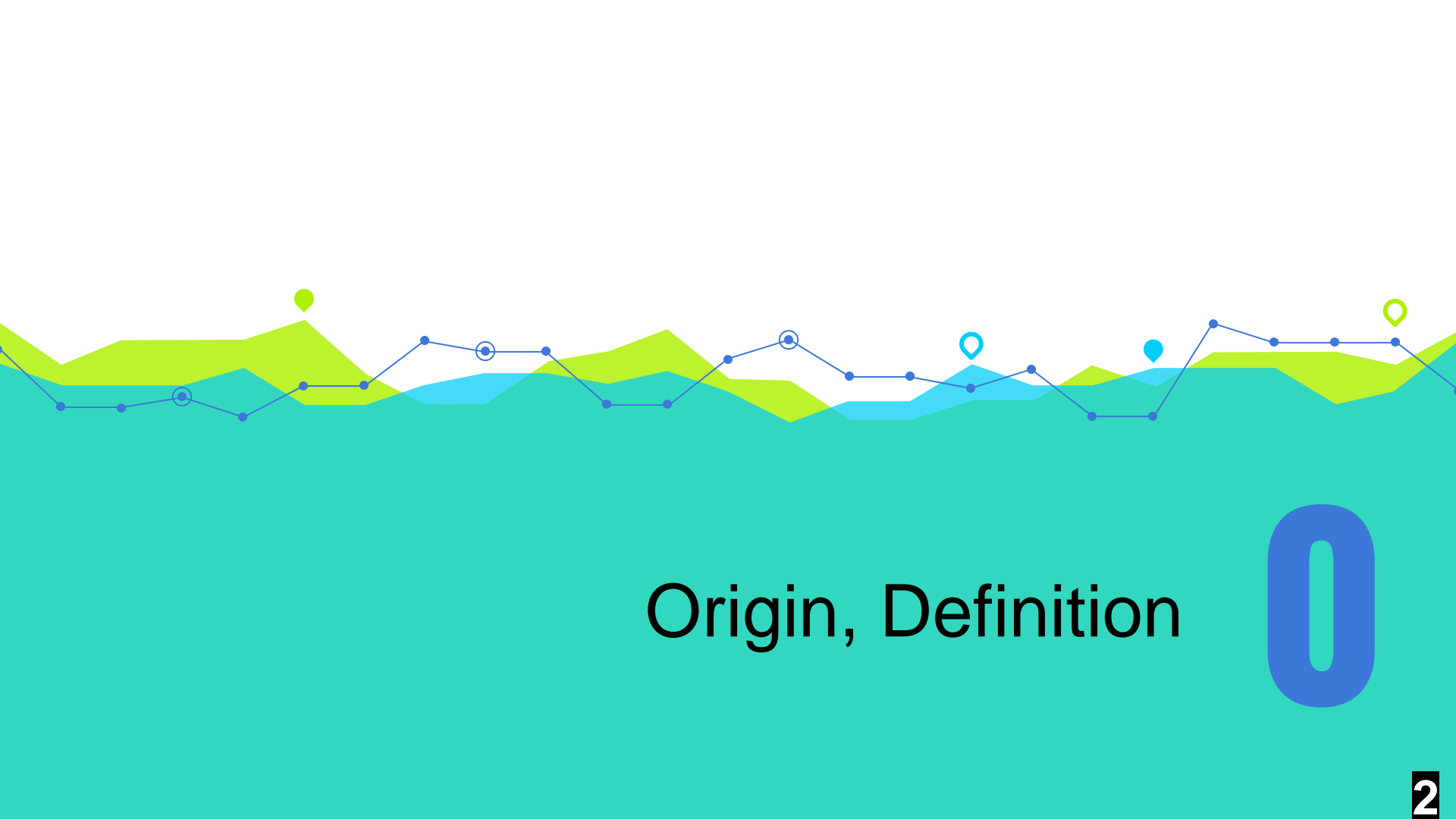


IoT Communication Network



Origin, Definition

0

Why Networks?

Networks were born from the need to transport data from one computer to another.

This data being put in the form of files, the basic application of networks was called file transfer.



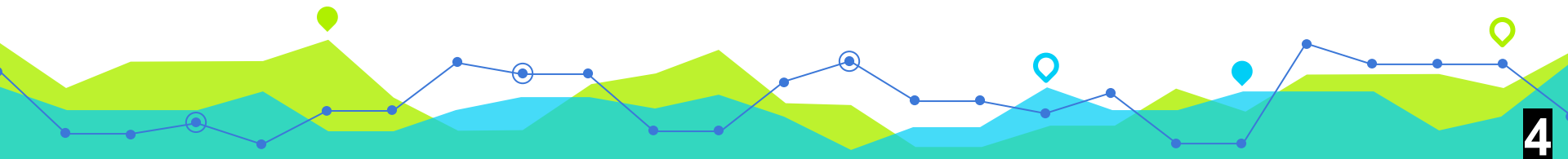
Why Networks?

A little later, the "transactional" appeared to allow a user to perform transactions with a remote computer

Example:

Reserving a plane seat.

We call it a session the set of transactions of a single user to perform a given task.

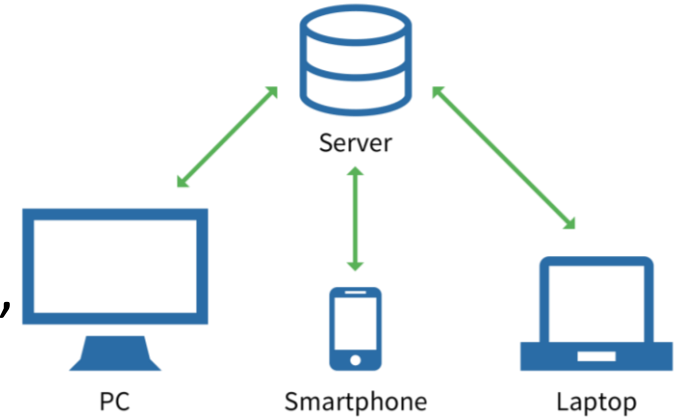


Why Networks?

With the development of the Web, the transactional service has diversified to allow the search for information through links.

These applications are called client-server, i.e., a client addresses itself to a server to obtain information.

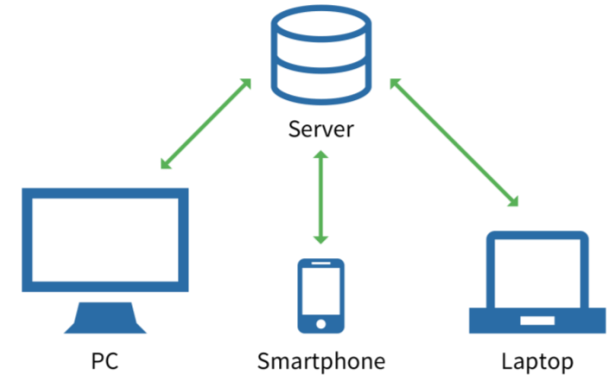
There are usually many more clients than servers.



Client/Server

In an information system, computers usually work in client-server mode through a network.

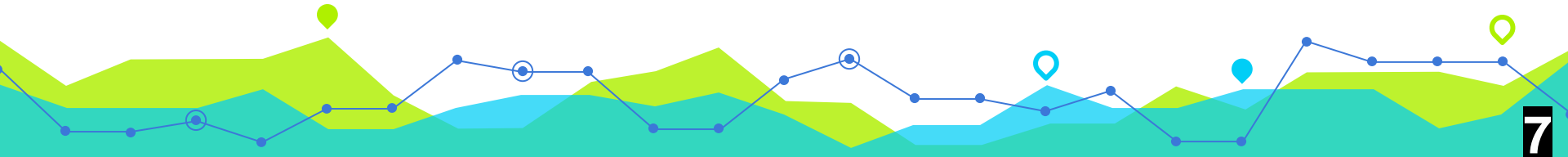
This simply means that a "server" computer receives requests from other "client" computers and in return the server sends back to the clients what they have requested. Just like when you ask a waiter at the local coffee shop for a lemonade.



Client/Server

The server:

A server is a computer generally of great power. It has no man-machine interface, except for its own administration and for loading information to be distributed to clients.



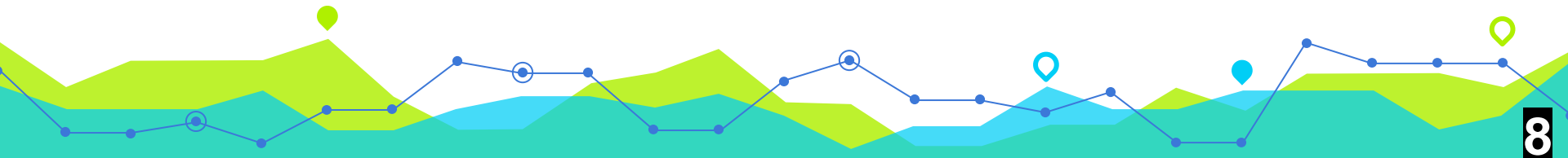
Client/Server

The server:

We differ:

- The file server that centralizes one or more sets of information, either as files or as databases,
- The print server which manages access to the queues of jobs submitted to the printers,
- The Web server which manages access to Intra/extranets and the Internet,

And so on...



Client/Server

The client:

A client is very rarely a high-powered computer. It is most often a microcomputer or even a tablet.

The client processes applications. To do so, it sends requests to the servers to obtain the necessary data, performs the processing and then stores and/or presents the result.



Client/Server

The client:

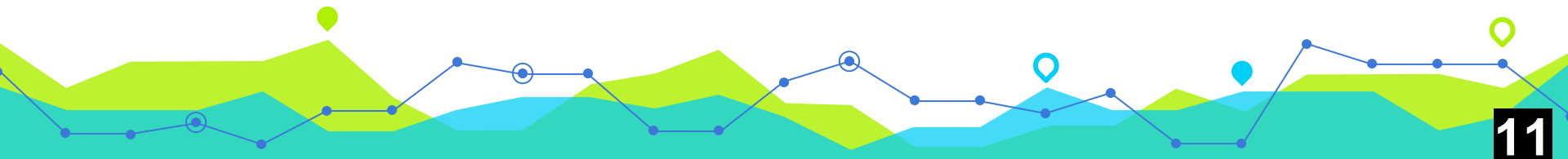
It has a human machine interface through which the user will enter the requests sent to the server and present the results.

These are typically the keyboard, the screen, the mouse and the printer. It has of course a disk, but it can receive external mass storage devices such as CD-ROMs, USB keys, etc. for data transport.

Notion of protocol

To communicate consists in transmitting information, but as long as the participants have not attributed a meaning to it, it is only data and not information. The participants must therefore not only speak a common language but also master the minimum rules for sending and receiving data. It is the role of a protocol to ensure all this.

A protocol is a set of rules that define how communication occurs in a network.

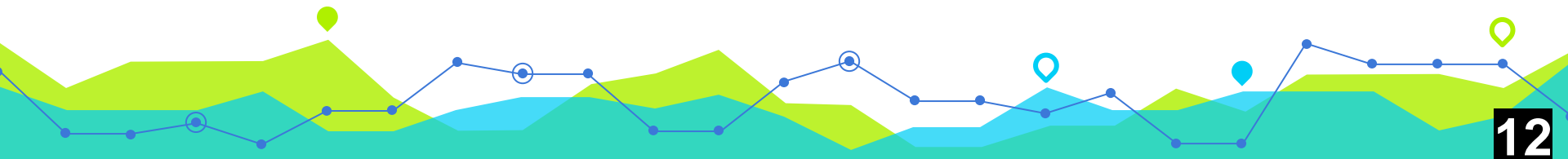


A protocol, multiple protocols

A protocol that does "everything" would be difficult for humans to manipulate.

A good practice is therefore to dedicate protocols for a precise task. We will therefore encapsulate the data with each protocol. And each protocol will know how to manage its own encapsulation and not that of the others.

Each level n is based on the result of level $n-1$: we will talk about layers.

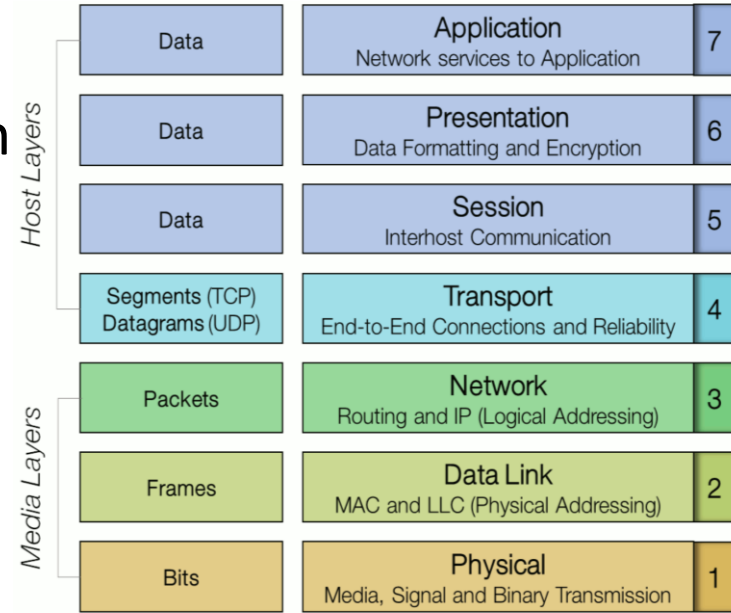


Two layered models

1/ The Open System Interconnect" model

The OSI model is a network communication standard for all computer systems.

It is a model of communications between computers proposed by the ISO which describes the functions necessary for communication and the organization of these functions according to layers.



