tradeoff: usability challenge; user may not be able to get their account back.

Attacker’s capability:
Password Cracking:
- Brute-forced: systematically try them all.

  - Precompute the hashes, but need huge space to store it.
- Dictionary attacks: common words in language.

Choosing passwords (Better Heuristics):
- choose passwords(16+ long)
- choose a phrase from song or move and reduce it in a way
- e.g: I double dare you, say “what” one more time —> i2Dy,s”w”omt

Have I been pwned (HIBP):
It will tell you has the password of that account ever been leaked or cracked anywhere historically. This website collects dumped passwords by the hacker. Modern browsers are incorporating bridge notification into themselves. Firefox/ Chrome will tell you your passwords have been cracked. These services have been bridged, you need to change the password, also anywhere you used this password.

Secrets Type - Something I have:
smart phone
SMS Two Factor: when login, a code will be send to your phone which only you have.
One Time Passwords: service and app(Duo Mobile) on your phone, When login, it asks the one time password on the app which changes every few minutes. It also use two factor.

Using QR code for secret sharing for Time-based One-time Password

Hardware Two Factor: USB Dongles contains the cryptographic keys.

Secrets Type - Something I am:

fingerprint / hand geometry / voice scan,
face / Retinal / iris recognition,
gait, typing cadence

Biometrics are immutable, once they are cracked, you cannot change yourself.

Telecommunication:
The transmission of information by various types of technologies over wire, radio, optical, or other electromagnetic systems. It has its origin in the desire of humans for communication over a distance greater than that feasible with the human voice, but with a similar scale of expediency; thus, slow systems such as postal mail are excluded from the field.

The transmission media in telecommunication have evolved through numerous stages of technology, from beacons and other visual signals (such as smoke, flags) to electrical cable and electromagnetic radiation, including light.

How the Internet was born:

First:
<u>a-stuttered-hello</u>

UCLA's computer talked to SRI(Stanford Research Institute)'s computer on October 29, 1969. "Lo" was the first historic message ever sent over the ARPANET, the experimental computer network built at the end of the 1960s to allow researchers from UCLA, SRI, the University of California Santa Barbara (UCSB) and the University of Utah to work together and share resources.

In 1957, Soviet sent the first satellite, cold war begins. President Dwight Eisenhower assigned to the newly established Advanced Research Project Agency (ARPA). The program succeeded with first two American satellites, then in the summer of 1958, all of its space and rocket projects was stripped by the new born National Aeronautics and Space Administration (NASA) to maximize the space effort. To avoid falling into oblivion, the agency had to quickly reinvent itself and find new goals and new sectors for its projects. With a reduced but still considerable budget, ARPA found the solution to its problems in the brand new sector of pure research (Computer Science) — a new galaxy of communication, first called ARPANET and later, the Internet.

Visionary leadership: Joseph Carl R. Licklider: if machine can take care of clerical activities, humans would have more time and energy to dedicate to “thinking”, “imaging”, creativity and interactivity. Licklider era’s computers were ranging from $500,000 to several million dollars).

To save money, they are forced to buy time–sharing computers. Time-sharing systems were designed to allow multiple users to connect to a powerful mainframe computer simultaneously, and to interact with it via a console that ran their applications by sharing processor time.

Before time–sharing systems were adopted, computers, even the most expensive ones, were bound to do jobs serially: one at a time. This meant that computers were often idle as they waited for its users' input or computation result.

If the first step was to force universities to use their funds to buy time-sharing systems, the next step was to allow access to off-site resources via other computers. In other words, to make those recourses available via a network. Because then, each task requires a machine to do. The same machine cannot multi-tasking like our smartphone does. But machines are expensive, so to save money, they need to share resources via a computer network.

Second:
<u>the-network-begins-to-take-shape</u>

In 1962: annual budget of $19 million, with individual grants ranging from $ 500,000 to $3M.

At the time, if researchers wanted to use the resources (applications and data) stored in a computer at their UCLA campus, they had to log in through a terminal. This procedure became more cumbersome when the researchers needed to access another resource, for instance a graphic application, which was not loaded on their mainframe computer, but was instead only available at another computer, in another location, let's say in Stanford. In that case, the researchers were required to log into the computer at Stanford from a different terminal with a different password and a different user name, using a different programming language. There was no possible mode of communication between the different mainframe computers. In essence, these computers would have been like aliens speaking different idioms to each other.

Until the end of the 1960s, running tasks on computers remotely meant transferring data along the telephone line. The analogue circuits of telephone network could not guarantee reliability, and too slow. Often the whole sets of information and input are lost. The solution was called packet-switching, a simple and efficient method to rapidly store-and-forward standard units of data (1024 bits) across a computer network.

In such a network, each computer would have to be different, with different specialization, applications and hardware. The next step, then, was to create that network.

Communication networks: centralized, decentralized, distributed.

In 1966, after a brief and quite informal meeting with Charles Herzfeld, then Director of ARPA, Taylor was given an initial budget of $1 million to start building an experimental network called ARPANET.

