Operating Systems and Computer Networks

SCOMRED, December 2020

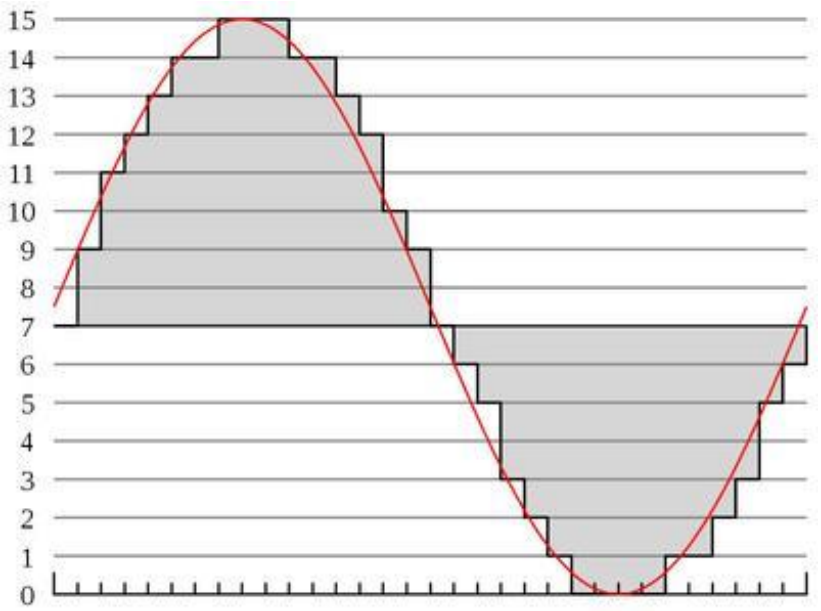
Digital data

Data in the real world is mostly analogue, in simple words this means that when measuring something it may have an infinite number of different values. Take for instance time, for smaller a time interval is, you can always split it into two halves.

Electronic devices can handle analogue data directly by representing it through an equivalent analogue electric voltage level, we still have analogue TV and radio.

The main issue with an analogue electric voltage level is electronic circuits are rather imperfect and the voltage level gets distorted.

The workaround to solve this is representing analogue data in a digital form by unique steps of electric voltage levels.



Binary digital data

Current electronic devices use well defined voltage levels to represent data, in fact they only use two voltage levels, so this is called **binary data**.

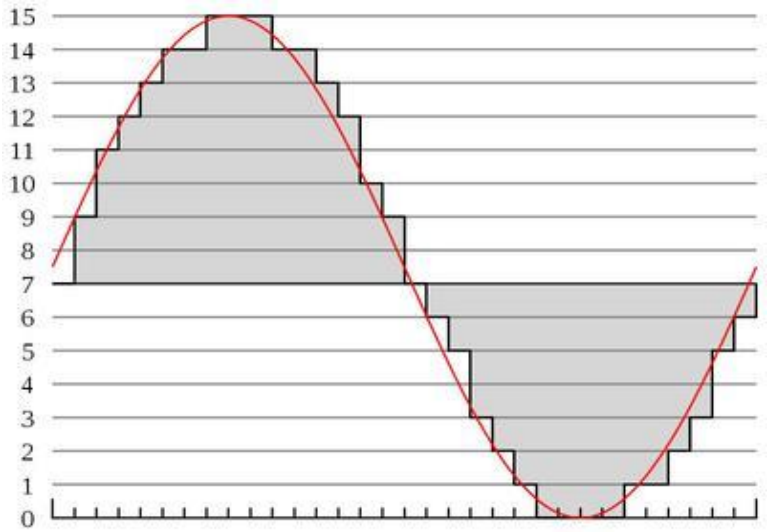
Having only two voltage levels means at each instant only one of two symbols can be represented, either 1 or 0. This is called a **bit**.

By using a low or high voltage level in a circuit only one bit can be represented. This is true for one electric circuit, but with several electric circuits in parallel (**bus**) several bits can be represented at once. By putting together bits of several electric circuits they can be taken as a binary representation on a bigger number, how big that number can be depends on how many circuits (bits) are used.

The image on the right shows analogue data being represented by 16 different digital values from 0 to 15. To represent it in a binary form, 4 bits (circuits) would be enough because $2^4 = 16$.

0 value is represented by bits 0000

15 value is represented by bits 1111



CPU (Central Processing Unit)

A CPU is an electronic chip capable of performing very simple operations over binary data represented by voltage levels in internal electronic circuits called **registers**. The number of circuits (bits) it uses to represent a number depends on the CPU type, older ones use 8 bits numbers, most recent ones operate with 64 bits numbers.

The operations a CPU is capable of performing are very simple, for instance incrementing or decrementing a number, adding or subtracting two numbers and bit by bit operations. There are two key points why CPUs are useful:

1. They are very fast, they execute millions or billions of operations in a single second.
2. They are programmable, a CPU is given a long sequence of instructions, each is a number itself, but for the CPU it represents an operation to perform.

Even though a CPU is capable of only simple operations, those simple operations can be used to implement more complex operations, for instance the multiplication can be implemented by addition operations and the division by subtraction operations. By implementing more complex operations, they can then be used to implement even more complex ones ending up with what a computer is capable of.