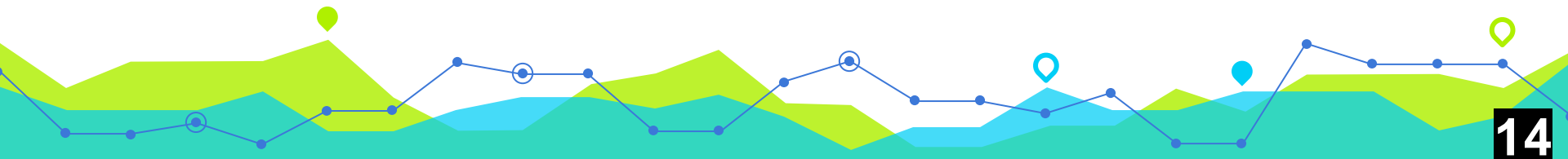
Two layered models

1/ The Open System Interconnect" model

OSI defines the distribution of roles between protocols.

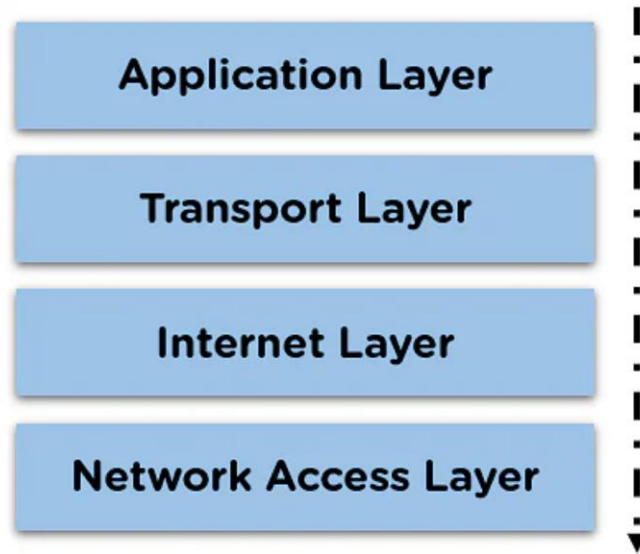
The OSI model is a purely theoretical model: it does not define which protocol to use in which layer.

It is a kind of "best practices" guide.



TCP / IP Stack Versus OSI Model

2/ The reality of the Internet: the TCP/IP protocol stack



Top To Bottom

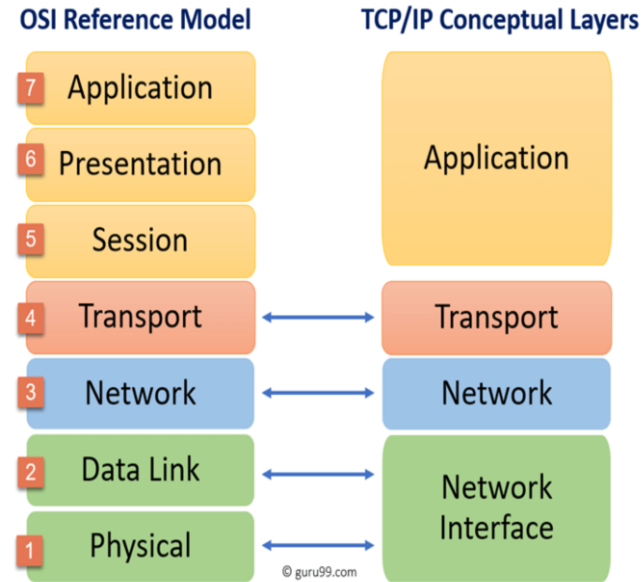
TCP / IP Stack Versus OSI Model

OSI has many more layers of abstraction.
The network access layer is subdivided into 2 layers:

- the "Physical" layer
- the "Data Link" layer

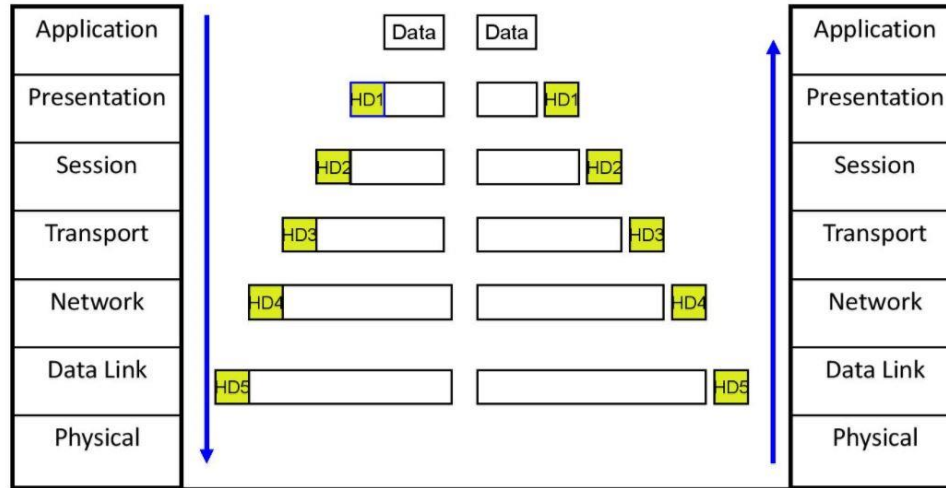
The application layer is subdivided into 3 layers:

- the "Session" layer
- the "Presentation" layer
- the "Application" layer



Encapsulation - Decapsulation

Encapsulation and Decapsulation of Data



<https://networkhunt.com/understanding-7-layers-osi-model/>



IoT Three-Layer Architecture

Why 3 layers?

The three-layer IoT architecture is a high-level classification that provides a basic framework for organizing the components of an IoT system.

It aims to help developers and designers to understand the different components of an IoT system and how they interact with each other.

However, it is important to note that this architecture is not the only possible framework for IoT systems, and that other models and architectures exist.

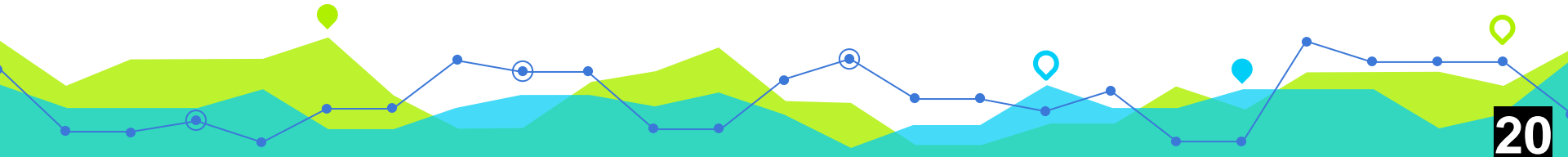
Why 3 layers?

The architecture comprises three tiers:

1- Application layer

2- Network layer

3- Perception layer



Three-Layer architecture: Application layer

1- Application layer:

The application layer is responsible for delivering application specific services to the user. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health. At the application layer, end-users interact with all the connected devices.

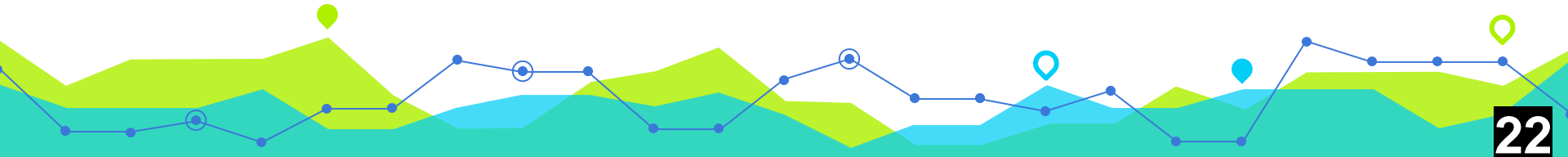
The application layer can include data storage, analytics, machine learning, and other software tools that enable users to monitor and control devices remotely.

Three-Layer architecture: Network layer

2- Network layer:

The network layer is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data.

This layer includes protocols, gateways, routers, and other network devices that enable communication between the perception and application layers.

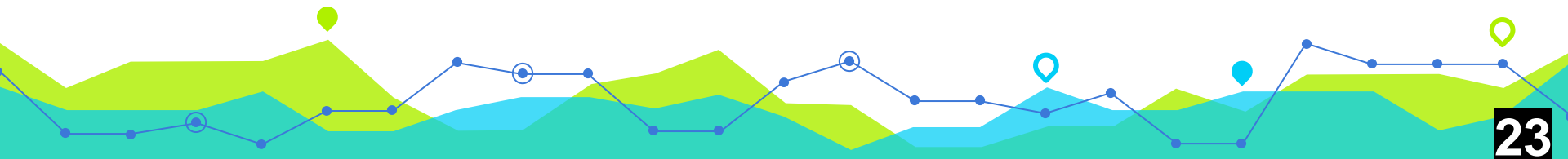


Three-Layer architecture: Perception layer

3- Perception layer:

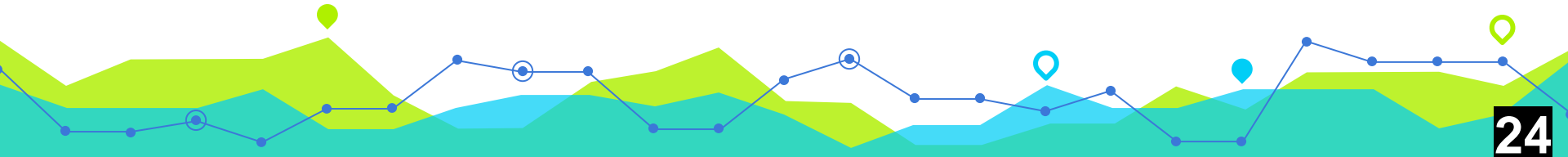
The perception layer is the physical layer, which has sensors for sensing and gathering information about the environment.

It senses some physical parameters or identifies other smart objects in the environment. The data collected in this layer can include temperature, humidity, light, motion, pressure, and other types of sensor data.



Other Architectures?

4 layers, 5 layers, 7 layers...





Back to the conceptual OSI Layers Architecture



Physical
layer

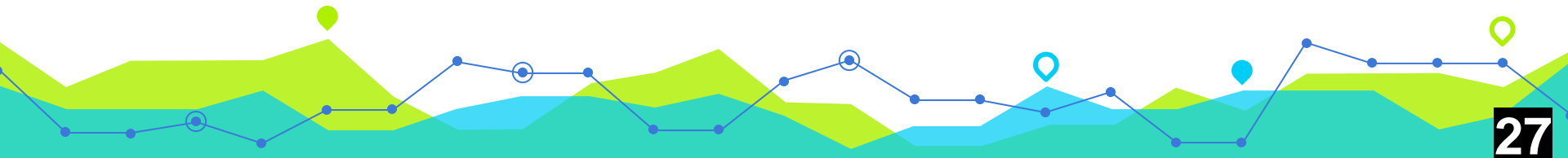
1

Layer 1: Physical layer

The physical layer contains the rules and procedures to be implemented to route the binary elements on the physical medium.

This layer (in other words the protocol that manages this layer) must guarantee the perfect transmission of data on the physical medium: a bit 1 sent must be received as a bit worth 1.

The ultimate goal is then to define how to send a 1 or a 0.

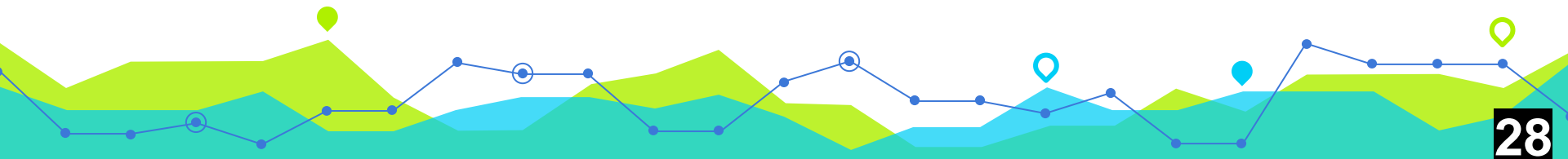


Layer 1: Physical layer

Different physical medium topologies affect the behavior of the physical layer.

In enterprises, for example, cabling plans are sometimes critical to the rest of the architecture.

The physical layer requires reliable hardware, and sometimes the network must be duplicated or meshed to achieve acceptable failure rates.



Layer 1: Physical layer

The typical information unit of this layer is the bit, represented by a certain potential difference.

In practical terms, this layer must normalize :

the physical characteristics (a bit 1 must be represented by a voltage of 5 V, for example)

the mechanical characteristics (shape of the connectors, topology...)

the functional characteristics of the data circuits and the procedures for establishing, maintaining and releasing the data circuit.

Transmission methods

- Unidirectional (simplex): Data is transmitted in one direction only. Example: Television
- Bidirectional (half duplex): Data is transmitted in both directions, but there is only one transmitter at any given time. Example: police radio.
- Simultaneous bidirectional (full duplex): Data is transmitted in both directions, and there may be multiple transmitters simultaneously. Example: telephone

Physical media

- Coaxial cable: Consists of a wire in the center surrounded by a metal braid that acts as a shield.
- Twisted pair: 2 wires twisted to reduce interference are used to transmit data.
- Optical fiber: Very thin cable, made of glass or plastic, has the property of conducting light and is used for lighting or digital data transmission.



Transmission on the medium

Two transmission techniques:

- Base band transmission :

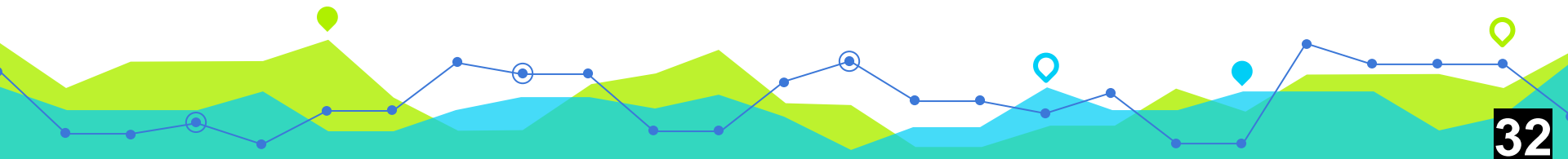
Signals are transmitted as they leave the source.