The decision took no more than a few minutes. Taylor explained:

I had no proposals for the ARPANET. I just decided that we were going to build a network that would connect these interactive communities into a larger community in such a way that a user of one community could connect to a distant community as though that user were on his local system. First I went to Herzfeld and said, this is what I want to do, and why. That was literally a 15-minute conversation.

Then, Herzfeld asked: “How much money do you need to get it off the ground?” And Taylor, without thinking too much of it, said “a million dollars or so, just to get it organized”. Herzfeld’s answer was instantaneous: “You’ve got it”.

In 1969, at UCLA, the first cornerstone of the Internet was finally laid, and the ARPANET, the first computer network was built. A list of reasons to build the network None of them were concerned with military issues. Instead, they looked towards sharing data load between computers, providing an electronic mail service, sharing data and programs and finally, towards providing a service to log in and use computers remotely.

Third:
<u>**ARPANET comets to life**</u>

The experiment was a “testing environment” that aimed to verify whether it was possible to build a computer network on a continental scale “without enforcing standardization”. Since the network was built “to overcome the problems of computer incompatibility”, it would have been ill-advised to enforce a standard protocol “as a prerequisite of membership in the network”. Instead, Roberts and Merrill argued, for a network to work efficiently, it required maximum flexibility.

The idea of flexibility is an important building block of the Internet we use today. It allows the development of different networks, with different standards, all of which are able to connect with each other. This variety of networks and the lack of enforced standardization, in time, have become a very important asset of the Internet. It has also made it a much more difficult environment to control as a whole.

As originally envisioned by Baran, the ARPANET was set to be a fully distributed network that made use of routers (small computers called Interface Message Processors - IMPs) at every node to speed up communication between computers. Each router had four critical tasks to accomplish:

- to receive packets of data from both the computers connected to it,
- break the message blocks into 128 byte packets, or 1024 bits (In his study of packet-switching, Donald Davies theorised that “the length of a packet can be any multiple of 128 bits up to 1,024 bits”. The 128-bit unit length guaranteed some measure of “flexibility to the size of packets” without ever overloading the computer while handling them.

- add the destination and the sender address, and

- use a “dynamically updated routing table”, or an updated map of the routes available in the network (“considering both line availability and queue lengths”) to send the packet over whichever free line was currently the fastest route toward the destination.

It was a sort of haven where people like Licklider worked, where dozens of graduate students and faculty members from either Harvard or MIT, free from any university duties but research, were encouraged “to do interesting things and move on to the next interesting thing”.

Fourth:
arpanet-to-the-internet

Connecting with the global network is possible because at the core of this worldwide infrastructure we call the Internet is a set of shared communication standards, procedures and formats called protocols. However, when in the early 1970s, the first four-nodes of the ARPANET became fully functional things were a bit more complicated. Exchanging data between different computers (let alone different computer networks) was not as easy as it is today. Finally, there was a reliable packet-switching network to connect to, but no universal language to communicate through it. Each host, in fact, had a set of specific protocols and to login users were required to know the host's own 'language'. Using ARPANET was like being given a telephone and unlimited credit only to find out that the only users we can call don't speak our language.

Both RFC and NWG helped shaped and strengthen a new revolutionary culture that highlights networking as the path to find the best solution to a problem, any problem. Within this kind of environment, it is not one's particular vision or idea that counts, but the welfare of the environment itself: that is, the network — a key-aspect in processes of human interaction.

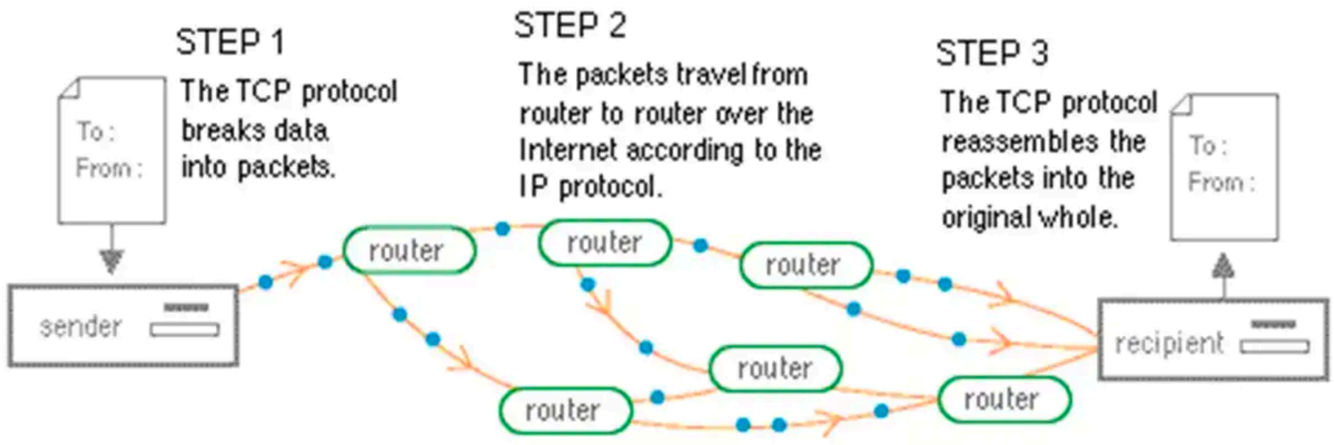
The NWG however needed almost two years to write the software, but eventually, by 1970 the ARPANET had its first host-to-host protocol, the Network Control Protocol (NCP). By December 1970 the original four-node network had expanded to 10 nodes and 19 hosts computers. Four months later, the ARPANET had grown to 15 nodes and 23 hosts.

