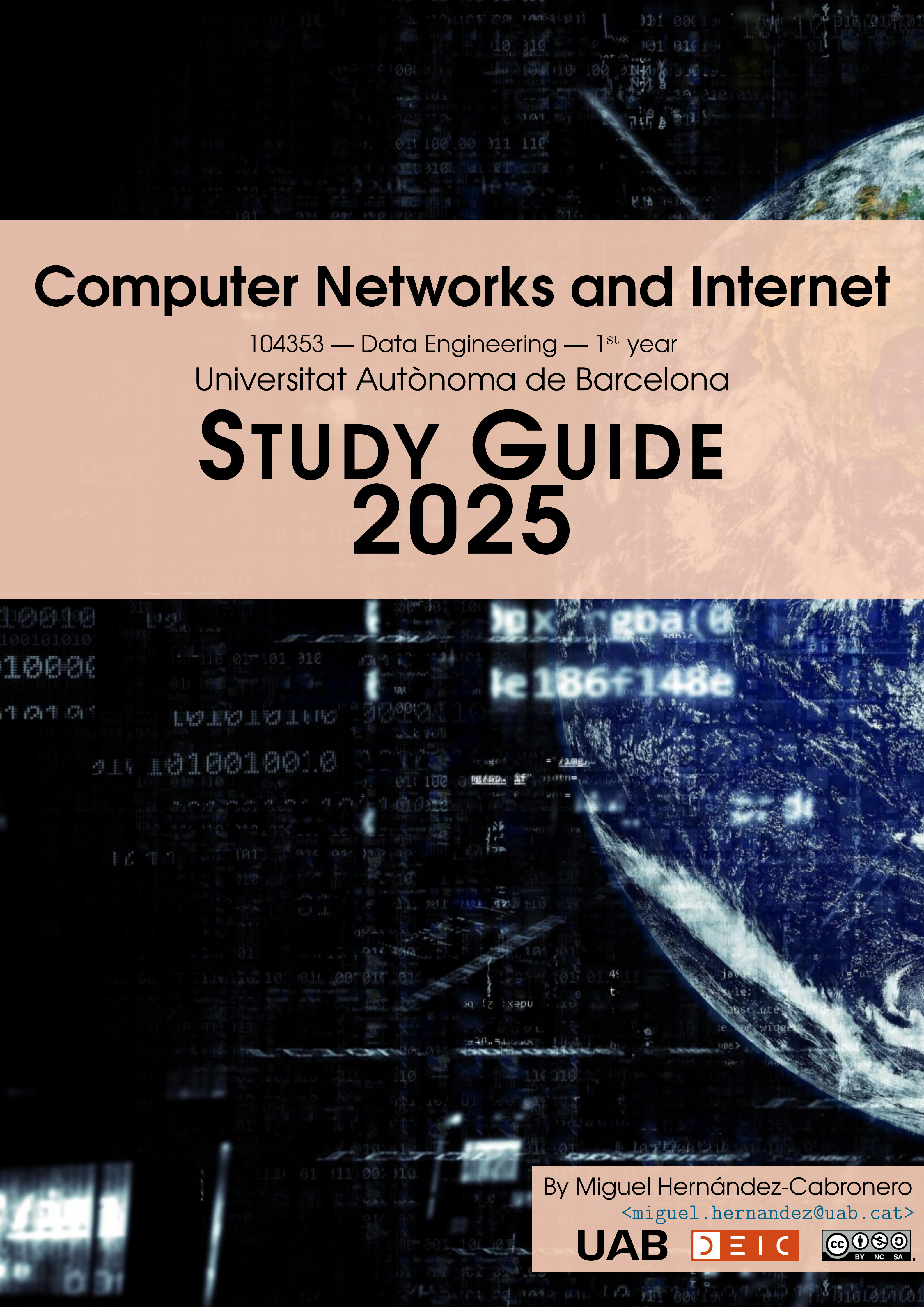


Computer Networks and Internet

104353 — Data Engineering — 1st year

Universitat Autònoma de Barcelona

STUDY GUIDE 2025

By Miguel Hernández-Cabronero
<miguel.hernandez@uab.cat>

UAB DEIC

License

Credits

Many of the visual contents were created by third-party artists and released under compatible licenses:

- Mathias Legrand and Vel: base LaTeX template
- TheDigitalArtist/pixabay: front cover image
- victorsteep/pixabay: Section 1 cover
- OpenClipart-Vectors/pixabay: Chapter Section 2 cover
- D-Kuru, Niridya, Joe Ravi, Shaddack / Wikimedia Commons (WC) : Ex. 2.6
- Timwether, Timewalk / WC : Ex. 2.7
- deepai : Iceberg Section 2.2
- Chetvorno : Antenna diagram Section 2.2
- Berserkerus : Modulation diagram Section 2.2
- Graham Rhind : Chapter 3 cover
- mrcolo : Chapter 4 cover
- Geek2003 Switch in Section 3.1.