Used when the signal is accepted as is by the medium.

- Modulation (broadband) transmission:

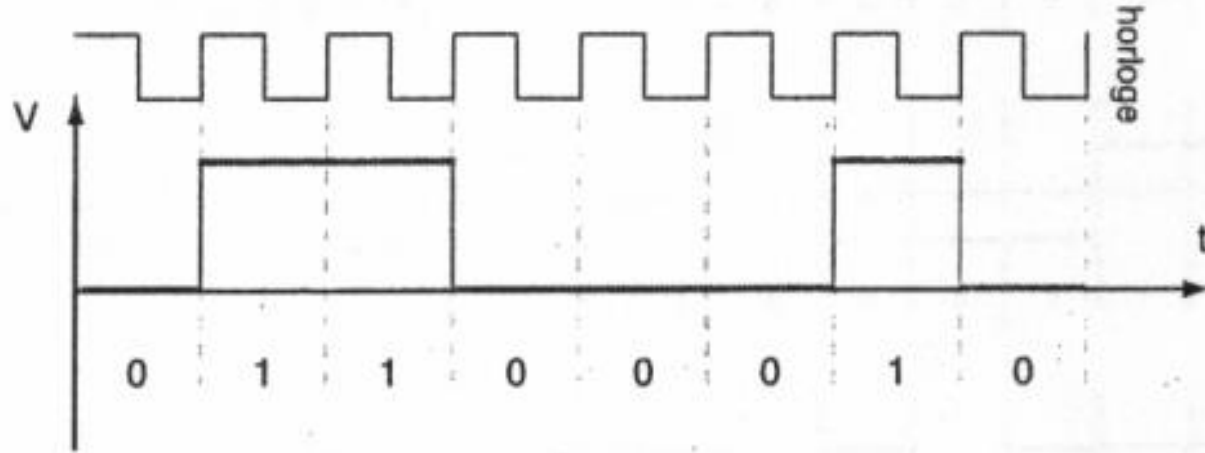
The signal coming out of the source is modified [modem] before being sent

Used when the medium is not well adapted to the signal coming from the source.



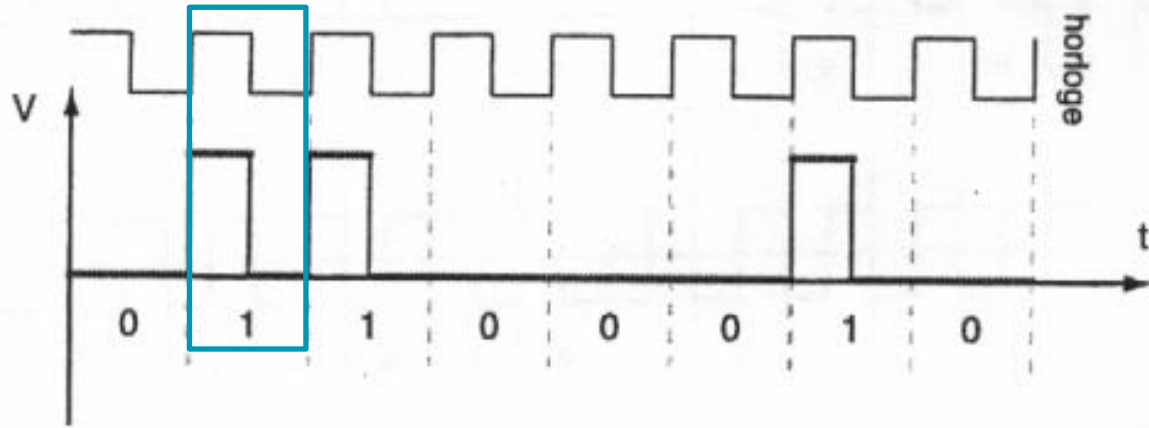
Base band transmission

Code tout-ou-rien



Base band transmission

Code RZ (Return to Zero)

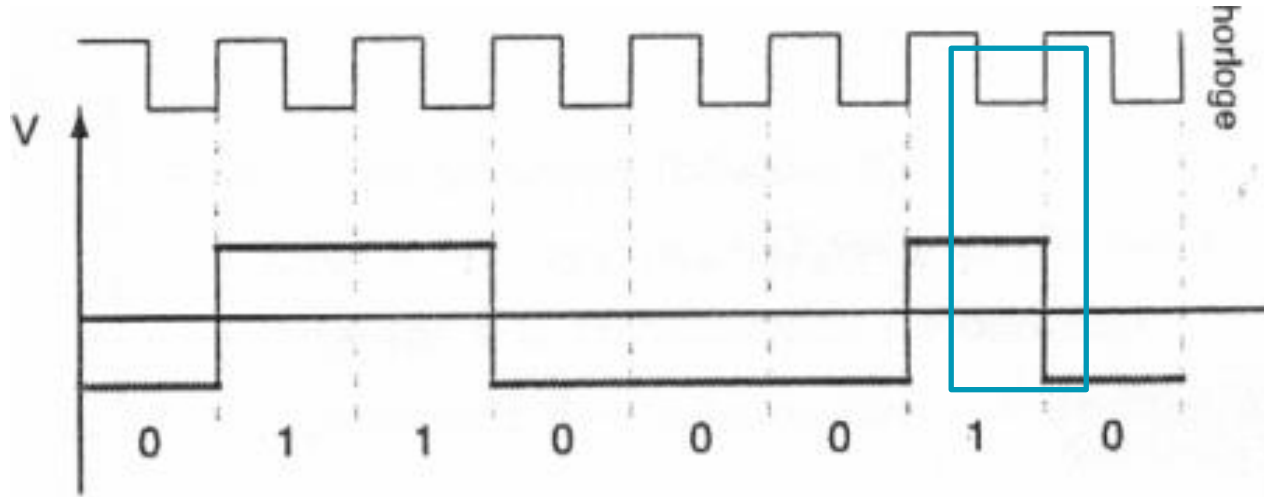


If 0 : $V = 0$ on the period

If 1: rising edge at the beginning / falling edge in the middle of the period

Base band transmission

Non Return to Zero (NRZ)



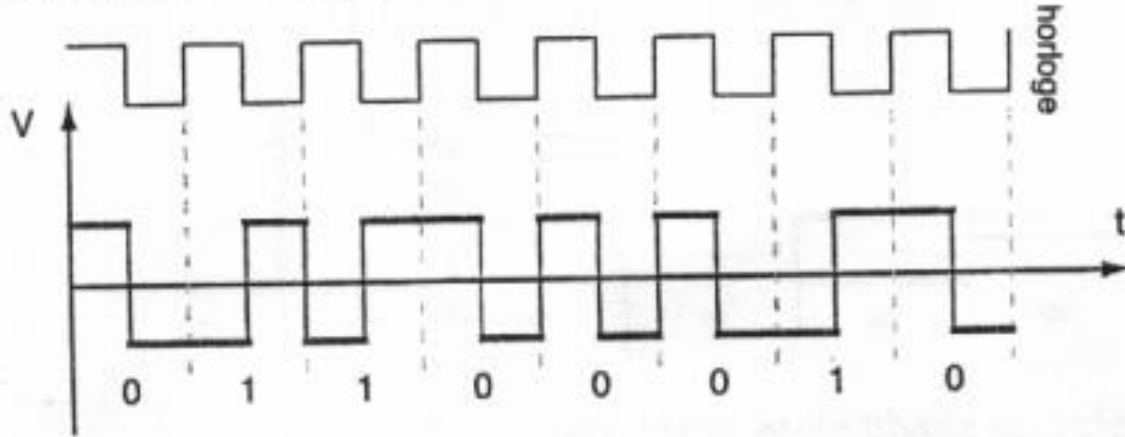
The NRZ (Non Return to Zero) method represents the simplest coding technique.

In this 2-level technique, the digital signal is coded according to the rules :

Data bit at 0 -> negative voltage
data bit at 1 -> positive voltage

Base band transmission

Code biphasé ou Manchester



An upward transition :

(L \rightarrow H) corresponds to the value "1" of the binary signal to be transmitted.

A downward transition: (H \rightarrow L) corresponds to the value "0" of the binary signal to transmit.

This is the coding used for 10 Mbit/s Ethernet.

Exercise

Represent the binary signal 0100 0010 in base band coded according to the all-or-nothing, NRZ, Manchester



EXERCICE

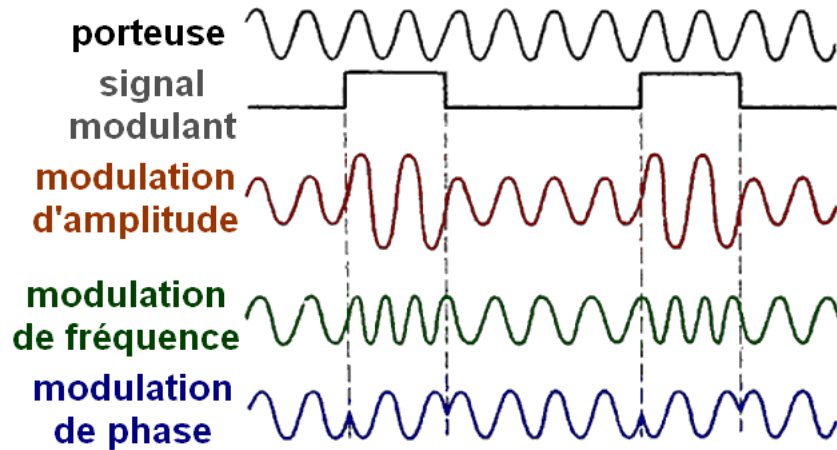
Exercise

Represent the binary signal 0100 0010 1000 0111 in 4-level base band.



Modulated transmission

Modify (modulate) one or more parameters of a carrier wave according to the rhythm of the binary signals to transmit.



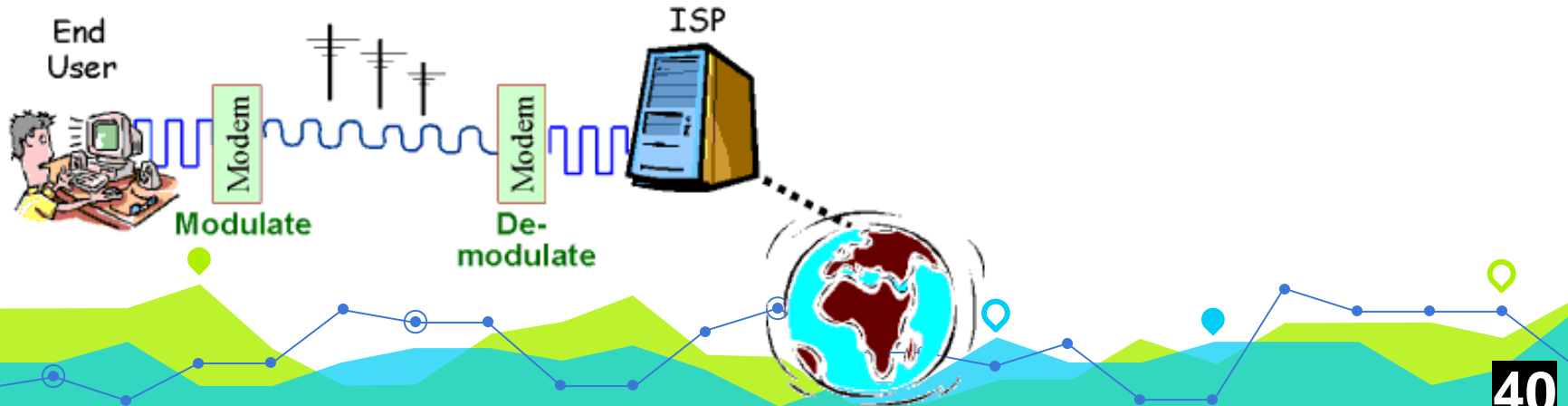
The carrier is a sinusoidal wave, which will see one of its parameters (amplitude, frequency or phase) being modified by the modulating signal. The parameter that varies defines the type of modulation. **We say: the carrier is modulated by a modulating signal**

The modulating signal is the information to be transmitted. This signal modifies one of the parameters (amplitude, frequency or phase) of the carrier. **We say: the modulating signal modulates the carrier.**

Equipment : Modem

The MOdulator-DEModulator will transform the digital signal into a modulated analog signal and vice versa.

Your ADSL box contains an ADSL modem, the numericable box a cable modem, your cell phone a 4G modem...



Equipment: Hub

This wired equipment has several connections to the local network and repeats the signal of each connection on all other connections.

Its sole purpose is to retrieve the binary data arriving on one port and broadcast it to all the ports

The equipment is symbolized as follows:

It is sometimes called a multi-repeater.