No sign of "useful interactions that were taking place on"

The hosts were plugged in, but they all lacked the right configuration (or knowledge) to properly use the network.

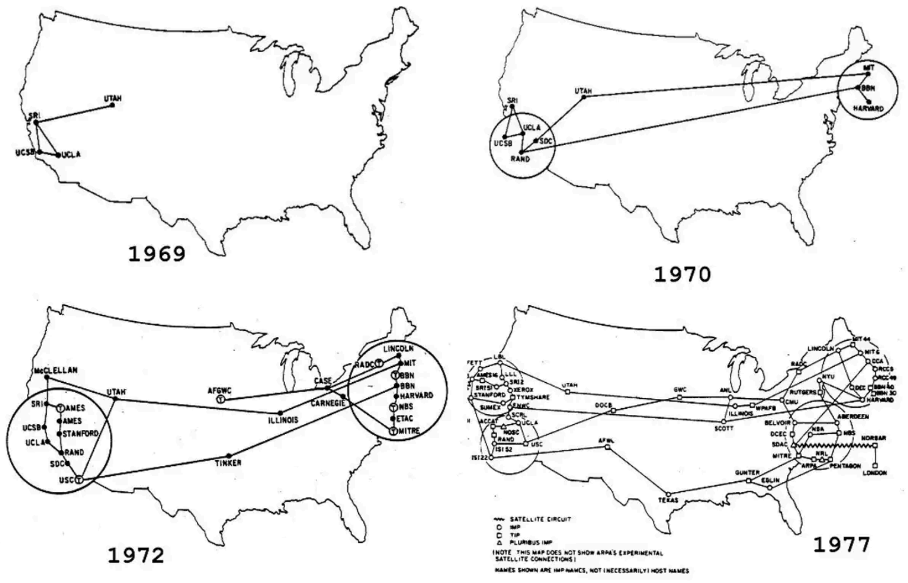
World was not noticing of packet switching. (So they did public demonstrations).

The existing Network Control Protocol (NCP) didn't meet the requirements. It had been designed to manage communication host-to-host within the same network. To build a true open reliable and dynamic network of networks what was needed was a new general protocol. It took several years, but eventually, by 1978, Robert Kahn and Vint Cerf (two of the BBN guys) succeeded in designing it. They called it Transfer Control Protocol/Internet Protocol (TCP/IP). The TCP on the sender's machine breaks the message into packets and send them out. The IP is instead the part of the protocol concerned with "the addressing and forwarding" of those individual packets. Thanks to TCP/IP the exchange of data packets between different and distant networks was finally possible.


STEP 1
The TCP protocol breaks data into packets.
STEP 2
The packets travel from router to router over the Internet according to the IP protocol.
STEP 3
The TCP protocol reassembles the packets into the original whole.
To:
From:
sender
router
router
router
router
router
router
recipient
To:
From:

Cerf and Khan's new protocol opened up new possible avenues of collaboration between the ARPANET and all the other networks around the world that had been inspired by ARPA's work. The foundations for a worldwide network were laid, and the doors were wide open for anyone to join in.

Then ARPANET consolidated and expanded — the mix of fast growth rate and lack of control could potentially become a serious issue for national security. By the early 1980s, the network was



essentially an open access area for both authorized and non-authorized users. This situation was made worse by the drastic drop in computer prices. In 1983, a young computer whiz managed to connect to the super computer at NORAD and almost start WWIII from his bedroom — “War Games”.

Then the ARPANET was effectively divided in two distinct networks: one still called ARPANET, mainly dedicated to research, and the other called MILNET, a military operational network, protected by strong security measures like encryption and restricted access control.

The ARPANET was officially decommissioned in 1990, whilst in 1995 the NFTNET was shut down and the Internet effectively privatized. By then, the network - no longer the private enclave of computer scientists or militaries - had become the Internet, a new galaxy of communication ready to be fully explored and populated.

The exponential grow: 100 Gigabytes of traffic per day ; World Wide Web (1989), to 16,000 Gigabytes per second(GBPs) in 2014.

Internet:

Client - Server world (or everyone is client and server at the same time). Computers exchange information (Webs, videos, emails) through internet. Physical infrastructure such as cables, optical fibers, modem, routers, switches, etc, connects networks (many computers) to other networks (many computers).

Server / Client:

A **server** is a software or hardware device that accepts and responds to requests made over a network. The device that makes the request, and receives a response from the server, is called a **client**.

Servers are used to manage network resources:
Once the server connects to a router that all other computers on the network use. Other computers can access that server and its features. For example, with a web server, a user could connect to the server to view a website.

Modem:

Computer information is stored digitally, however information sent over phone and cable lines are analog waves. Modem takes the analog waves, changes it to digital and delivers it to your computer.