Contents

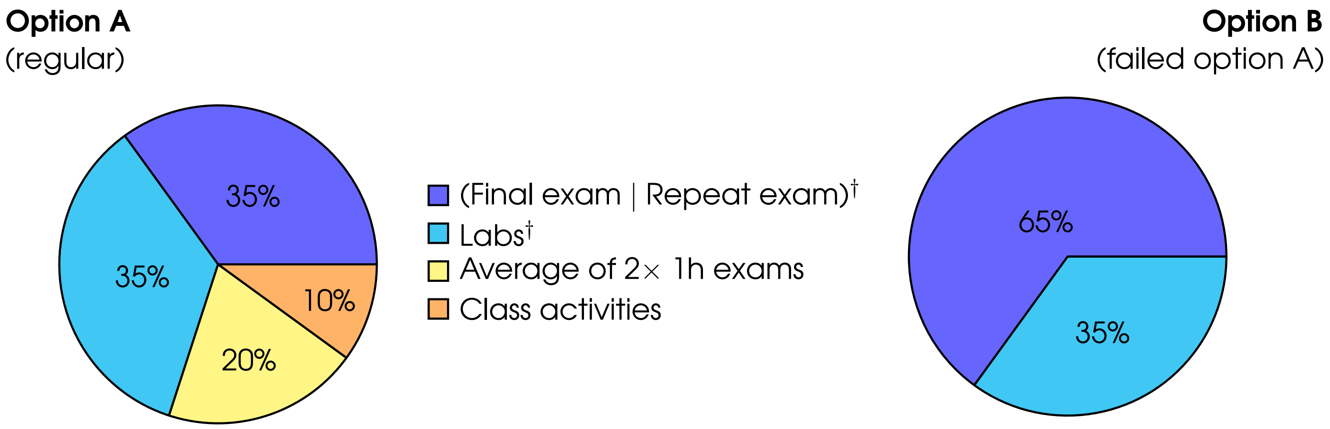
1. Computer Networks and Internet 2025

1.1 What you paid for

	Monday	Tuesday	Wednesday	Thursday	Friday
10:00					
10:30				**Problems** PAUL/**811**	**Labs** PLAB/**813**
11:00					
11:30				**Problems** PAUL/**812**	**Labs** PLAB/**814**
12:00					
12:30		**Lecture** Full class			
13:00					
13:30					
14:00					
14:30					
15:00					

13×	2h **Lectures**
12×	1h **Problem sessions**
12×	1h **Lab sessions**
Unlimited	**Office hour sessions** (*tutories*), individual or group: send email
90h	**Autonomous work** (total, expected)

1.2 Evaluation



†: A passing grade ($\geq$ 50%) is required in the part to pass the subject.

Dates:

- 1st 1h exam: during a Lecture session *early April*
- 2nd 1h exam: during a Lecture session *early May*
- Final 3h exam: Thursday June 5, 2025, 9h
- Repeat exam: Thursday June 26, 2025, 9h

1.3 This Guide

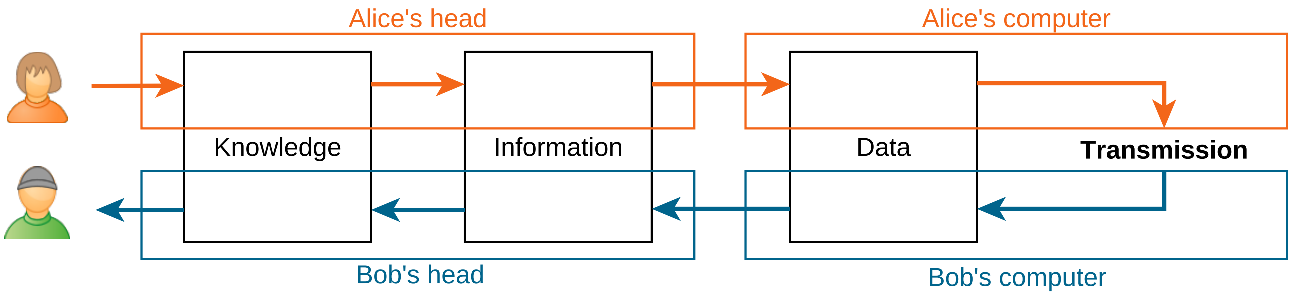
This guide is designed to help you progress through all parts of the "Computer Networks and Internet" course. Here you can find written materials, exercises and other tools to make the most of your effort and help you acquire valuable skills. It is strongly recommended that you become familiar with this guide and use it all throughout the semester, starting the very first week of class:

- During the lectures, we will use it to present and refer to new and previously visited concepts. We will also use it to drive activities and problem sessions.
- At home, it can help you gather and organize new knowledge, as well as to find useful exercises to practice your newly acquired skills.
- After you complete the course, it can be useful as an index of contents and pointers to useful reference materials.

> ♪ This guide is not a book, a reference material or even a complete set of class notes, and **it is not intended to substitute your own notes**. The guide's goal is rather about presenting you with new, interesting questions than about providing complete answers. This guide does not substitute continuous, active class attendance and autonomous work at home. Instead, you are encouraged to use the guide to help you understand the lectures, and vice-versa.