IoT protocols operating at OSI layer 1

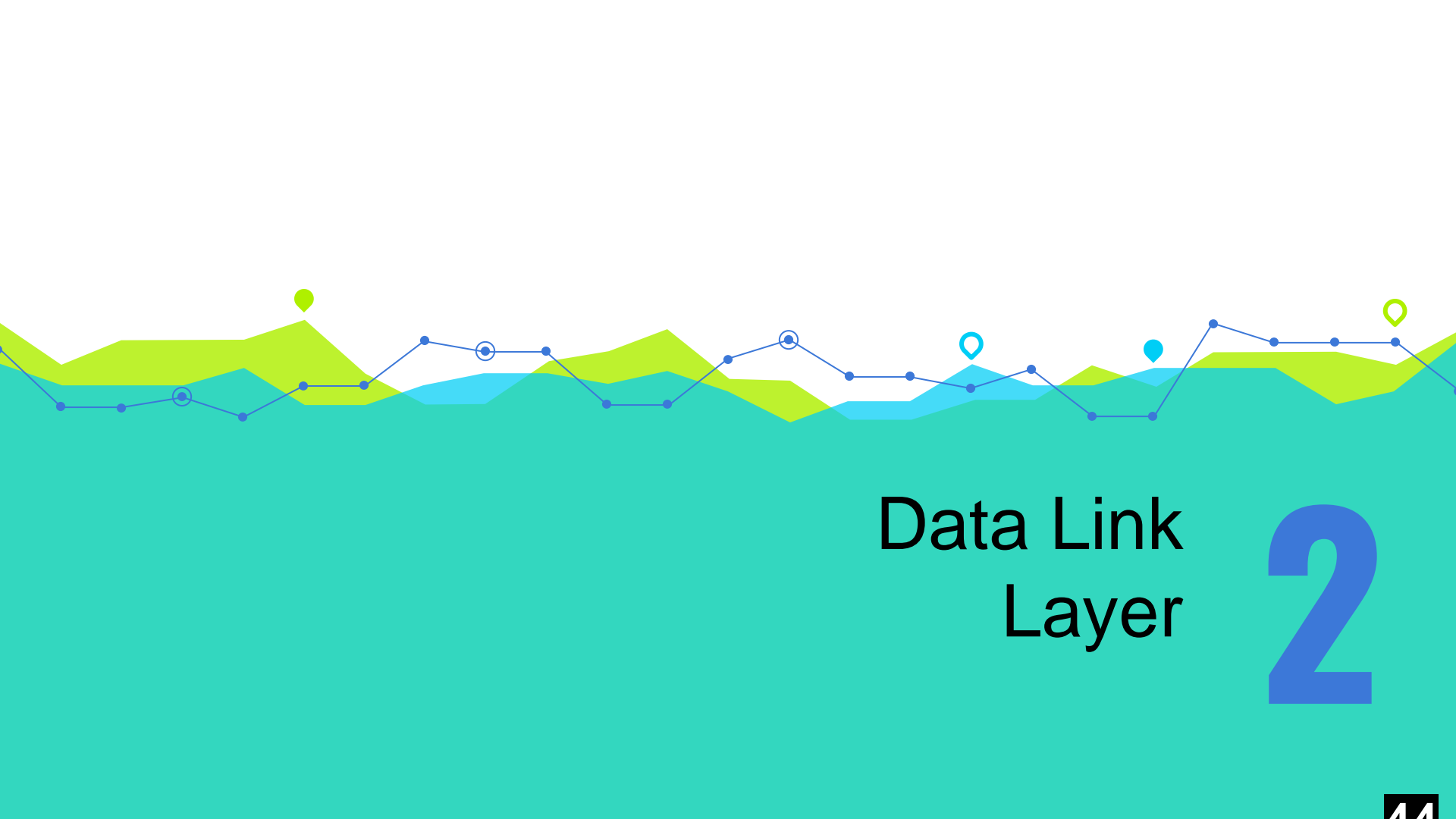
Zigbee: The ZigBee Alliance has developed a set of industrial standards for networking and applications on the basis of the IEEE 802.15.4 standard.

At Layer 1, Zigbee defines the physical characteristics of the wireless communication, including frequency band, modulation scheme, data rate, and transmission power

IoT protocols operating at OSI layer 1

LoRa: In the LoRaWAN protocol, LoRa is the physical (PHY) layer, i.e., the wireless modulation used to create the long-range communication link.

Bluetooth: This includes Bluetooth **Radio frequency (RF) transceiver**, **the physical interface** which defines the electrical and mechanical characteristics of the Bluetooth connection, such as the size and shape of the connector and the type of data transfer (e.g., serial or USB), etc.



Data Link Layer

2

Layer 2: Data link

This layer defines the way and manages the accesses of the network layer to the physical layer: it is the MAC sub-layer.

It creates "frames" that it sends bit by bit to the physical layer.

It also integrates error detection/correction mechanisms.

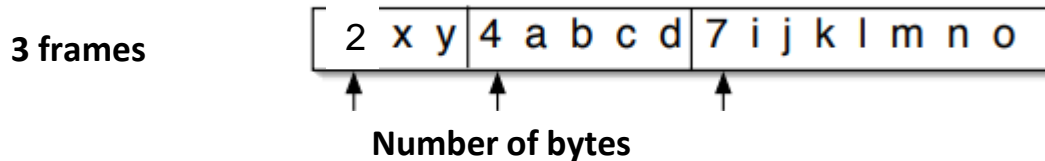


Frame

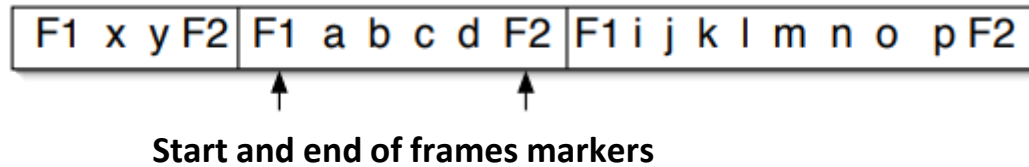
The purpose of putting data in frames is to set a unit for error control.

Several techniques are possible:

- counting characters



- use start and end of frame markers

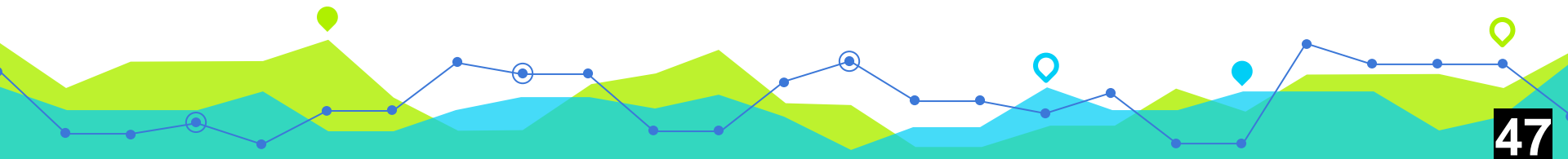


Error detection and correction

The information that arrives from the network layer consists of bits. The link layer interprets it as a frame.

In this frame there could be transmission errors.

It is estimated that there is an error every 1.10^6 bits in a classic Ethernet network.... much more in wireless.



Error detection and correction

If we can determine that a transmission error has occurred at the physical layer, we can :

ask to receive the same frame again (via the physical layer)... but there may be new errors, it may be too late, etc.

try to correct the error at the link layer.



Code theory

How to code messages to facilitate detection and correction of transmission errors?

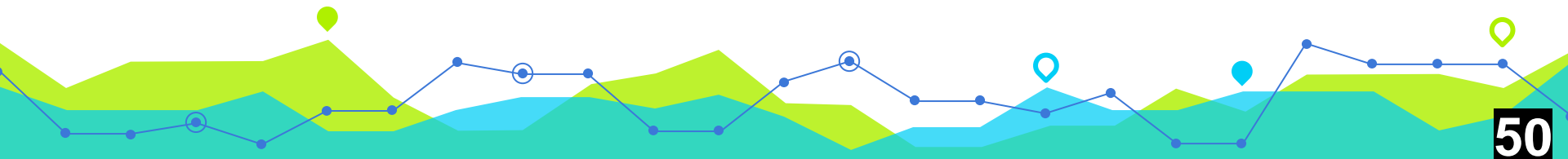
```
00111010101100011111000011111101101100011111001110001  
11110000011100011110100011100011111111111100011100000  
11111110000000
```

Redundancy

In order to detect and correct errors, we will usually repeat the information several times:

- On the phone you repeat the number or spell it out
- Use of the universal radio alphabet (Alpha, Bravo)

Waste of space, reduced throughput... More optimal solution?



Bit of parity

Additional bit added to the data to detect transmission errors.

Even parity bit: a bit worth 0 [respectively 1] is added to the data (at the beginning or end) if the sum of the other bits is even [respectively odd].



Exercise

Consider the message composed of the string: "NET", the transmission control of each character is ensured **by an odd parity bit**.

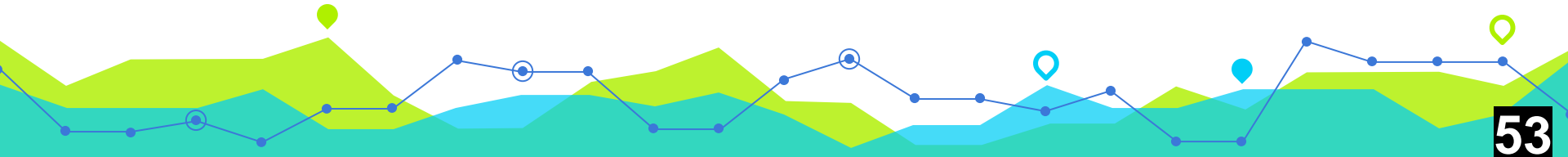
Give the binary representation of the transmitted message. It is supposed that the characters are coded according to the ASCII code, using 7 bits. We recall that the ASCII code of the transmitted characters are : N : 1001110, E : 1000101, T : 1010011.



EXERCICE

Hamming

Hamming distance: Given two words of the same length m_1 and m_2 , we call Hamming distance the number of bits they differ. We note $d(m_1, m_2)$



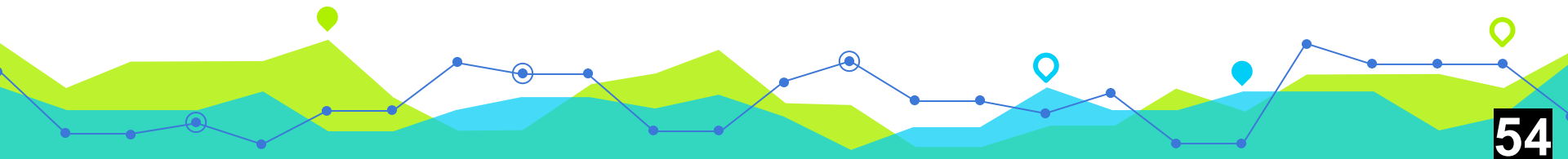
Hamming detection / correction

Hamming distance of the code: minimum of the distance between all words of a code.

With a code having a Hamming distance of d , we can :

- > detect up to $d-1$ errors

- > correct as many as possible $\text{rounding_inf}[(d-1)/2]$



Exercise

An error correcting code contains the following four words:

0000000000 0000011111 1111100000 1111111111

- 1) What is the Hamming distance of this code?
- 2) How many errors can it detect? And how many errors can it correct?



EXERCICE

Equipment: Switch

This active equipment routes "frames" between the different ports that connect it to its local network. It works in "store and forward".

The equipment is symbolized as follows:



Equipment: Switch

Hub vs Switch

- Hub: any packet received is forwarded to all ports.
Disadvantage: the throughput is shared between all connected computers.
- Switch: A switch is at a minimum intelligent and knows to which port it should forward traffic.

Exercise

The Shadok language contains four basic words: ga, bu, zo and meu.

1°) Give a minimal fixed-length binary encoding to encode the above words.

2°) Let the binary encoding be: (ga) = 0000, (bu) = 0110, (zo) = 1001, and (meu) = 1111.

How many errors can it detect? How many errors can it correct?



EXERCICE

Exercise

Give a 2-corrector binary coding of 3 words

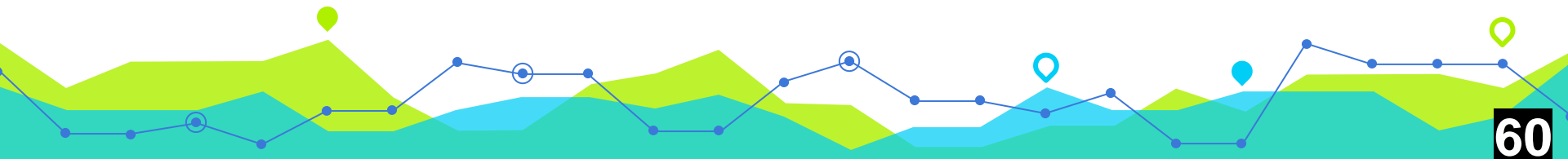


EXERCICE

IoT Protocols operating at OSI layer 2

Zigbee: The ZigBee Alliance has developed a set of industrial standards for networking and applications on the basis of the IEEE 802.15.4 standard.

At Layer 2, Zigbee provides data link layer services, including error detection and correction, packet framing, and medium access control (MAC) protocols.



IoT Protocols operating at OSI layer 2

Bluetooth: Bluetooth technology operates at the data link layer (Layer 2) of the OSI model, which is responsible for establishing reliable communication between two devices over a shared communication channel. This includes **baseband** per example.

Baseband – This protocol takes the services of radio protocol. It defines the addressing scheme, packet frame format, timing, etc

Note that Bluetooth operates as well at the higher layers as well with different protocols

IoT Protocols operating at OSI layer 2

6LowPAN: 6LowPAN uses an adaptation layer between the **network** (IPv6) and **data link layer** (IEEE802.15.4 MAC) to fragment and reassemble IPv6 packets. The routing in 6LoWPAN is primarily divided on the basis of routing decision taken on adaptation or network layer.

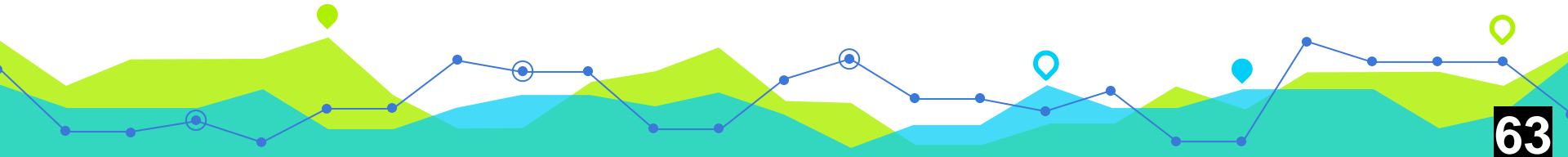
It is designed to enable the transmission of IPv6 packets over low-power wireless networks, such as those used in IoT devices, which typically have low bandwidth, low power, and limited processing capabilities.