Router:

Transfer packets between internet according to IP addresses, public or private. At home, a router sits between modem and computer, deliver information to all the desktop, laptop, smartphones.etc that are linked to it.

Network & Distributed System:

—————————————————

Socket:

Stream Socket (two way, error-free):
Telnet (Terminal Network) —> uses —> stream socket —> uses —> Transmission Control Protocol (TCP) which makes sure your data arrives sequentially and error-free. (“TCP/IP” (Internet Protocol): IP deals primarily with Internet routing and is not responsible for data integrity).

Web browsers get pages —> uses —> Hypertext Transfer Protocol (HTTP) —> uses —> stream sockets —> uses —> TCP

Datagram Socket (connectionless):
They use “User Datagram Protocol” (UDP).

A way to speak to other programs using standard Unix file descriptors; A file descriptor is simply an integer associated with an open file which can be anything (network connection, a FIFO, a pipe, a terminal…). To get this file descriptor for network communication, you make a call to the socket() which returns the socket descriptor, and you communicate through it using the specialized send() and recv() socket calls.

The **socket** is primarily a concept used in the Transport **layer** of the Internet model or Session **layer** of the OSI model.

Strictly speaking, **Sockets** is not a protocol, but rather a programming method.

A socket is externally identified to other hosts by its socket address, which is the triad of transport protocol, IP address, and port number.

Acronyms:

OSI: Open System Interconnect Model

LAN: local Area Network.

WAN: Wide Area Network.

MAN: metropolitan Area Network.
DNS: Domain Name System; translates human readable domain names (for example www.amazon.com) to machine readable IP address (for example 192.0.2.44).
MAC (media access control): media like wifi can be accessed by many hosts, sending data simultaneously causing collision which destroys data, MAC detecting collisions, avoiding collisions, recovering collisions.

Coaxial cable:

A type of transmission line, used to carry hight-frequency electrical signals with low losses.

Optical fiber:

A flexible, transparent fiber made by drawing glass(silica) or plastic to a diameter slightly thicker than human hair. It's used to transmit light between two ends over longer distances, at higher bandwidths with less loss than electrical cables. In addition, fibers are immune to electromagnetic interference, a problem from which metal wires suffer.

Analog VS Degital:

A signal is an electromagnetic or electrical current that carries data from one system or network to another. In electronics, a signal is often a time-varying voltage that is also an electromagnetic wave carrying information, though it can take on other forms, such as current, frequency of the signal. There are two main types of signals used in electronics: analog and digital signals.

An analog signal is time-varying; there is an infinite number of values within a continuous range. When plotted on a voltage vs. time graph, an analog signal should produce a smooth and continuous curve. There should not be any discrete value changes.

A digital signal represents data as a sequence of discrete values. When plotted on a voltage vs. time graph, digital signals are one of two values.

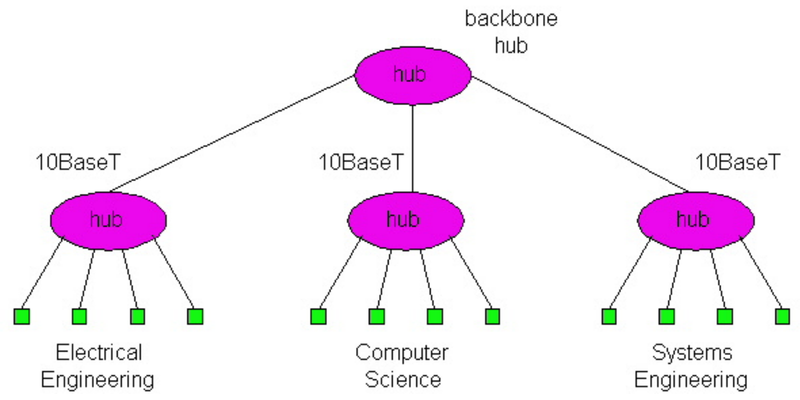
Computer Networking book:
https://www.net.t-labs.tu-berlin.de/teaching/computer_networking/index.html

Connecting LANs:

(Hubs—Bridges — Switches)

Hub:

(Operate on bits: Physical-layer devices)
The simplest way to interconnect LANs is to use a hub. A hub is a repeater that copy the input (frame's bits) and broadcast the bits to all the outgoing ports.