Central memory (RAM and ROM)

Internally a CPU can store only little more than a handful of numbers over which it can perform operations, they are called registers. Attached to the CPU (usually in independent chips) there's the central memory, the central memory stores bits of information, usually organized in bytes (a set of eight bits), each storage place in the memory is identified by a memory address (a number itself).

The main interaction between the CPU and the central memory is data transfers from a register to some memory address and data transfers from a memory address to a register, to achieve that two buses (set of parallel circuits) are usually required:

The address bus: used by the CPU to inform the memory about which memory address it wants to use.

The data bus: used to transfer data between the CPU and the memory.

We can distinguish two types of memory:

RAM (Random Access Memory) - data in this type of memory is lost when power is turned off (it's volatile), but when power is on it can then be used to store data (it's writeable).

ROM (Read Only Memory) – holds unchangeable persistent data. It's normally used to instruct the CPU on how to boot the device.

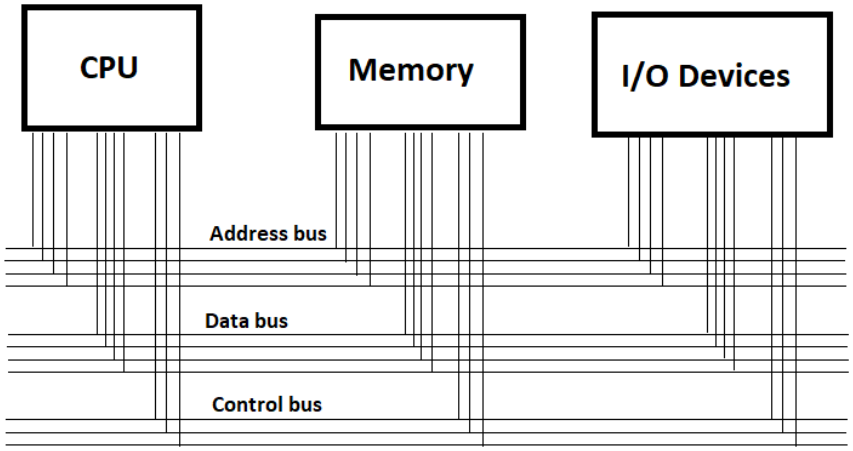
CPU program execution and I/O devices

A CPU executes a sequence of instructions it supports (program), each operation is represented by a number. The program (a sequence of numbers) is stored in memory, one important register every CPU has is the Program Counter (PC).

The PC register contains the memory address of the instruction being executed by the CPU, once the instruction is executed the PC is incremented so the instruction in the memory's next position will be executed next. However, the PC value may be changed in other ways to make the program execution jump to a different memory position.

Many CPUs, when powered on start with PC=0, so the program that will be executed is the one placed on memory address starting from zero. Usually this will be a ROM, because at that stage RAM contains nothing.

The CPU and the memory make a closed system, to interact with the external world, I/O (Input/Output) devices are required. An I/O device is usually controlled by the main CPU but in some cases it may access memory directly and it may even have its own CPU.



I/O devices and controllers

What are usually called I/O devices like for instance video displays and keyboards are not directly attached to the system buses, special chips called controllers have that mission, so we have keyboard controllers, video controllers and so on.

One relevant type of I/O device are external storage devices, often called disks (HDD and SSD), they are attached to disk controllers that provide appropriate cable interfaces like for instance the SATA (Serial AT Attachment) bus.

Two significant characteristics of disks are:

1. Their high storage capacity, usually several hundreds of GB (Gigabytes - 10^9 bytes) or several TB (Terabytes - 10^{12} bytes), when compared with central memory that is usually some gigabytes only. However, regarding speed, they are much slower than RAM, specially for a traditional HDD, not so much for an SSD (Solid State Drive).
2. Unlike with RAM, stored data is non-volatile, it persists even if power is off.

Nowadays, many I/O devices use a USB (Universal Serial Bus) cable connection, therefore they are accessed through USB controllers.

Programming

A program is a sequence of instructions to be executed one after the other, though we have already mentioned jumps from one execution point to another can be made.

Ultimately it's the CPU that executes instructions, but they are too simple to be useful for a standard software developer, CPU level instructions are called machine code. So special software is used to receive higher level programming languages and automatically output machine code for the CPU to run, this software is called a compiler.

The compiler mission is transforming a text with a high level programming language called source code into machine code, also called binary code. Then the binary code can then be directly run by the CPU. So we must use the appropriate compiler, it must understand one specific high level programming language, and it must produce binary code for a specific CPU.

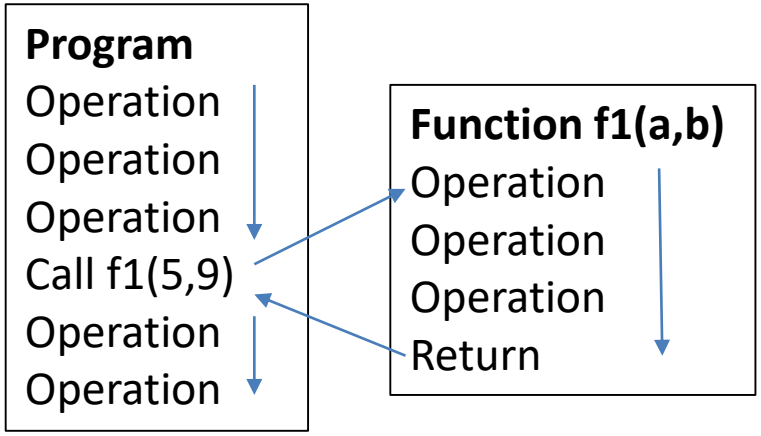
One other way to run high level language programs is through an interpreter, instead of transforming the source code into binary code like a compiler, an interpreter executes the high level language program line by line, one at a time. Often interpreted programs are also called scripts. Notice that unlike with the compiler, the interpreter program must be running to execute the program.