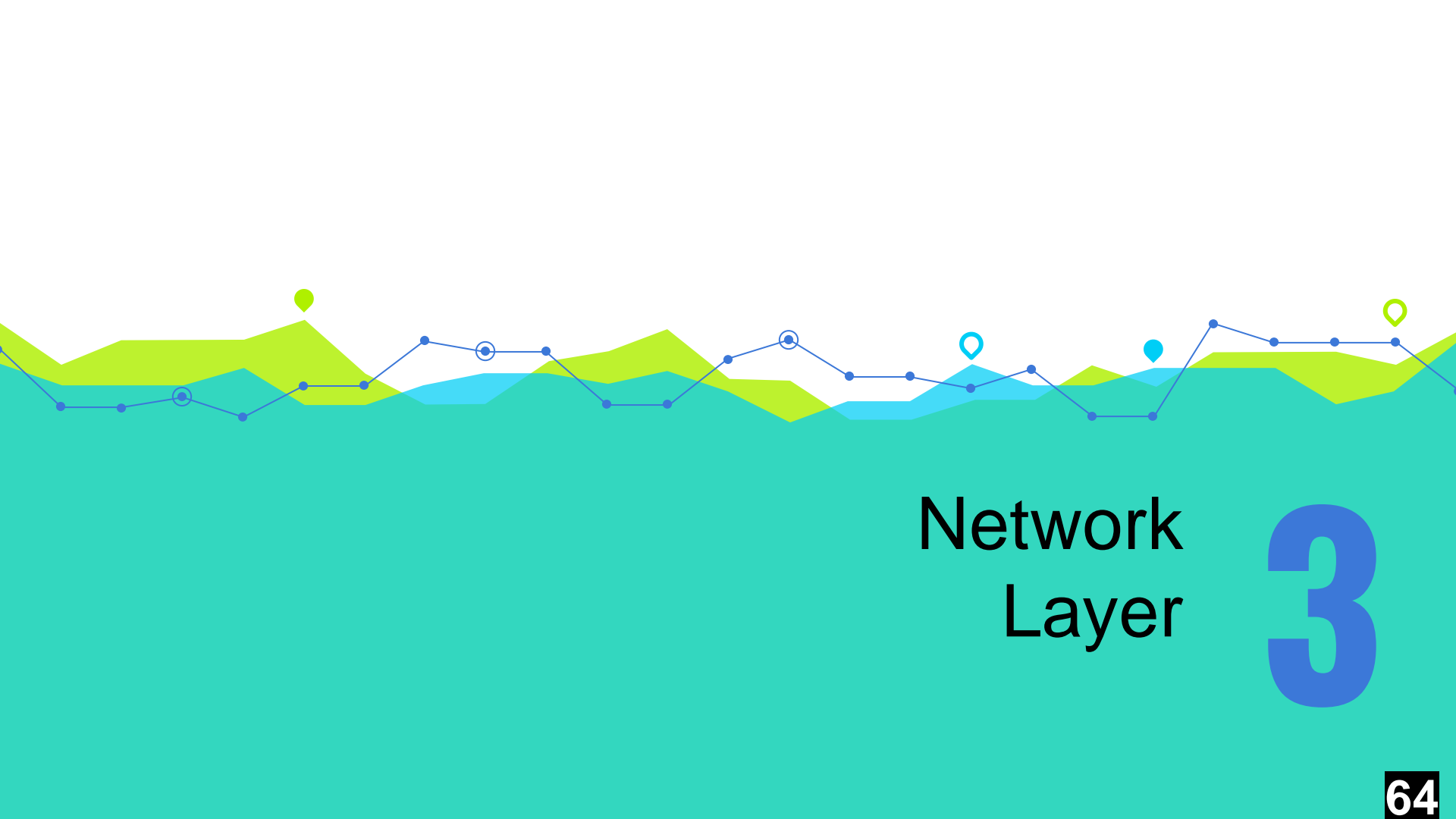
IoT Protocols operating at OSI layer 2

02/10/2023

08:57:03

What about WiFi?





Network
Layer

3

Layer 3

This layer defines the way in which the set of "Network Access" is used to carry information in the form of a "packet" from one point to another.

The "**packet**" is transmitted in a "frame" via layer 2.

The "IP" protocol is, for the Internet and the TCP/IP model, the layer 3 protocol. It exists in two versions:

IP V4 (the most used version)

IP V6 (the new version)



Addressing

The addresses are used to indicate the destination of the information and to be able to locate it via the layer 3 protocol.

There are 3 different types of addresses:

- broadcast" addresses
- multicast" addresses
- unicast" addresses



Broadcast address

We call "broadcast address", The address that identifies the entire community of machines.

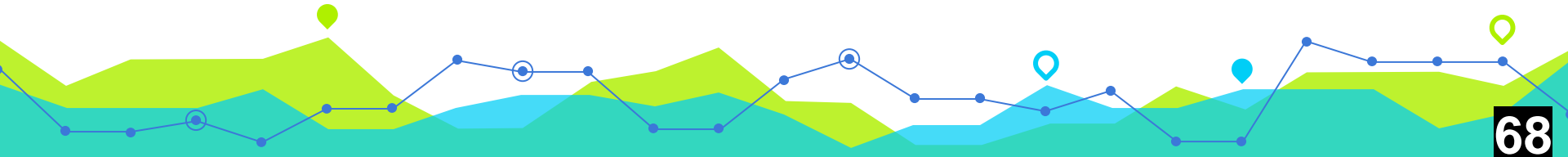
Sending a packet to this address leads to broadcasting this packet to all the machines on the network.



Multicast addresses

These addresses identify machine groups that have chosen to be members. Each address designates a different community.

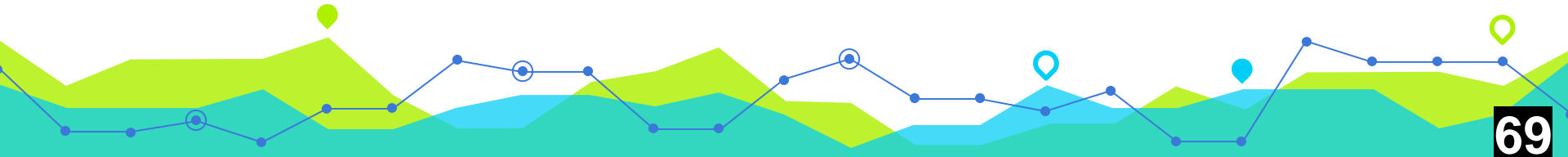
Sending a packet to one of these addresses leads to the packet being broadcast to the entire community designated by this address.



Unicast addresses

These addresses identify each machine in a unique way.

Even though a machine can have several unicast addresses, an address will only correspond to one machine.



Unicast addresses

Due to the lack of addresses in the IPV4 protocol, it was necessary to define private sub-spaces of the Internet called intranets.

The rule of uniqueness of unicast addresses is reduced to one intranet at a time.

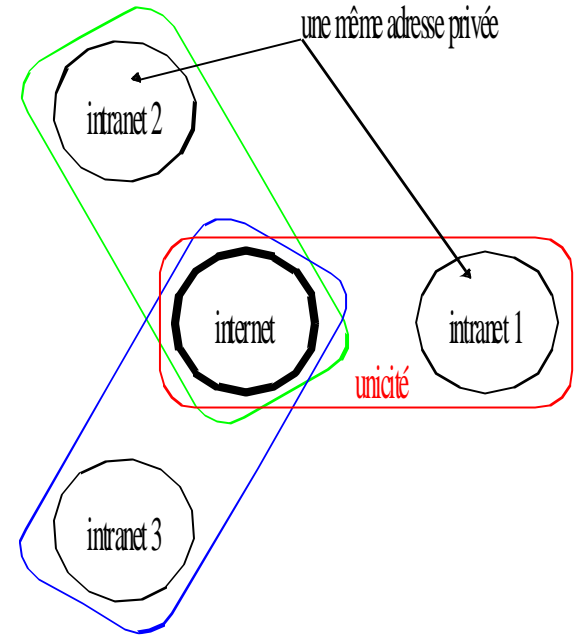
A public address will be unique on the whole Internet while a private address can exist simultaneously in several intranets.



Unicast addresses

- N.B. A device with a private address will not be able to communicate directly with the Internet, it will have to transit the communication via a device of its intranet with a public address.

Technology: proxy or NAT (Network Address Translation).



Equipment: router

This active equipment interconnects several networks (LAN or WAN). It implements at least layers 1 to 3



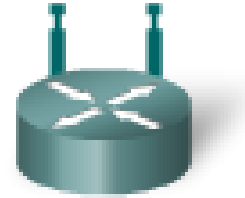
The equipment is symbolized as follows:

Each interface must have its own IP address.

Equipment: wifi router

It is a router interconnecting a WIFI network to another network (LAN or WAN).

The equipment is symbolized as follows:



Each interface must have its own IP address.



Why IP V6?

With its 32-bit address format, IPv4 can handle a maximum of 4.3 billion unique IP addresses.

While this number may seem very high, it is not enough to support the rapid growth of the Internet.



Why IP V6?

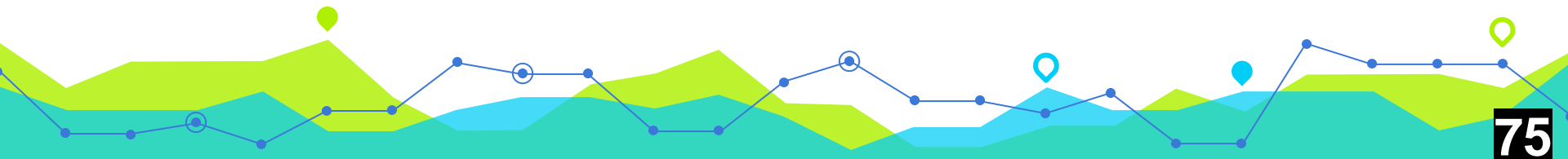
With its 128-bit address format, IPv6 can support 3.4×10^{38}

or

340,282,366,920,938,463,463,374,607,431,768,211,456 unique IP addresses.

This number of addresses is large enough to set up a unique address on every node on the Internet and still have plenty of addresses left.

It is also large enough to eliminate the need for NAT



IP V6 addresses

The IPV6 standard defines an address format on 128 bits, i.e. 16 bytes.

They are written as 8 integers in hexadecimal (between 0000 and ffff) and separated by colons.

Example: 0000:0000:0000:0000:0000:0000:0000:0001

Note: There are rules for simplifying the writing.

IP V6 addresses

- Deletion of the insignificant 0's in each group of 16 bits.

2031:0000:130F:0000:0000:09C0:876A:130B

→ 2031:0:130F:0:0:9C0:876A:130B

- Replacement of the longest null value sequence with "::"

2031:0:130F:0:0:9C0:876A:130B

→ 2031:0:130F::9C0:876A:130B



Exercise

Simplify the following addresses:

- 1) 2041:0000:140F:0000:0000:0000:875B:131B
- 2) 2001:0db8:0000:0000:0000:ff00:0042:8329
- 3) 2001:0001:0002:0003:0004:0005:0006:0007
- 4) 2340:0023:AABA:0A01:0055:5054:9ABC:ABBO
- 5) 1454:0045:0000:0000:4140:0141:0055:ABBB



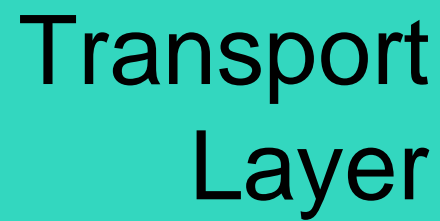
EXERCICE

IoT Protocols operating at OSI layer 3

RPL: RPL operates at Layer 3 of the OSI (Open Systems Interconnection) model and uses a directed acyclic graph (DAG) to organize the network topology. Each node in the network maintains a parent-child relationship with other nodes in the network, forming a DAG that represents the network topology.

See course 1





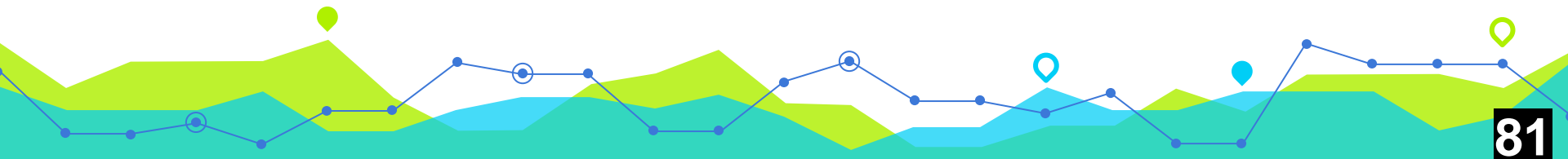
Layer 4

This layer defines the methods for managing end-to-end communication.

It meets the requirements demanded by communicating applications.

There are two types of communication:

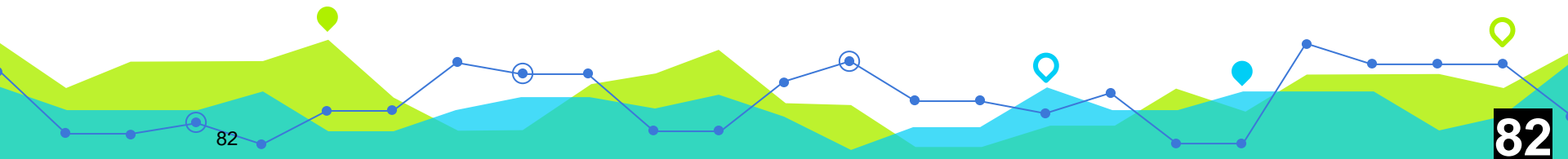
- by sending datagrams
- by opening a connection



Datagram

It is a piece of information, of limited size, which is sent over a network from one machine to another.