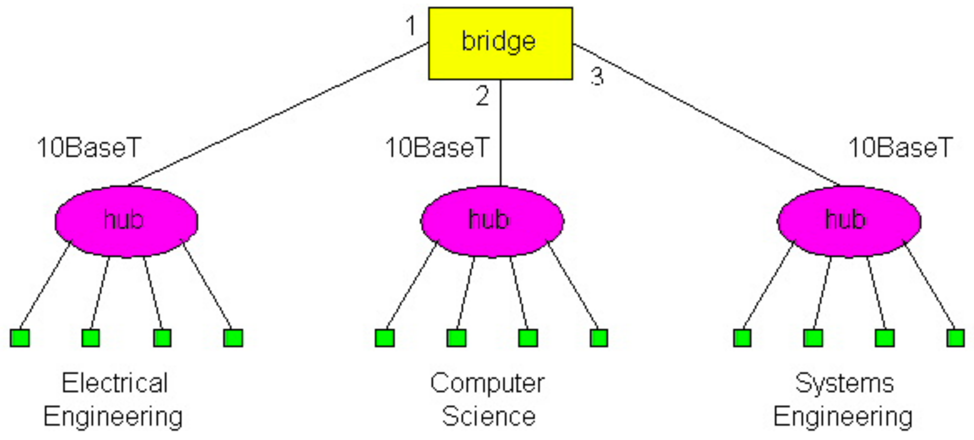


In a multi-tier design, we refer to the entire interconnected network as a LAN, and we refer to each of the departmental portions of the LAN (i.e., the departmental hub and the hosts that connect to the hub) as a **LAN segment**. It is important to note that all of the LAN segments belong to the same **collision domain**, that is, whenever two or more nodes on the LAN segments transmit at the same time, there will be a collision and all of the transmitting nodes will enter exponential backoff. If any one of the departmental hubs starts to malfunction, the backbone hub can detect the problem and disconnect the departmental hub from the LAN; in this manner, the remaining departments can continue to operate and communicate while the faulty departmental hub gets repaired.

Bridge:

(Operate on frames: Data-link layer devices)

Bridges are packet switches that filter and forward frames using the LAN destination addresses.



We again refer to the entire interconnected network as a LAN. In contrast to the multi-tier hub design each LAN segment is now an isolated collision domain. Recall that when a hub forwards a frame onto a link, it just sends the bits onto the link without bothering to sense whether another transmission is currently taking place on the link. In contrast, when a bridge wants to forward a frame onto a link, it runs the CSMA/CD algorithm. In particular, the bridge refrains from transmitting if it senses that some other node on the LAN segment is transmitting.

Filtering is the ability to determine whether a frame should be forwarded to an interface or should just be dropped. When the frame should be forwarded, **forwarding** is the ability to determine which of the interfaces the frame should be directed to. Bridge filtering and forwarding are done with a **bridge table**. For each node on the LAN, the bridge table contains (1) the LAN address of the node, (2) the bridge interface that leads towards the node, (3) and the time at which the entry for the node was placed in the table.
To understand how bridge filtering and forwarding works, suppose a frame with destination address DD-DD-DD-DD-DD-DD arrives to the bridge on interface x. The bridge indexes its table with the LAN address DD-DD-DD-DD-DD-DD and finds the corresponding interface y.

- If x equals y, then the frame is coming from a LAN segment that contains adapter DD-DD-DD-DD-DD-DD. This means that the frame has already been broadcast on the LAN segment that contains the destination. The bridge therefore filters (discards) the frame.

- If x does not equal y, then the frame needs to be routed to the LAN segment attached to interface y. The bridge performs its forwarding function by putting the frame in an output buffer that precedes interface y.

S**elf-learning:**

A bridge has the *very cool* property of building its table automatically, dynamically and autonomously – without any intervention from a network administrator or from a configuration protocol. In other words, bridges are **self-learning**. This is accomplished as follows.

- The bridge table is initially empty.

- When a frame arrives on one of the interfaces and the frame's destination address is not in the table, then the bridge forwards copies of the frame to the output buffers of all of the other interfaces. (At each of these other interfaces, the frame accesses the LAN segment using CSMA/CD.)

- For each frame received, the bridge stores in its table (1) the LAN address in the frame's *source address field*, (2) the interface from which the frame arrived, (3) the current time. In this manner the bridge records in its table the LAN segment on which the sending node resides. If every node in the LAN eventually sends a frame, then every node will eventually get recorded in the table.

- When a frame arrives on one of the interfaces and the frame's destination address is in the table, then the bridge forwards the frame to the appropriate interface.

- The bridge deletes an address in the table if no frames are received with that address as the source address after a period of time (the *aging time*). In this manner, if a PC is replaced by another PC (with a different adapter), the LAN address of the original PC will eventually be purged from the bridge table.

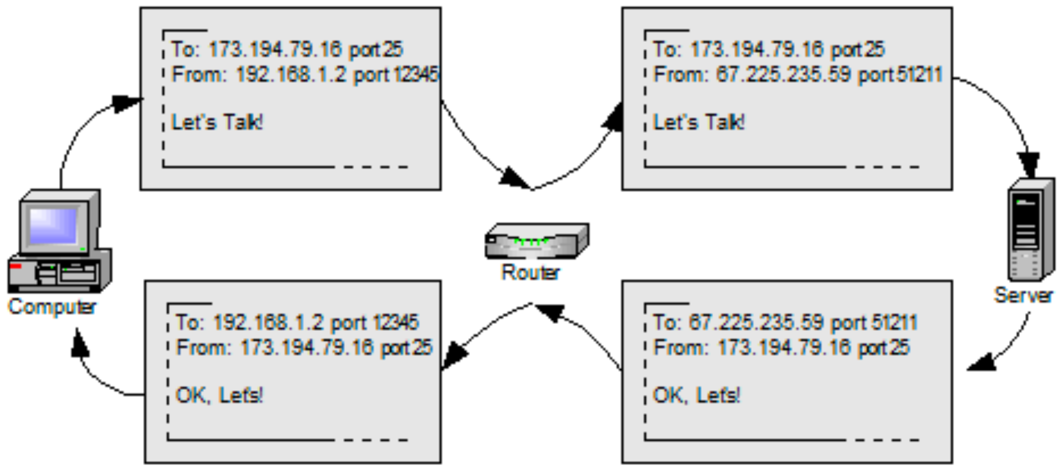
Bridges are **plug and play devices** because they require absolutely no intervention from a network administrator or user. When a network administrator wants to install a bridge, it does no more than connect the LAN segments to the bridge interfaces. The administrator does not have to configure the bridge tables at the time of installation or when a host is removed from one of the LAN segments. Because bridges are plug and play, they are also referred as **transparent bridges.**