Functions

Many high level operations executed by a compiled program (binary code) are not implemented by the program itself, if that was so the binary code of each program would be huge.

Take for instance the multiplication case, this operation isn´t supported by many CPUs but it's widely used by many programs, it would be pointless for every program to implement it. The solution is providing a shared implementation to be used by every program.

A function is a piece of binary code, placed somewhere in memory, implementing an operation, when needed, programs can call the function by jumping execution to that place in memory, when the function execution ends, the execution returns back to the calling program. The calling program may pass data to the function in the form of arguments, the function may also return a result to the calling program.



System Functions and abstraction

Functions avoid the need of every program to implement the code for every operation they use, thus programs' binary code becomes significantly shorter. There's one additional key benefit on using functions, they can be used to provide abstraction.

Take for instance an operation of printing a string on the device's console. This operation may be very dependent on the hardware being used, however if together with the hardware a software function **printString(STRING)** was provided, then a program could just call that function, whatever hardware is was running on.

This is because the way to call the function (the function's interface) is hardware independent. For each different hardware an appropriate **printString(STRING)** implementation should exist, but the interface is always the same.

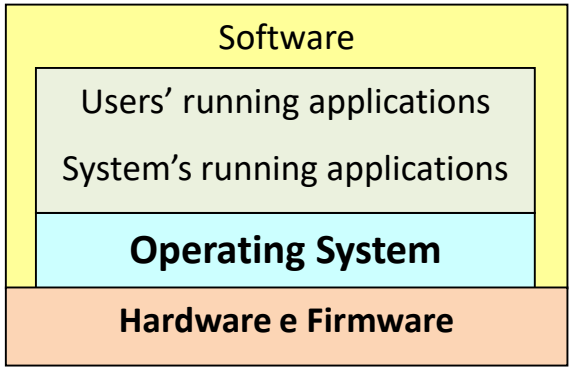
These are usually called system functions or system-calls, some of them are provided together with the hardware, often in a ROM. In a personal computer many on them are stored in the BIOS (Basic Input/Output System).

The Operating System (OS)

The abstraction level and features provided by low-level system functions related to the existing hardware is not enough to attain a stable platform to run user programs, over them, a more sophisticated software called the **Operating System** is running, usually loaded in RAM.

The operating system uses the hardware directly or through low-level system functions and provides a whole new higher level and more abstract set of system function for programs running over it.

With modern operating systems, applications are no longer allowed to access the hardware or low-level system functions directly, they must call the operating system's functions.



Because applications don't interact with the hardware, they become hardware independent, nevertheless they are OS dependent.

However, an application developed for a given operating system will always run on that operating system whatever the underling hardware is.

Multiuser and multitasking operating systems

Current operating systems are multiuser and multitask, this means they are able to run applications belonging to different users at the same time. Each user is identified by a unique short name, usually called the username or login name, to check a user's authenticity when he presents itself to the system a secret password is typically required.

Each user has an user account with several attributes regarding the user, among them each user has some user rights, meaning what he is able to do.

Because applications only interact with operating system, when a user's application calls a system function it's easy to check if that user is entitled to make that specific call (has the required user right to do so).

Also, because there are several applications running at the same time, concurrency issues when they access the same system resource can be properly handled by the operating system. Again, this is true because all accesses are made through the operating system.

Operating system – processes and memory

A process is a running program, the binary code is loaded from a file into memory and run by the operating system.

Beyond the running code itself a process also has data (variables) both are together in a closed box, meaning the process's code can only interact with its own data. Direct interactions between different processes are no allowed.

The operating system manages running processes in such a way they appear to be running all at the same time, that might be true if there are several CPUs available, but usually there are more processes than the number of CPUs on the system.

Even with a single CPU the operating system is able to keep several process running at the same time, it does so by executing code from one process during some time and then skipping to another process. **Ultimately all code of every process is executed but not all at once.**

Within a process it's also possible to have different parts of the code running at the same time, they are called threads. All threads that run inside the same process share the process's data (variables). When using multiple threads the programmer should deploy locking mechanisms to avoid the concurrent access to the same variable by several threads at the same time, otherwise result are unpredictable.

File-systems

The operating system also manages external devices, that encompasses everything beyond the CPUs and central memory. For each device an appropriate device driver is required. The device driver is a piece of software added to the operating system with a set of functions appropriate to interact with the device's controller.

Among several other types of external devices, one most important are external disks. External disks are persistent mass data storage devices, but they are not suited to be directly used by applications. If applications were allowed to read and write wherever they wanted within a disk it would be chaotic.

To solve this issue disks are organized by the operating system. To start with, each disk is divided in different parts called partitions. Then, in each partition, a file-system is created.