In TCP/IP, the UDP protocol is responsible for routing datagrams between applications via the IP layer.



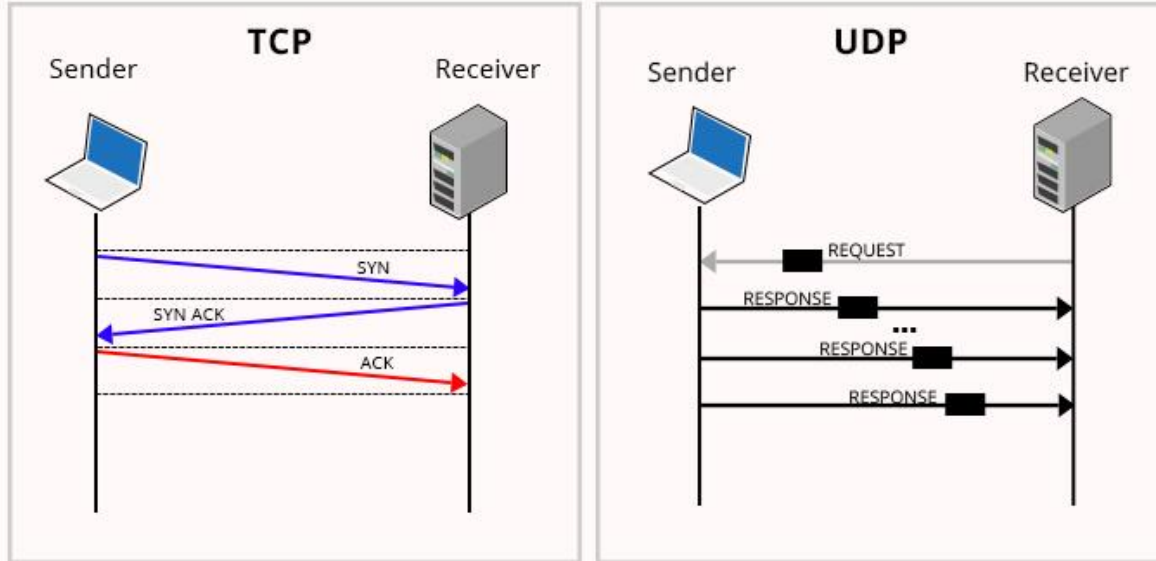
TCP: The connected mode

This form of communication opens a channel between the two machines that guarantees a quality of routing (nothing is lost, everything arrives in order).

In TCP/IP, the TCP protocol is responsible for establishing and maintaining the connection. The data flow is divided into segments whose orderly delivery is guaranteed and TCP reconstitutes the flow at the end of the channel.



TCP Vs UDP Communication



Exercise

You are doing a video conference via Skype.

In your opinion, does Skype use UDP or TCP for the video?



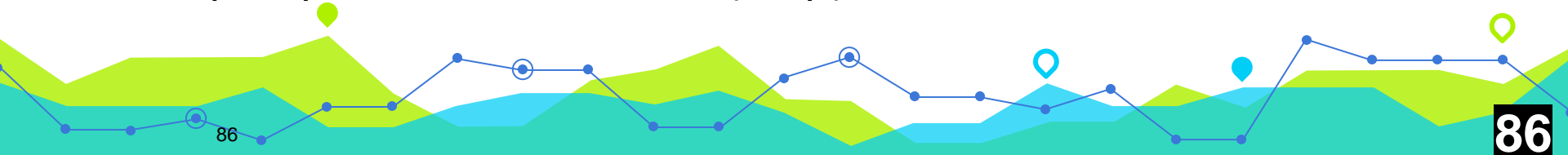
EXERCICE

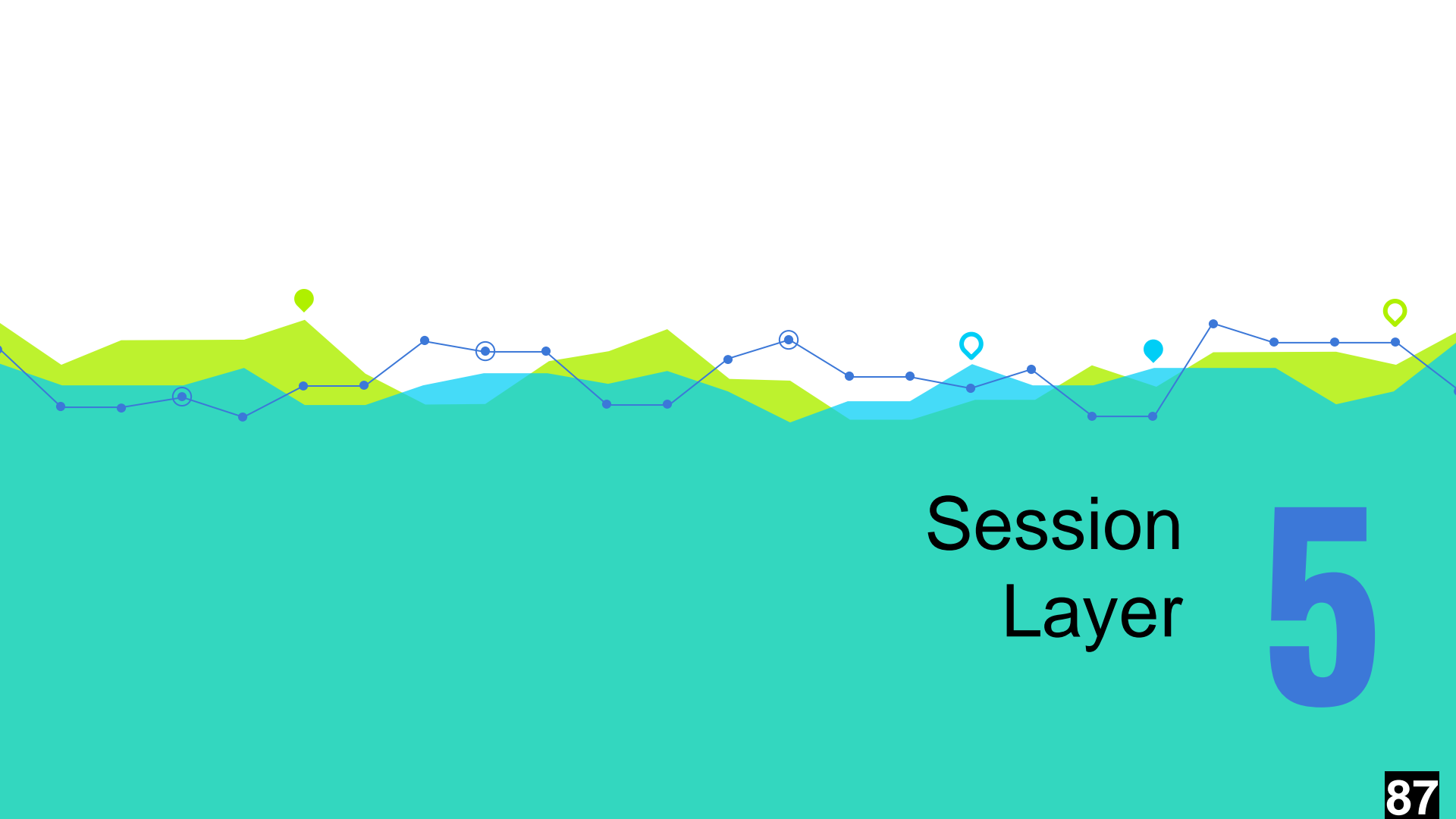
Port number

While the network layer ensures communication between machines, the transport layer must be able to identify the application that is receiving the communication.

The TCP and UDP protocols use a port number (coded on 16 bits) between 1 and 65535. The port numbers of "server" applications are standardized according to the type of service provided (and therefore the application protocol)

Example: port 80 for the web (http)



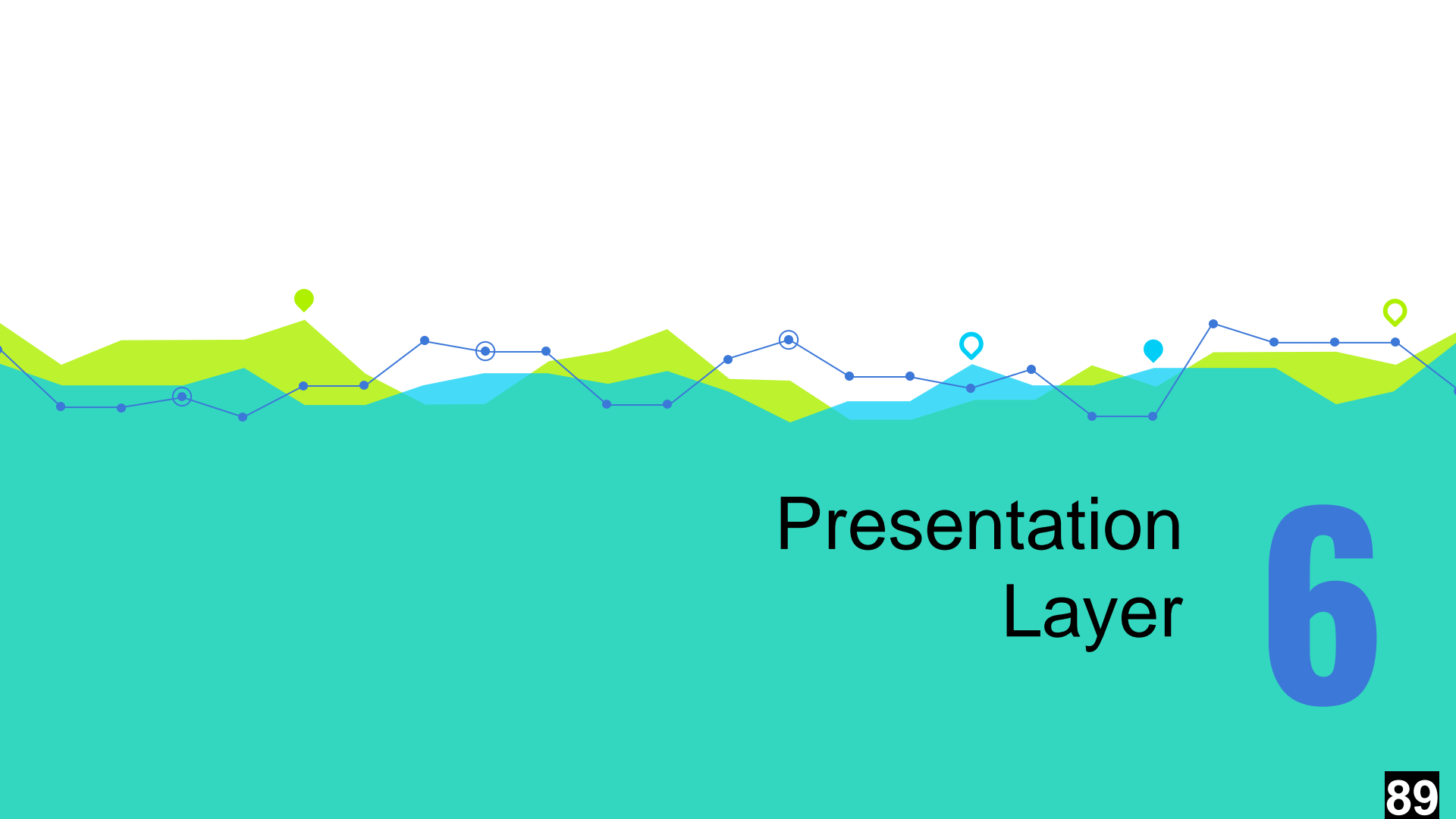


Layer 5

session layer of the OSI model is responsible for managing communication sessions between devices.

It provides the mechanism for opening, closing and managing a session between end-user application processes, i.e., a semi-permanent dialogue.





Presentation Layer

6

Layer 6

This layer defines the methods for managing the problems of transmitting information regardless of its binary representation.

This transmits the information coding formats and implements transcoding mechanisms when the source and recipient do not have the same representation systems.

Example: switch from ASCII to UNICODE





Application
Layer

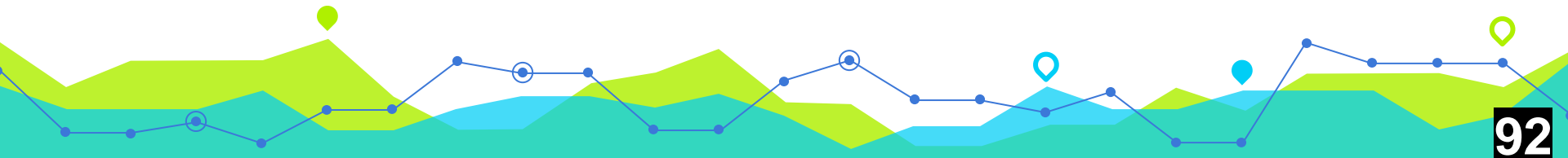
7

Layer 7

This layer contains the applications (clients or servers) that use the network stack to exchange data.

In the TCP/IP model, the application layer must deal with the problems of the session and presentation layers, which are absent from the network stack.

Application protocols define how applications use the network stack to communicate.

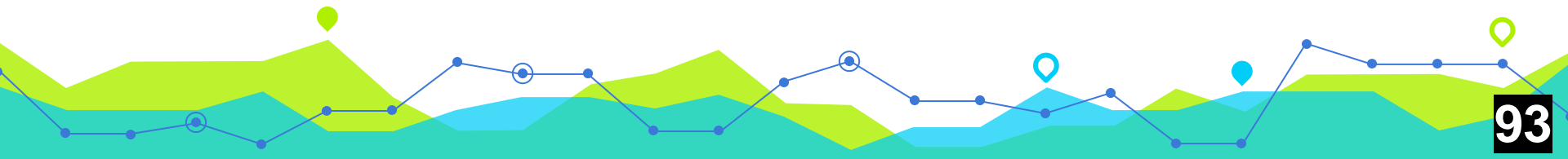


Application protocol

In TCP/IP mode, applications (client and server) exchange information via datagrams (UDP) or via a connection (TCP).