Bridges VS Routers:

Routers are store-and-forward packet switches that forward packets using IP addresses. Although a bridge is also a store-and-forward packet switch, it is fundamentally different from a router in that it forwards packets using LAN addresses. Whereas a router is Network-layer-3 packet switch, a bridge is a Data-Link-layer-2 packet switch.

NAT:

(Network address translate: a router does it)


To: 173.194.79.16 port 25
From: 192.168.1.2 port 12345
Let's Talk!
To: 173.194.79.16 port 25
From: 67.225.235.59 port 51211
Let's Talk!
Computer
Router
Server
To: 192.168.1.2 port 12345
From: 173.194.79.16 port 25
OK, Lets!
To: 67.225.235.59 port 51211
From: 173.194.79.16 port 25
OK, Lets!

Router uses a unique made-up response-port number each time for each internal computer (a specific internal IP address).

For each outgoing connection, the router remembers and keeps track of the port number for each local IP and response-port number that it was given as long as the connection is open.

From the diagram above, the router remembers:
I assigned prot 51211 for the connection that goes back to 192.168.1.2 port 12345.

Now, as long as that connection remains open, as long as that conversation continues, when anything comes to the router from the internet destined for port 51211, the router knows that it needs to send it along to port 12345 on the computer at 192.168.1.2.

As soon as the connection between the local computer and the remote server is closed and that conversation is over, the router can simply forget this information because there's no need to use it again.

The next time that a new connection is made, the router will simply assign a different "get back to me here" response-port.

Character Encoding:
ASCII — 2^8 (256) characters (English alphabet, roman numerals, math and other special characters).

UTF-8/16/32: (Unicode Transformation Format) including all symbols from other languages.

Theory of computation:

———————————————

[Automata]
[computability]
[complexity]

Real world problem can be formed in “Language”.

Regular language (solve by DFA, NFA) —> context-free language (solve by PDA or CFG) —> recognizable —> decidable (solvable)

Turing machine VS computing:

A **Turing machine** is a mathematical model of computation that defines an abstract machine[1] that manipulates symbols on a strip of tape according to a table of rules.[2] Despite the model's simplicity, given any computer algorithm, a Turing machine capable of simulating that algorithm's logic can be constructed.[3]

Turing completeness **is** the ability for a system of instructions to simulate a **Turing machine**. A **programming** language that **is** **Turing** complete **is** theoretically capable of expressing all tasks accomplishable by computers; nearly all **programming** languages **are** **Turing** complete if the limitations of finite memory **are** ignored.

The Turing machine model is a *theoretical* model of what "computing" is all about. As a theoretical model it was designed so it is simple to manipulate and to prove things about it, specifically to explore what can or can't be computed. It also serves as a simple model in which to discuss (and prove things about) the time required to do a computation, or how much space (memory) is required. The emphasis is on *simplicity* (specially in using only basic mathematical concepts).

A programming language, in contrast, is designed to make it easy to write (and read!) by humans, often also such that the concepts of its application area are handled directly. For example, a language like Perl handles operations on strings, even complex stuff like searching for patters, directly. SQL is tailored to operating on relational databases, doing queries searching for data with certain constraints and manipulating the database. And so on.

The Church-Turing thesis asserts that anything that can be computed in any meaningful sense of the term can be computed by a Turing machine. This is the final outcome of a decade-long frenzy in coming up with models of computation, all of which turned out equivalent. So, in theory, yes, they are equivalent (as far as the definition of the language and its implementation, and the machine on which it runs are correct). But as the saying goes, in theory, theory and practice are the same thing; in practice, they are very different. To write down a Turing machine to do even simple tasks is a lot of painstaking work, and that might be just one simple line in your favorite programming language.

True, all (general purpose) programming languages are believed to be equivalent to the Turing machine. (According to the Church-Turing thesis, they will not compute more, and usually it is clear how to simulate a TM in your favorite language). That does not mean that programming a Turing machine is a practical thing to do. Far from it. To do real programming better languages are developed. In fact those languages evolve over time, when we learn what features make programming languages simpler to use, or less error prone.
Still, the Turing machine is around. It serves as a yard stick to define complexity, and computability.
(added.) As noted in the comments, not every programming language is designed to be of Turing power, some deal with specific tasks, like regular expressions. On the other hand, there are powerful exotic languages designed in such a way to make programming virtually impossible. For fun.