The file-system is an organization of the storage are in files and folders and that is very suitable for both the running applications and for the operating system to enforce access control.

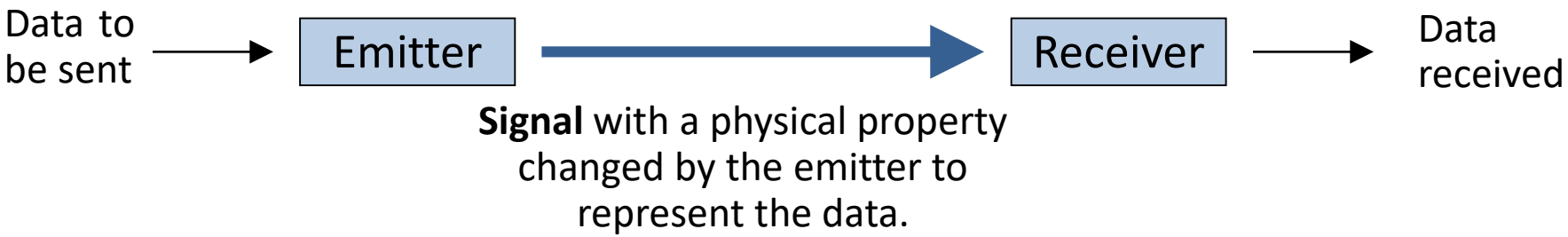
Applications use disks indirectly by using file oriented functions like open(), read(), write() and close() made available by the operating system.

All information regarding the existing partitions and the file-system contents is stored in the disk itself.

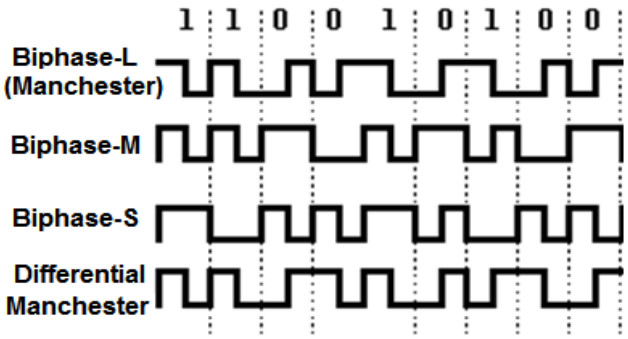
Computer Networks - signals

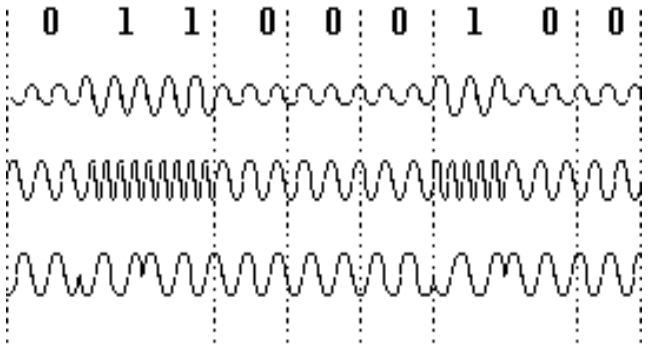
A computer without a network connection to the internet is useless, if asked, that would be the view of most users. Computer networks' objective is allowing data transfer between apart computers.

The base stone of computer networks is data transmission through signals. A signal is a physical phenomena capable of propagating itself, for instance electric current, light or radio waves.



By changing signal properties like intensity, frequency or phase, data to be sent is represented, and thus transmitted along with the signal. Because they carry data, signals are also known as carriers.





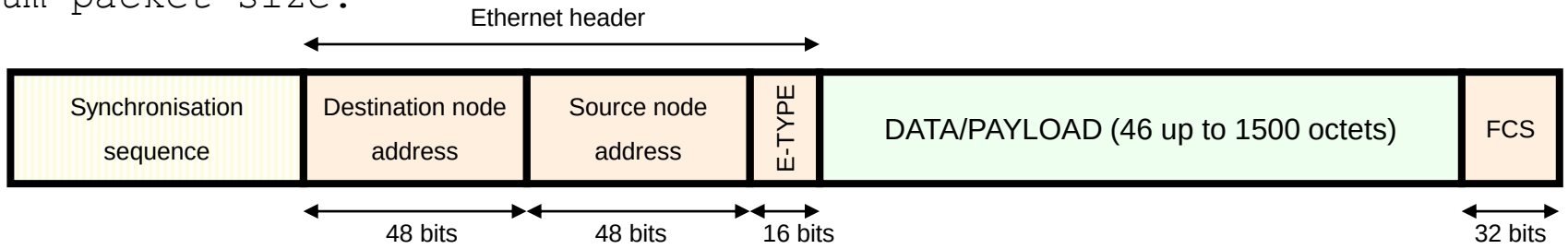
Networks nodes, packets and node addresses

A computer connected to a network is both an emitter and a receiver, it's called a network node. Depending on the network it may be even able to emit and receive at the same time (full-duplex).

Networks are shared by many nodes, so each must have a unique address called node address. When a node sends data, attached to data there must be the **destination node address**, that will be used by the network to know where the data is supposed to be delivered.