The protocol defines how UDP or TCP is used.

One or more standardization documents called "RFCs" define how the protocol works.



UDP-based application protocol

In the case of a dialogue via UDP, the client initiating the request sends a first datagram to the server.

The protocol defines the structure (number, size and coding) of the information transmitted in this datagram.

As a general rule, the server responds by sending a datagram to the client. The structure of this datagram will also be defined by the protocol.

TCP-based application protocol

In the case of a dialogue via TCP, a bidirectional channel is opened, at the request of the client, to the server.

The protocol defines how this channel will be used by defining :

- which of the client or the server sends the data in the channel first
- who indicates the change of sender (switch to client or server)
- when to stop the connection.



The services

Depending on the service provided, different application layer protocols have been defined:

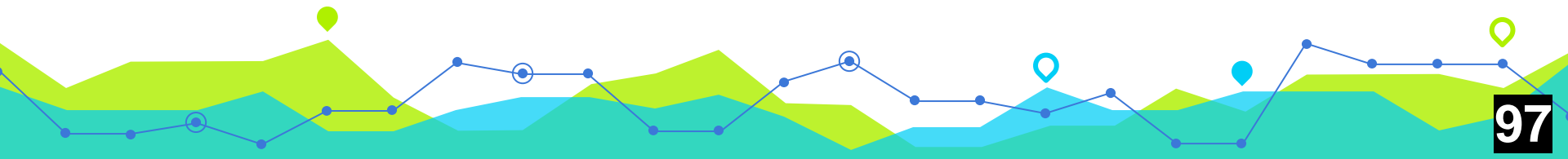
- The "Domain Name Service" (or DNS) via UDP
- The Web (http, https) via TCP
- File transfer (ftp) via TCP
- SSH (including SCP) via TCP
- MQTT via TCP

Etc

DNS

The DNS service was built so that each machine (DNS client) obtains the correspondences between the complete names (FQDN for "Fully qualified domain name") and the IP addresses (V4 or V6).

On each machine, the address of its DNS server is therefore indicated in the IP configuration.



The Web

The http protocol was defined to allow the transfer of multimedia content between the browser (the client) and the web server.

It uses a TCP connection and dialogues via human readable text (ASCII).

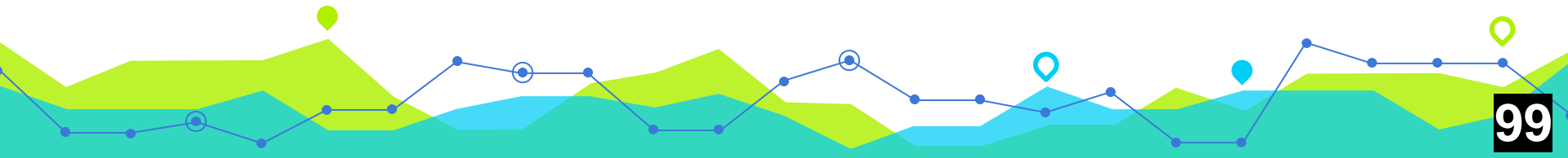


SSH

Secure Shell, sometimes called Secure Socket Shell, is an application protocol that allows you to securely connect to a remote computer or server using a text-based interface.

When a secure SSH connection is established, a shell session will be launched, and you will be able to manipulate the server by typing commands into the client on your local computer.

Example of the Lab: `ssh userxx@212.36.220.222`



SSH

System and network administrators are the ones who use this protocol the most, as well as anyone who needs to manage a computer remotely in a highly secure way.

This will be your case in the raspberry lab and in the projects
It was also your case in the Linux and python labs

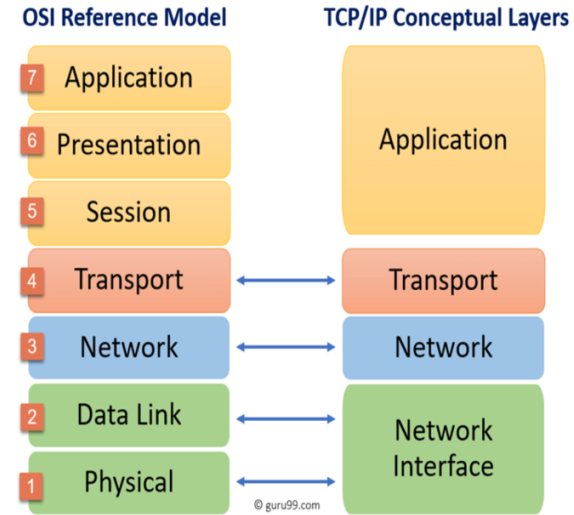


SSH - use case: Communication via a Raspberry Pi

Example of CLI (Command Line Interface) type control, i.e. via a Bash console, of a Raspberry Pi in wired and via the network (wired or wireless), to replace the direct connection via HDMI screen and USB keyboard.

Case 1 : *Communication via RS232 (Layer 1 and 2 in the OSI model) - Network access layer in the TCP/IP model)*

Cas 2 : *Communication via Telnet (Layer 7 in the OSI model) and SSH (Layer 5 and 7 in the OSI model). For the 2 protocols (Telnet and SSH) it is the application layer in the TCP/IP model.*



SSH - use case: Communication via a Raspberry Pi

Case 1: Wired communication via RS-232

RS-232 (sometimes called EIA RS-232, EIA 232 or TIA 232) is a standard for a serial communication channel. It was available on almost all PCs from 1981 until the mid-2000s. (see <https://fr.wikipedia.org/wiki/RS-232>)



It was essentially used to connect a peripheral (for example a printer) to a computer. The physical RS232 connector is DB9 (see DB25 for the oldest) male or female depending on the equipment.

SSH - use case: Communication via a Raspberry Pi

Case 1: Wired communication via RS-232

RS232 connection via USB to a PC:

Nowadays the RS-232 interface is only rarely integrated in the PC but can be added on the USB bus:

On one side, we find the DB9 male corresponding to the RS232 interface of a computer.

On the other side, there is the USB connector.

To use this type of cable, a specific driver must be used.

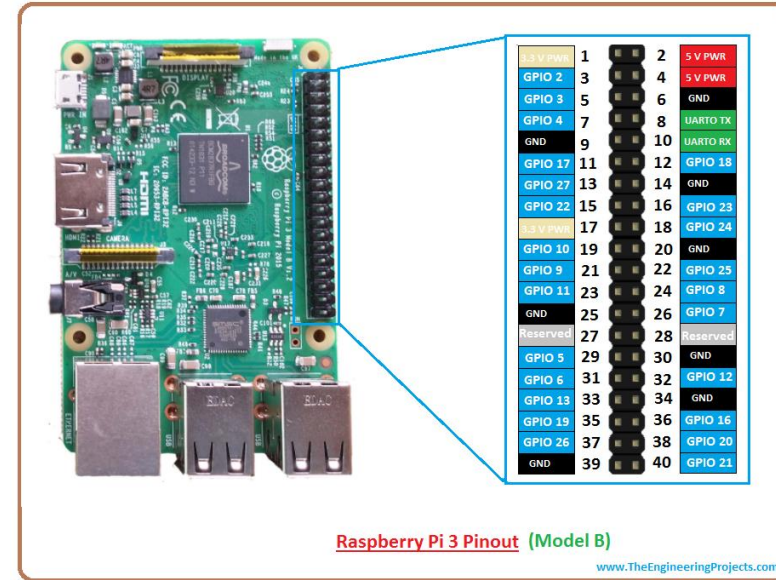


SSH - use case: Communication via a Raspberry Pi

Case 1: Wired communication via RS-232

The RS232 connection on a Raspberry Pi:
example for a Raspberry Pi 3.

The GPIO (General Purpose Input Output) integrates two pins RX and TX (8 and 10 respectively) which are used for RS232 communication (in 3.3 volts)



SSH - use case: Communication via a Raspberry Pi

Case 1: Wired communication via RS-232

The RS232 connection on a Raspberry Pi: example for a Raspberry Pi 3.

Attention!!

The RS232 works according to the standard in 12 volts. It is therefore necessary to connect an RS232 voltage converter as shown in the photo.



To go further and for the curious among you ☺

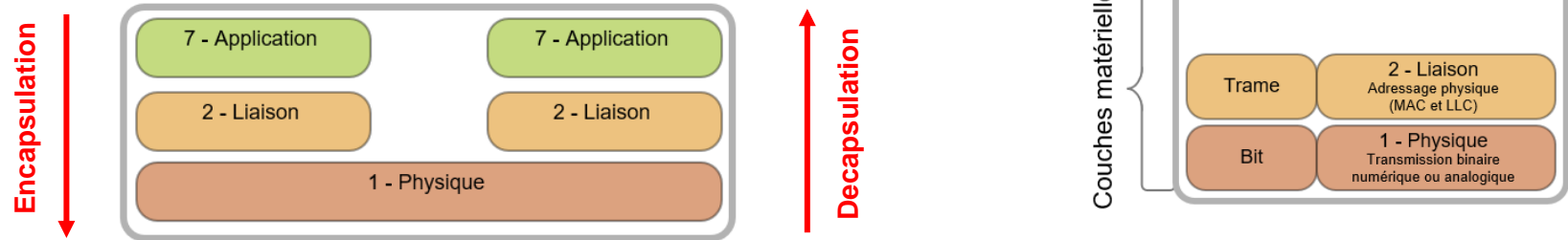
<https://www.framboise314.fr/le-port-serie-du-raspberry-pi-3-pas-simple/>

SSH - use case: Communication via a Raspberry Pi

Case 1: Wired communication via RS-232

The layers 3 to 6 do not exist because the functionalities they provide are not part of the requirements.

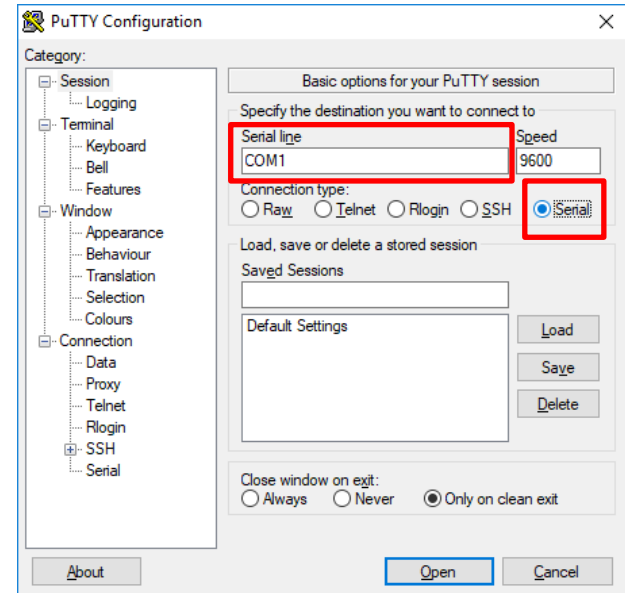
Layer 7 receives directly the characters extracted from the frame.



SSH - use case: Communication via a Raspberry Pi

Case 1: Wired communication via RS-232

In order to establish the communication on the PC side, you have to use an application like PuTTY

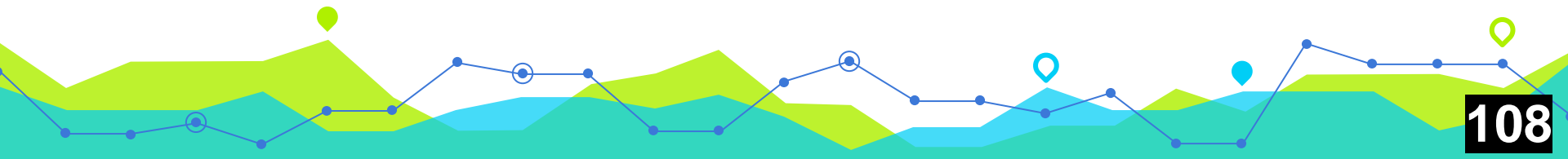


SSH - use case: Communication via a Raspberry Pi

Case 2: Communication via the network

This time, we want to be able to access the CLI of an equipment without having a direct RS232 between the latter and the administration computer, but by using the network infrastructure that connects them.

We use potentially all types of communication support available on the Internet.



SSH - use case: Communication via a Raspberry Pi

Case 2: Communication via the network

Telnet and SSH are two application protocols that allow the emulation of a bidirectional RS232 type channel between two devices (thus the exchange of data). They differ from each other on several aspects related to security.