VirtualBox:
A program allows you to install an operating system without changing your computer's main operating system.

Operating System (OS):
An operating system is the primary core set of software that manages all the hardware and other software on a computer.

OS handle everything from keyboard and mice, storage devices, and display.

When an application wants to print something, it hands that task off to the operating system. The operating system sends the instructions to the printer, using the printer's drivers to send the correct signals. The application that's printing doesn't have to care about what printer you have or understand how it works. The OS handles the details.

The OS also handles multi-tasking, allocating hardware resources among multiple running programs. The operating system controls which processes run, and it allocates them between different CPUs if you have a computer with multiple CPUs or cores, letting multiple processes run in parallel.

When you run Minecraft, you run it on an OS. Minecraft doesn't have to know exactly how each different hardware component works. Minecraft uses a variety of OS functions, and the OS translates those into low-level hardware instructions. This saves the developers of Minecraft—and every other program that runs on an OS—a lot of trouble.

Desktop OS: Microsoft Windows, Apple macOS, Google's Chrome OS, and Linux

Smartphone OS: Apple's iOS and Google's Android

Smart TVs, game consoles, or Wi-Fi routers is a computing device that runs an OS: embedded OS: specialized operating systems with fewer functions, designed specifically for a single task like running a Wi-Fi router, providing GPS navigation, or operating an ATM.

The "kernel" is the core computer program. This single program is one of the first things loaded when your operating system starts up. It handles allocating memory, converting software functions to instructions for your computer's CPU, and dealing with input and output from hardware devices. The kernel is generally run in an isolated area to prevent it from being tampered with other software on the computer.

Why use linux:
Does that operating system you're currently using really work just fine?? Or, do you find yourself battling obstacles like viruses, malware, slow downs, crashes, costly repairs, and licensing fees?

UNIX:
Most operating systems can be grouped into two different families. Aside from Microsoft's Windows NT-based operating systems, nearly everything else traces its heritage back to Unix.
Linux, Mac OS X, Android, iOS, Chrome OS, Orbis OS used on the PlayStation 4, whatever firmware is running on your router — all of these operating systems are often called "Unix-like" operating systems.

Assembly language:

The code recognized by the processor is called instruction code. Each instruction code must be programmed byte by byte in the proper order for the application to run. Instead of forcing programmers to learn all of the instruction codes, developers have created high-level languages, which enable programmers to create programs in a shorthand method, which is then converted into the proper instruction codes by a compiler. This enables programmers to concentrate on the logic of the application program, rather than worry about the details of the underlying processor instruction codes.

The downside of using high-level languages is that the programmer is dependent on the compiler creator to convert programming logic to the instruction code run by the processor. There is no guarantee that the created instruction codes will be the most efficient method of programming the logic. For programmers who want maximum efficiency, or the capability to have greater control over how the pro- gram is handled by the processor, assembly language programming offers an alternative.

Assembly language programming enables the programmer to program with instruction codes, but by using simple mnemonic terms to refer to those instruction codes. This provides programmers with both the ease of a high-level language and the control offered by using instruction codes.

Assembly language development environment:

Unlike a high-level language environment in which you can purchase a complete development environment, you often have to piece together an assembly language development environment.

- An assembler
- A linker
- A debugger
- A compiler for the high-level language
- An object code disassembler
- A profiling tool for optimization

Assembler:

To create assembly language programs, obviously you need some tool to convert the assembly language source code to instruction code for the processor. This is where the assembler comes in.

Granddaddy of all assembler — Microsoft assembler(MASM).

NASM, GAS, HLA.

The GNU assembler program (called gas) is the most popular assembler for the UNIX environment.

Compiler:

most professional programmers attempt to write as much of the application as possible in a high-level language,   such as C or C++, and concentrate on optimizing the trouble spots using assembly language programming. To do this, you must have the proper compiler for your high-level language.

The GNU Compiler Collection (gcc) is the most popular development system for UNIX systems.

Use the GNU debugger program (gdb) to debug and troubleshoot C and C++ applications.

Profiler:

To determine how much processing time each function is taking, you must have a profiler in your toolkit.

Process and program:

Program sits in the server.

Process: if we want to process the program, processor fetch it from the memory, and set up a route to execute the program.

Tor:

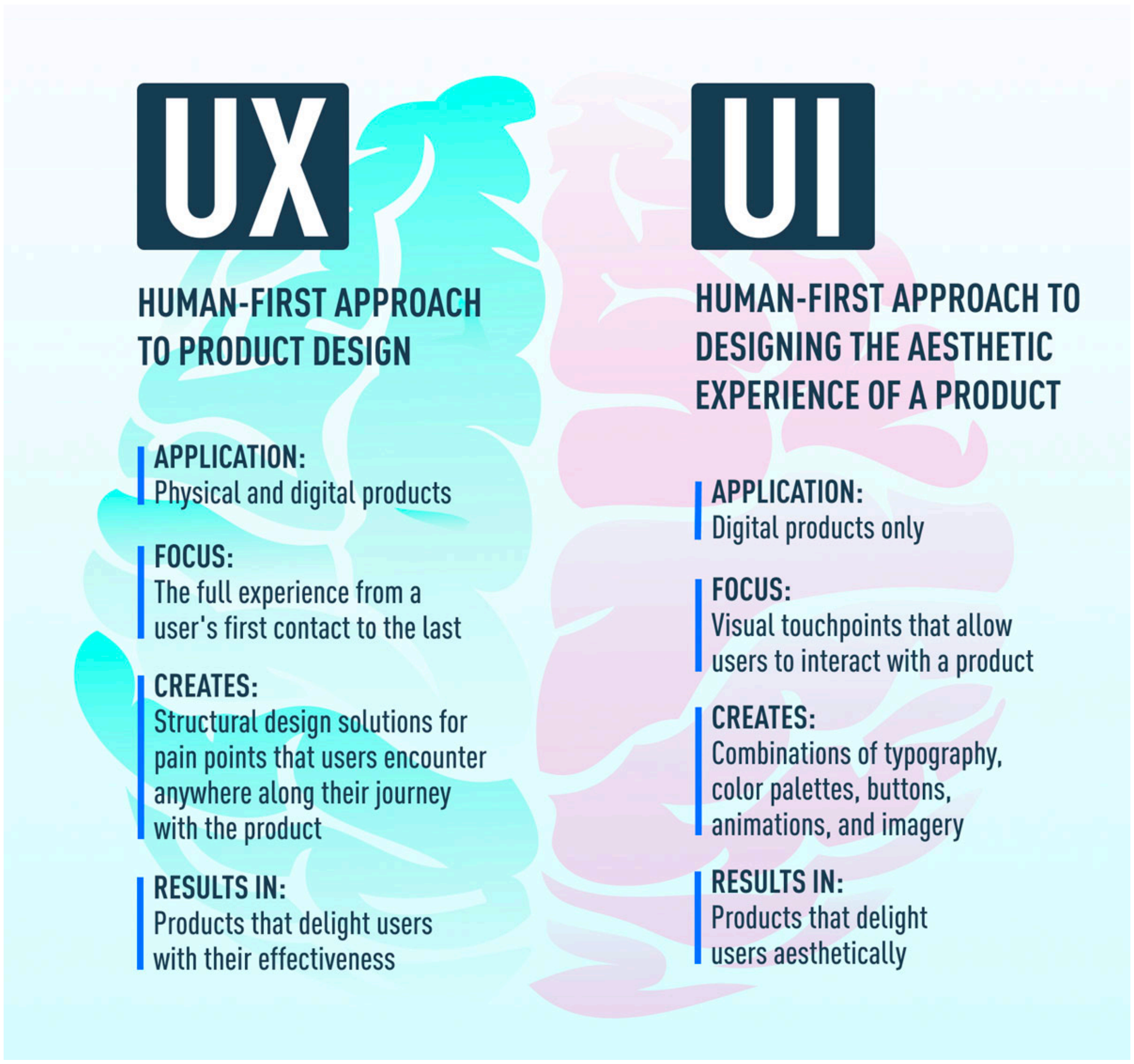
The Tor is designed to stop people - including government agencies and corporations - learning your location or tracking your browsing habits.

UX: User experience **UI**: User interface

Front-end — back-end — full-stack:
Website development is the process of building websites and applications. Unlike UX and UI Design, web development focuses more on coding and making sure a website functions well.

Front-end:
These developers take the visual designs from UX and UI designers and bring the website to life.

It focuses on coding and creation of elements and features of a website that will be seen by the user. It's about making sure the



UX

HUMAN-FIRST APPROACH TO PRODUCT DESIGN

APPLICATION:
Physical and digital products

FOCUS:
The full experience from a user's first contact to the last

CREATES:
Structural design solutions for pain points that users encounter anywhere along their journey with the product

RESULTS IN:
Products that delight users with their effectiveness

UI

HUMAN-FIRST APPROACH TO DESIGNING THE AESTHETIC EXPERIENCE OF A PRODUCT

APPLICATION:
Digital products only

FOCUS:
Visual touchpoints that allow users to interact with a product

CREATES:
Combinations of typography, color palettes, buttons, animations, and imagery

RESULTS IN:
Products that delight users aesthetically

visual aspects of a website are functional; the site is easy to interact with while also running smoothly.

Elements:
Buttons; Layouts; Navigation; Images; Graphics; Animations; Content organization

Languages:
HTML (Hyper Text Markup Language): for creating webpages;

CSS(Cascading Style Sheets): While HTML is used to create structure on a site, CSS is used to bring style and flair. It defines a site's colors, fonts, and other site content;
JavaScript: make a site interactive and fun
Frameworks and libraries: AngularJS; React.js; JQuery; Sass

Back-end:
As an example, when running a social media website, we need an accessible place to store all of user's information. This storage center is called a database. Databases are run from a server, which is essentially a remote computer. A back end developer will help manage this database and the site contents stored on it. This ensures that front end elements on your social media website can continue to function properly as users browse uploaded content and other user profiles.

Tasks:
Troubleshooting and debugging web applications; Database management; Framework utilization.

Languages:
PHP; C++; Java; Ruby; Python; JavaScript; Node.js
frameworks: Express; Django; Rails; Laravel; Spring