In addition to the destination node address, the **source node address** must be also added, that´s the emitter own node address. The source node address may be used by the receiver to reply back to the emitter, it may be also used by the network to inform the emitter of an error during the delivery.

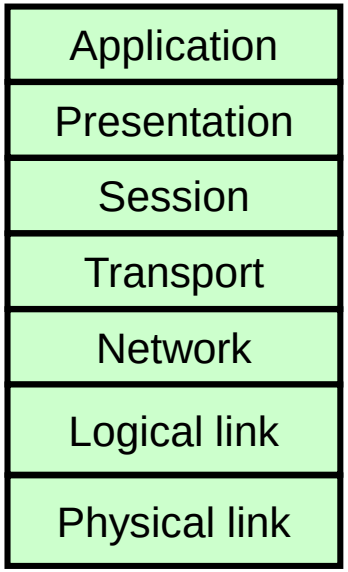
Also, for the sake of equity when using the network, each node can only send a limited number of bytes each time. Each chunk of data sent by a node is called a packet and there's always a limit on the maximum packet size.



The image above represents an Ethernet packet, also called frame.

Network layers – the OSI model

Several components of hardware and software are required by a node to able to make the network available to applications at the operating system level. These components are organized in layers as defined by the OSI (Open Systems Interconnection) model, which establishes 7 layers. The general idea is, **a layer uses services of the layer below it, improves them is some way, and provides to the layer above it those improved services.**



Applications are placed at the top layer (**layer 7**), it's also known as the **Application Layer.**

Layer 1 is the base, also called **physical layer**, it describes cables, cable connectors and the way data is transformed into signals.

Layer 2 establishes packets, also establishes node addresses and error detection. Packets at layer 2 are often called frames, the image on the previous page was about a layer 2 Ethernet frame.

Ethernet networks use 48 bits node addresses, they are called Ethernet addresses, physical addresses or MAC addresses. For human representation, hexadecimal symbols are used (one for each set of 4 bits) and each set of 8 bits is separated by a colon or a dash, for instance : **00:60:B0:3C:93:DB** or **00-60-B0-3C-93-DB** .

The network concept – Layer two network

Strictly speaking, the network concept refers to a layer two network, meaning a set of computers interconnected by the same layer two technology, for instance, an Ethernet network. In such a way those computers are able to communicate with each other just by using that layer two technology and nothing else.

This network concept coincides with the **broadcast domain** concept. Within a network data may be sent to a special destination address called the **broadcast address**, when that happens each and every node on the network receives a copy of that data. In an Ethernet network the broadcast address is **FF:FF:FF:FF:FF:FF**.

Nevertheless, data sent to the broadcast address never travels beyond the layer two network limits, this area, up to where broadcast traffic travels, is called the **broadcast domain**.

Bigger networking infrastructures like the internet are made of several interconnected layer two networks, but each of them is independent, often they will use different layer two technologies, thus frames will have different formats and can't be transferred from one layer two network to another.

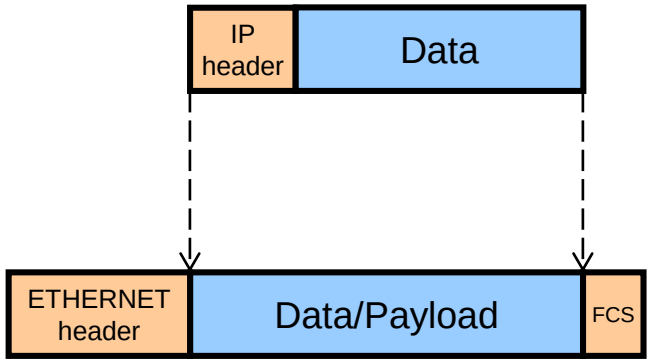
The layer three (network layer) exists to solve this issue, it establishes an universal packet format and an universal node addresses format to be used in all networks.

Internetworking – Layer three

To allow data to pass-through across several different layer two networks, layer three establishes new packet formats, in layer three they are not called frames anymore, they may be called datagrams or simply packets. At present, two layer three packet formats are used in the internet, they are defined by IPv4 and IPv6 (Internet Protocol version 4 and Internet Protocol version 6).

The way these layer three protocols operate is by using any existing layer two technology to transport their IP packets as payload, this is called encapsulation.

The image on the right represents an IP packet being encapsulated inside a layer two Ethernet frame.

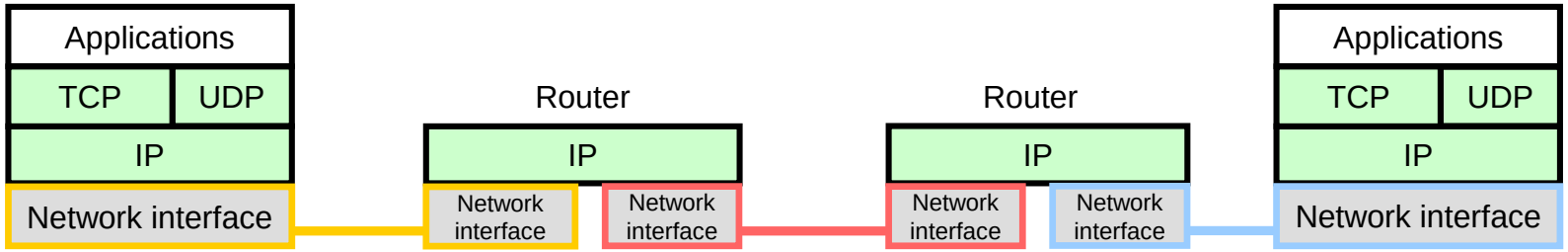


Because the IP packet has the same format whatever the layer two technology is, it may be now transferred from on layer two network to another, even if they are completely different.