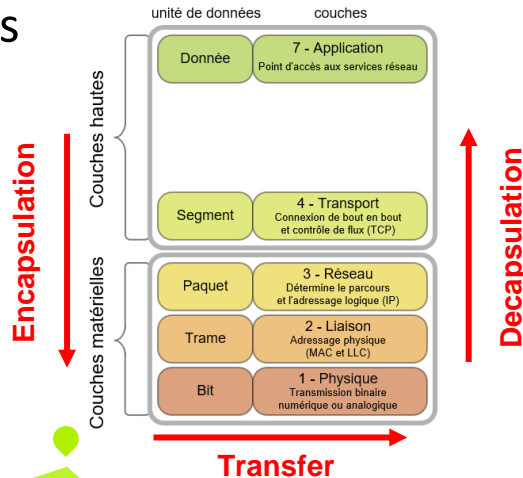
Criteria	Telnet	SSH
Port number	23	22
Security	No	Yes
Authentication	No	Yes
Bandwidth usage	Light	Heavy
File transfer	No	Yes

SSH - use case: Communication via a Raspberry Pi

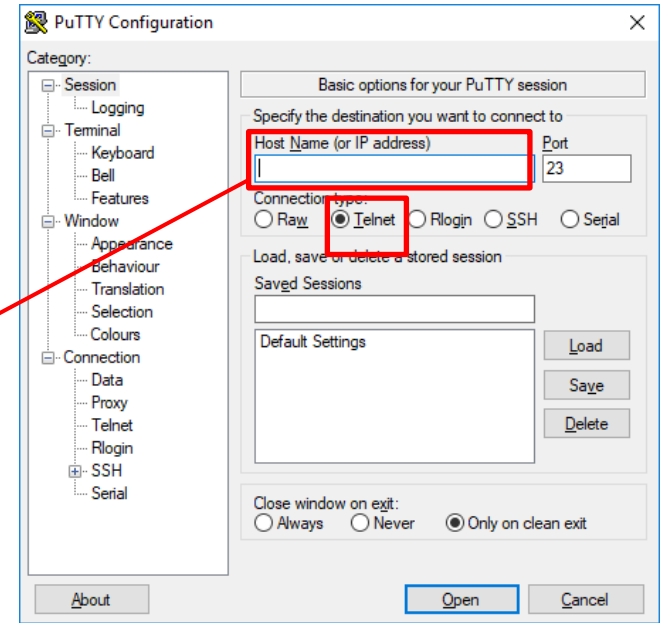
Case 2: Communication via the network

In order to establish the communication on the PC side, you have to use an application like PuTTY under windows

For Telnet ->



IP address
or DNS

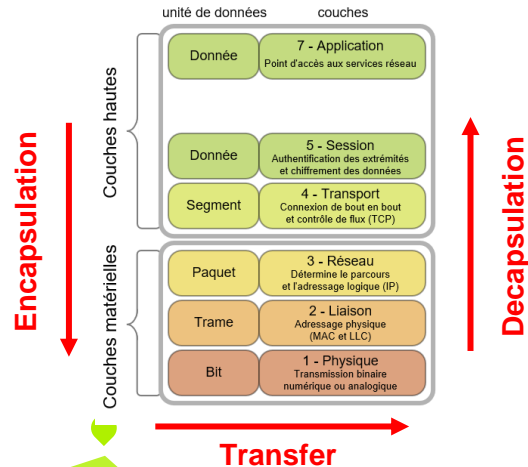


SSH - use case: Communication via a Raspberry Pi

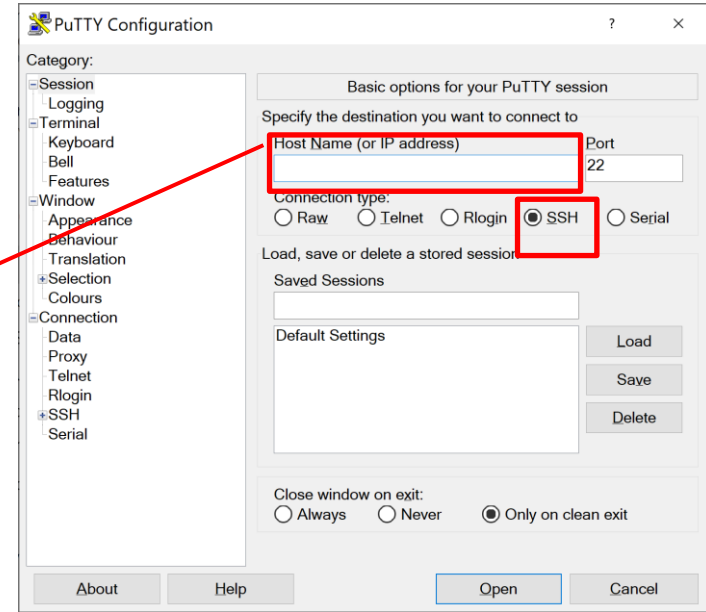
Case 2: Communication via the network

In order to establish the communication on the PC side, you have to use an application like PuTTY with Windows

For SSH ->



Adresse IP ou
Nom DNS



SSH - use case: Communication via a Raspberry Pi

Case 1 and Case 2: For MAC or Linux

With MAC or Linux, you must use the following commands in the SHELL Bash (Terminal):

- For RS232: screen *Eg: screen /dev/cu.usbserial 9600*
- For Telnet: telnet *Eg: telnet 212.36.220.222*
- For SSH: ssh *Eg: ssh userxx@ 212.36.220.222*

SSH: SCP

To do file transfer between a client and an SSH server you need to use the scp command on MAC or Linux (and on a recent version of windows 10) in the following way:

```
scp <source> <destination>
```



SSH: SCP

Example from your computer to download exercise ex1.py from server [212.36.220.222](#) of user01 in the home directory to the current directory:

```
scp user01@ 212.36.220.222:/home/user01/ex1.py .
```

Source

Destination

Example from your computer to upload on the server [212.36.220.222](#) in the personal directory of the user01 the exercise ex1.py from our current directory:

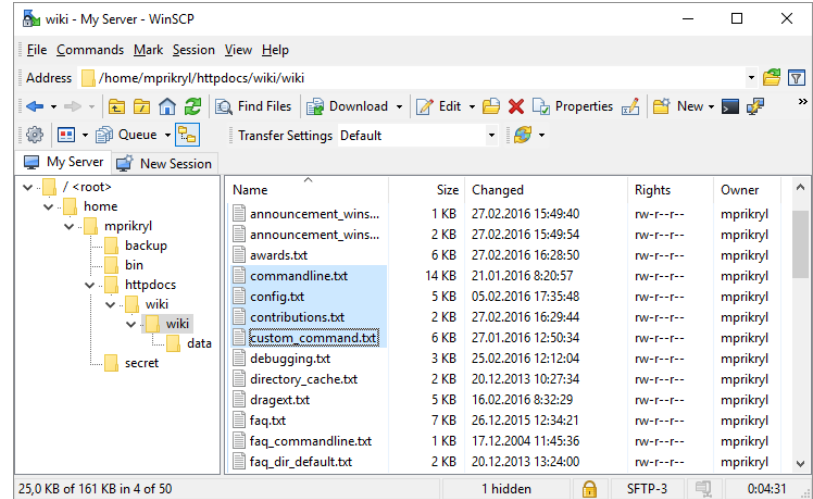
```
scp ex1.py user01@ 212.36.220.222:/home/user01
```

Source

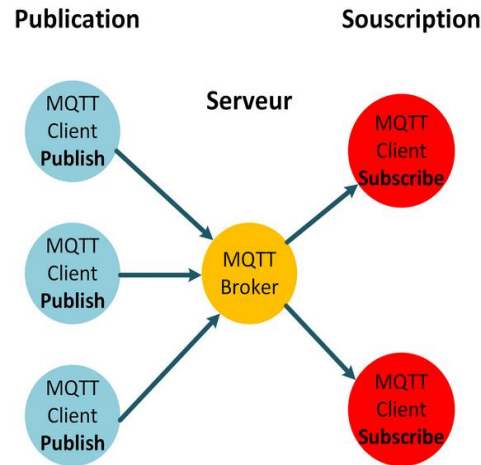
Destination

SSH: SCP

On Windows (in general):
To transfer files between a client
and an SSH server, the WINSCP
program must be installed on the
client (the computer)



MQTT (Message Queuing Telemetry Transport) is a publish-subscribe messaging protocol based on the TCP/IP protocol

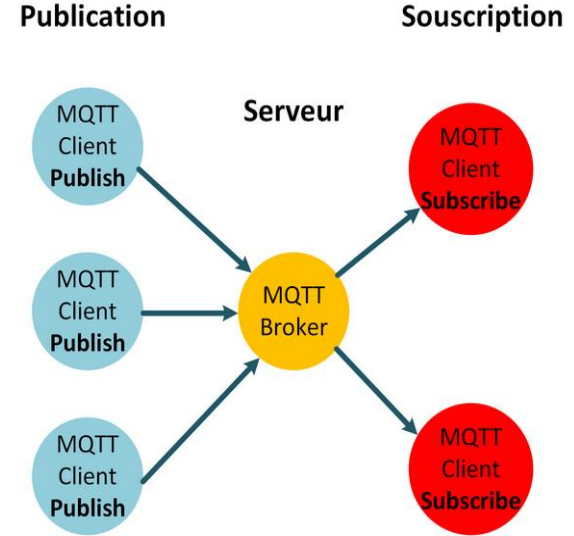


https://www.framboise314.fr/utiliser-le-protocole-mqtt-pour-communiquer-des-donnees-entre-2-raspberry-pi/#Le_protocole_MQTT

MQTT

The diagram on the right shows the principle of operation of MQTT.

On the left are the "publishers". These are machines (Arduino, ESP8266, Raspberry Pi...) that capture values (temperature, humidity, pressure, power consumption, water consumption...) and send them to a server called "Broker".



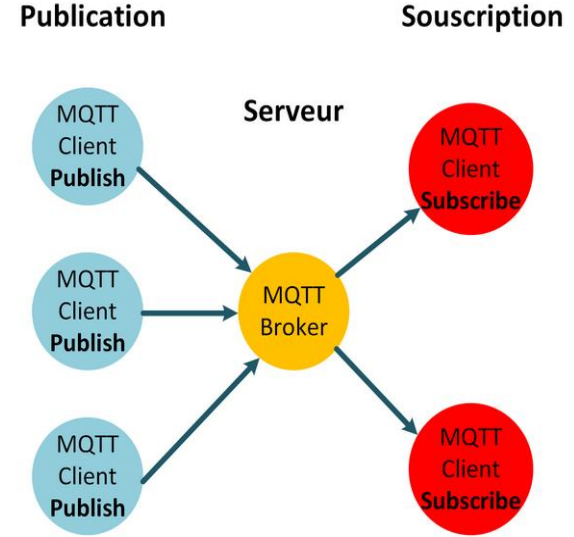
https://www.framboise314.fr/utiliser-le-protocole-mqtt-pour-communiquer-des-donnees-entre-2-raspberry-pi/#Le_protocole_MQTT

MQTT

"Subscriber" customers are connected to the "Broker". They have requested certain data to be sent.

When the Broker receives this data, it retransmits it to the clients who have subscribed to this data flow. For example, one of the subscribing customers will receive the temperature and humidity to make a real-time graph.

The other customer will receive the water consumption and will trigger an alert if it exceeds a specific value.



https://www.framboise314.fr/utiliser-le-protocole-mqtt-pour-communiquer-des-donnees-entre-2-raspberry-pi/#Le_protocole_MQTT

MQTT: The Topics

MQTT clients register with the broker on topics, a kind of access path to a resource.

These paths start with the character "/" and are composed of a list of words separated by "/"

Example:

/sensor/1/temperature.

N.B: No spaces in the words and no accented characters

MQTT: The Topics

Publishing clients post on a given topic

Subscribing clients ask to be notified when someone posts on these topics.

This can be a temperature topic for example:

/sensor/1/temperature.



MQTT: The Topics

You can subscribe to a set of topics by using # or + wildcards.
For example, if a client publishes on topics:

/sensor/1/temperature

And

/sensor/1/humidity

another client can listen to both topics at the same time:

/sensor/1/#.



MQTT: The Topics

If several clients publish their temperatures and humidities by inserting their client number on their topics, another client can listen to all temperatures this way:

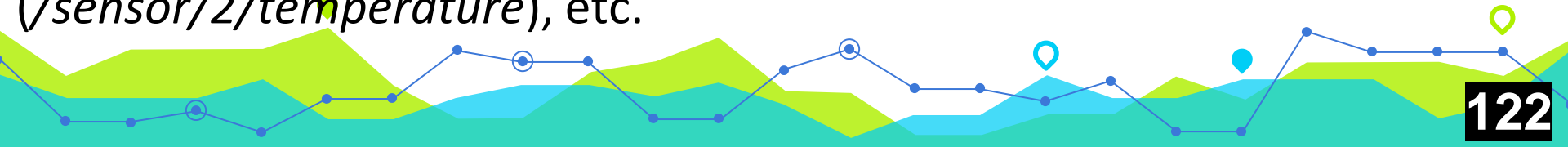
/sensor+/temperature.

It will then receive the temperatures of client 1:

(/sensor/1/temperature)

And client 2 :

(/sensor/2/temperature), etc.



MQTT: Topic usage

For sending information we can have two possibilities of use:

a) 1 single topic for an elementary typed information.

Example: int, float, string

b) 1 topic for a set of linked information.

Example: an int, a float and a string in an object (list, data structure...) which will be exchanged on a topic.

=> Need for serialization => Use of JSON or XML

MQTT: Topic usage

Example:

Consider the set of information to be transmitted below:

Place, Temperature, Pressure



*Place of taking
of the measurement*

*room
temperature*

Pressure



MQTT: Topic usage

Example:

For case a: one topic per information

Place, Temperature, Pressure

Place of taking
of the measurement

room
temperature

Pressure

~~/Application/Sensor/Place/Température~~

No accentuated characters!

Exemple: Lille

/Application /Sensor/Place/Temperature

Example: 22.5

/Application /Sensor/Place/Pressure

Example: 720



MQTT: Topic usage

Example:

For case b: linked information

/Application/Sensor

Place, Temperature, Pressure

Place of taking
of the measurement

room
temperature

Pressure

Example in JSON: `{'Place':'Lille','Temperature':22.5,'Pressure':720}`

Exercise: How do you encode/decode this message in python?


MQTT: Topic usage

Example:

For case b: linked information

/Application/Sensor

```
Import json
m=["Lille", 22.5, 720]
type(m) # liste
json.dumps(m) # encode in json
type(m) # string
json.loads(m) # decode to python
type(m) # list
```



Example in JSON: {'Place':'Lille','Temperature':22.5,'Pressure':720}

Exercise: How do you encode/decode this message in python?

THANKS!