Intermediate nodes called **routers** take that mission. An intermediate node is a node that forwards data, it receives packets and retransmits them, in opposition to end nodes that are the origin or final destination of packets.

Internetworking – Routers

Because a router is an intermediate node that operates at layer three, it may be connected to different types of layer two networks. To transfer an IP packet from one network to another it receives a frame from one layer two network, picks its payload (the IP packet) and then encapsulates the received IP packet in a frame of the other layer two network.



The image above presents this idea, the network interface is a specific layer two connection, each different colour represents a different layer two technology.

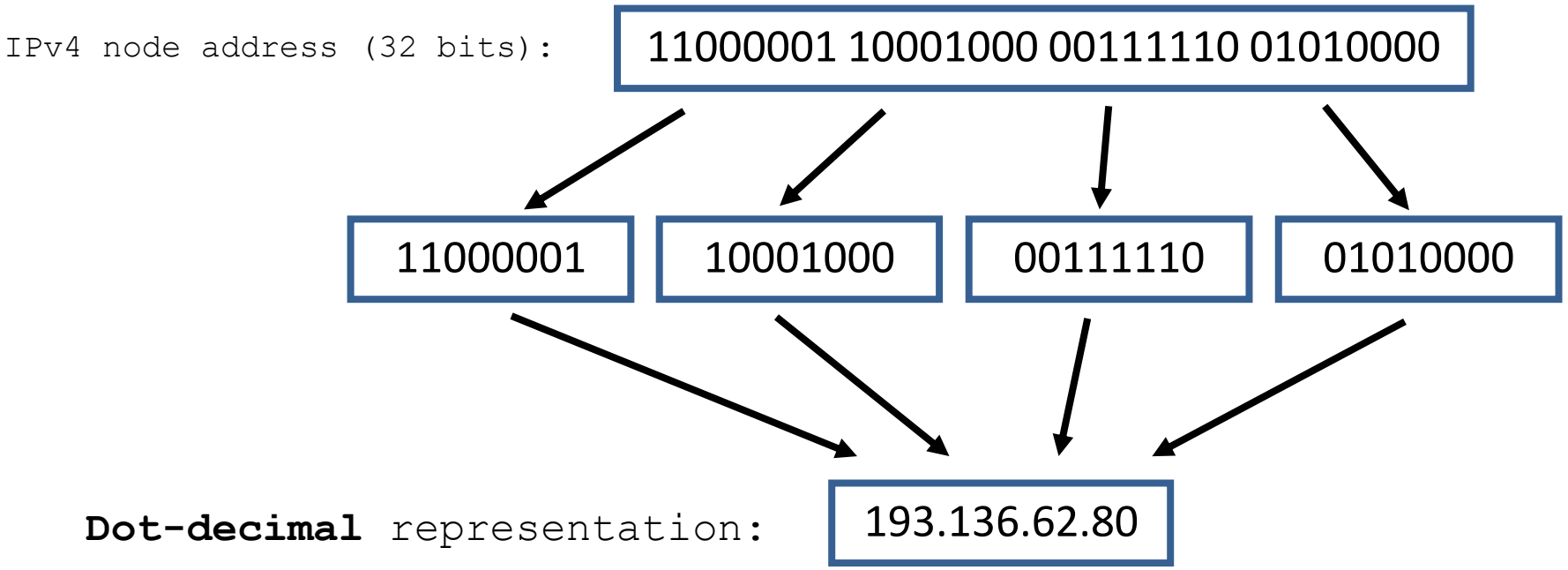
This is how large networks like the internet operate, it's called internetworking, meaning interconnecting different types of networks to build a larger network.

Because each layer two technology uses different node address formats, to work across diverse layer two technologies, IP protocols must also establish a new and independent format for the node address, it's called the IP address.

IP node addresses

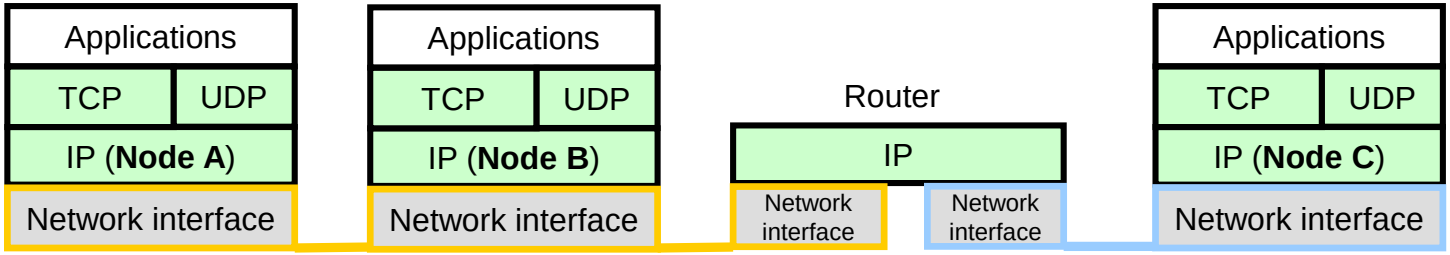
Although they are used in a similar way, IPv4 and IPv6 use node addresses with different sizes. IPv4 uses 32 bits addresses and IPv6 uses 128 bits addresses.

Starting with IPv4 addresses, they have 32 bits, for human representation they are split into 4 sets of eight bits (octets), each represented as a decimal number going from 0 up to 255. Octets are separated by a dot, this is called the **dot-decimal** representation. For instance:



Network addresses

Because layer three protocols transfer packets between networks, they must define **network addresses**, this is required because when sending a packet it must be established if the destination address belongs to the network the sender is on, if so no router is required and the packet should be sent directly using the layer two network, otherwise the packet must be sent to a router to be transferred to a different network.



On the image above, if Node A (leftmost) wants to send to Node B no router is required (both are on the same network), however if it wants to send to Node C it must use the router because Node C is on a different network.

Both for IPv4 and IPv6, the network address is included in the node address itself, a number of most significant (leftmost) bits of the node address are the network address. The exact number of bits being used to identify the network, and consequently the number of bits that are left to identify nodes within the network is the **network prefix length**.

IPv4 network address

To fully identify an IP network address, both the network address and the **prefix length** must be specified together. The network address itself is also called the **network prefix**, it's the IP address used by nodes on the network, but all bits beyond the prefix length are zero.

A network address becomes fully defined by being specified as:

network prefix/prefix length

Examples of IPv4 networks: **193.136.62.0/24** and **10.30.0.0/16**

The following examples are wrong. They are not correct definitions for IPv4 networks: **190.200.50.0/16** and **30.200.0.0/8**

Why?

For IPv4 only, in some systems the network prefix length must be specified as a network mask. It's a dot-decimal representation with all bits within the prefix length having value 1 and remaining bits having value zero. Some examples are presented in the table.

Prefix Length	Network Mask
/8	255.0.0.0
/16	255.255.0.0
/24	255.255.255.0
/26	255.255.255.192

Maximum number of nodes on a network

Along an internetworking infrastructure like the internet, every network prefix (network address) being used must be unique and every node address must also be unique. This is because they both must be addressable.

All nodes within a network must use that network's prefix, thus the maximum number of nodes a network can have depends on the network´s prefix length.

Take for instance the IPv4 network **172.18.52.0/24**, all nodes belonging to it must have the same prefix, so node **172.18.52.10** belongs to it, node **172.18.53.10** does not. Addresses within this network go from **172.18.52.0** up to **172.18.52.255**, although that's true, in IPv4 the first and the last addresses of each network are reserved, so actually, **valid node addresses** on this network go from **172.18.52.1** up to **172.18.52.254** , a total of 254.

In general: $Number\ of\ valid\ node\ addresses = 2^{(32-Network's\ prefix\ length)} - 2$

Examples:

Network's prefix length	Number of valid node addresses
/16	65534
/24	254
/25	126
/30	2

Special IPv4 addresses

Worldwide it´s IANA (Internet Assigned Numbers Authority) that assigns IP network addresses and ensures they are unique.

IANA has reserved some IPv4 addresses for special uses:

Prefix	Use
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	These network prefixes are reserved for local private use, often called private networks. They are not recognized on the internet, so networks and nodes using them won't be able to communicate through the internet. Having said that, nodes and networks using private addresses may be able to use the internet if they use a local router performing NAT (*Network Address Translation*), such a router is able to hide all private addresses behind a single public (non private) address.
127.0.0.0/8	These are called *loopback* addresses, they are valid inside a node only. If a computer has no network interface connection, it will still have a loopback interface with address 127.0.0.1/8, it can be used to contact network services running in the node itself.
224.0.0.0/4	Multicast, when a packet is sent to one of these addresses a copy is received by a set of nodes called the a multicast group. The broadcast address may be viewed as a special case of multicast.
255.255.255.255	Generic broadcast address. Even though each IPv4 network has its own broadcast address (the last network address), if the network prefix is not known a node can yet send data to every node on the network by using this destination address.

Configuration of an IPv4 node

When a computer has a network interface connection to an IPv4 network, the interface must be properly configured with at least two attributes: the unique **node address** and the **network prefix length.**

With these two attributes, the node is able to determine the **network prefix** of the local network.

Knowing the local network, is key. When sending a packet to some destination address, a node must decide if it can do all by itself or it must get aid from a router. Aid from a router is required whenever the destination address belongs to a **remote network** and not the local network.

By comparing the destination address of a packet to be sent with the local network's address, a node may easily conclude if the destination address is not local. Then knows the packet must be sent to a router, this decision taking is called **routing.**

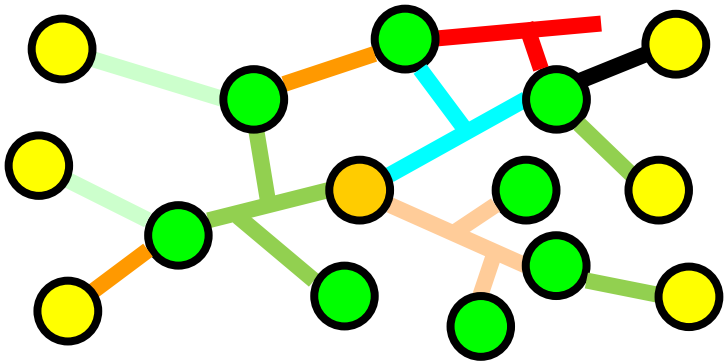
Usually end nodes only know on router around them, it's called the **default router** or **default gateway**, in this scenario every packet not intended to local networks is sent to that router.

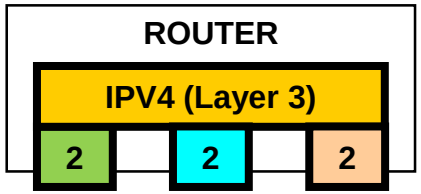
These three attributes (node address, network prefix length and default gateway) allow a node to communicate with whichever another node on the internet.

Configuration of an IPv4 router

Since a router's mission is transferring packets between different networks, it should be connected to more than one local network.

Also, unlike an end-node, usually a router is aware of several neighbour routers around it, not just the default gateway.





On the above image, the central router (in orange) has eight neighbour routers (in green), yellow routers are remote (not neighbours under the orange router's point of view).

Behind each neighbour router there're some networks that can be reached be sending to that neighbour router.

To specify this situation the default gateway is not enough, a full routing table is required with a list of networks, and for each, the neighbour router to be used, called the next-hop.

The IPv4 routing table

The routing table is a table of routes, meaning a tables of destinations (network addresses) and for each destination the way to get there.

For routers the "way to get there" is the IP address of the neighbour router (next-hop) that places the packet one step nearer the destination. Each router has its own routing table. Example:

This routing table shows there're three neighbour routers being used around this router.

The first route is for packets destined to network 190.16.10.0/24, those packet are sent to 172.14.5.8.

DESTINATION	NEXT-HOP
190.16.10.0/24	172.14.5.8
172.14.0.0/16	192.168.100.200
191.16.12.0/24	172.14.5.1
0.0.0.0/0	192.168.100.200

The second route is for packets with destination addresses belonging to network 172.14.0.0/16, they are sent the to neighbour router 192.168.100.200.

The last route is called the **default route**, it has a special network address (0.0.0.0/0) that matches any destination address. Because the routing table is sequentially searched until a match is found, this route should always be on the table's bottom.

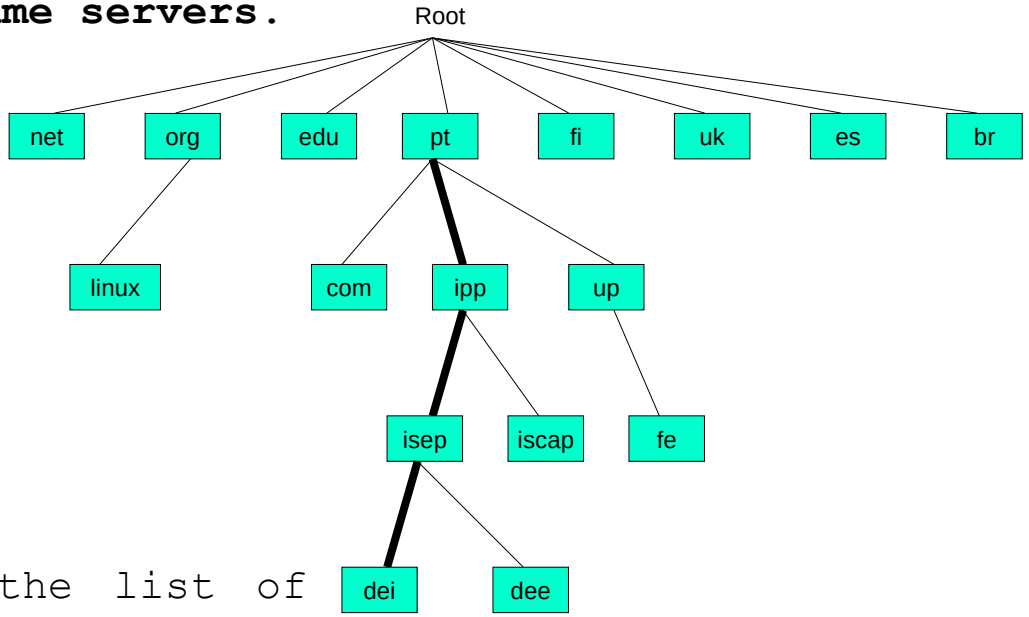
The next-hop for the default route is, of course, the default gateway.

DNS (Domain Name System)

Handling IP addresses it's not acceptable for most users, DNS allows the use of names instead. DNS establishes a tree of domain names based on an infrastructure of **name servers.**

A node can send a query to a name server to attain an IP address from any valid fully qualified domain name (FQDN).

For users that provide unqualified names, a default domain and additional search domains can be established on the node's configuration.



Therefore, we can now update the list of attributes an IP node is required to know:

- For each local network: the unique **node address** and **prefix length.**
- The **default gateway.**
- The **IP address of a DNS name server** and the **default DNS domain.**

This configuration data may be either statically settled by the node's administrator or received from a DHCP (Dynamic Host Configuration Protocol) server on the local network.