2. Piercing the analog-digital veil

We send and receive messages constantly in our personal, professional and political spheres. The almost-instantaneous availability of virtually any desired information is a key characteristic of the current era. When we use a computer of any kind to share or retrieve knowledge, first it must be transformed into data that can be understood both by humans and computers. This can be imagined as a sequence of steps:



Computers work with and transmit digital data, *i.e.*, sequences of binary 0 and 1 symbols. However, data transmission happens in the real world, where 0 and 1 are ideas and not tangible objects.

This chapter studies how those digital symbols *pierce the veil* from the digital to the analog (physical) world on transmission, and back from analog to digital on reception.

2.1 Information ↔ Data ↔ Binary messages

Knowledge, information, and data are not the same thing, although they are often used used interchangeably. In technical contexts, they should be used and interpreted accurately.

> **Exercise 2.1** Suppose we are outdoors and we want to tell our friend about the weather. We have knowledge about everything related to the weather, because we are there. For our communication, we need to focus on part of that knowledge, and decide what information to send. For instance, we could want to communicate the temperature, and say something like "the temperature is about 22ºC". There are many ways in which we can store that information as data inside a computer. Most likely, we will represent the number 22 as part of those data. **How would you define knowledge, information and data?**

Even though there are many types of data, *i.e.* numerical, textual, visual, scientific, *etc.*, these are eventually represented by numbers. For instance, one could use ASCII encoding to represent the string "Bobby" as the following sequence: $66, 111, 98, 98, 121$.

Inside a computer, those numbers are stored in binary registers, *e.g.* in the CPU and in the RAM. However, there are multiple ways of representing numerical data in binary format. More specifically, we need to pay close attention to at least the following aspects whenever reading or writing data:

- integer or with decimals?
- bitdepth? (*e.g.*, 8 bits per number)
- signed or unsigned?
- For multibyte numbers: big endian or little endian?
- For numbers with decimals: floating-point or fixed-point?

Exercise 2.2 Explain **how to complete the second row given the first** (and vice-versa), and what assumptions you need to make in each case.

Number	7	2	-3	0.75	260
Binary	00111	00000010	1101	1100	0000 0100 0000 0001

How about à ↔ 1100 0011 1010 0000?

Sometimes we need to inspect the contents of a binary register (*e.g.*, for debugging or other analysis purposes) or to set those contents manually. For humans, it is inconvenient and *very* prone to error to handle binary strings longer than a few bits. When we need to inspect or modify the contents of a binary register (*e.g.*, for debugging purposes), we often use base 16, *i.e.*, hexadecimal notation.

In hexadecimal, there are exactly $16 = 2^4$ different digits (0 to 9, a to f) with decimal values from 0 to 15, each of which represents exactly 4 bits. When we writing hexadecimal values to a computer, *i.e.*, in source code, hexadecimal values are typically preceded by `0x`, *e.g.*, `0x58a5b0`. Similarly, binary expressions are preceded by `0b`, *e.g.*, `0b0011`.

- Spaces and even line breaks between digits do not carry any meaning, they are only used to facilitate visual inspection.
- In some texts, when multiple bases are applicable in a context, they are shown as subscripts as in $\texttt{58a5b0}_{16}$ and $\texttt{0011}_{2}$.

Exercise 2.3 Consider the following message, which is composed of a concatenation of 3-bit unsigned integers: `a4 3f 20`.

- How many bits and bytes long is it?
- **What are the first five integers** contained in the message?

Most often, we will let our code handle data manipulation. To do that, we need bitwise and boolean logic operations such as the following

Operation **Python**	bitwise AND `&`	logic AND `and`	bitwise OR `|`	logic OR `or`	left shift `<<`	right shift `>>`

Other useful python tools are the `bin()` and `hex()` functions, that convert an integer into its binary and hexadecimal representation, respectively, and `int()`, which can parse a string describing a number and return that number. We can also control the format in which we show numbers, *e.g.*, `print(f'{n:08b}')` would print the value of variable `n` in binary form, using at least 8 positions and filling the empty leading positions with zeros (more in the python docs).

Exercise 2.4 Consider the code shown next. The numbers printed when run are of special importance in networking and computer science. **What do they have in common?** (Hint: You may want to modify the code to show their binary representations).

snippets/bitwisemanipulation.py

```python
a = 0xff
print(a)
for _ in range(8):
    a = (a << 1) & 0xff
    print(a)
```

For code listings like the one above, you can download the source code and its output.

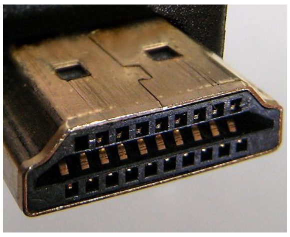
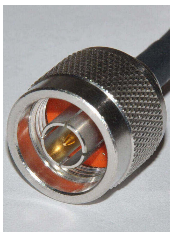
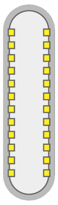
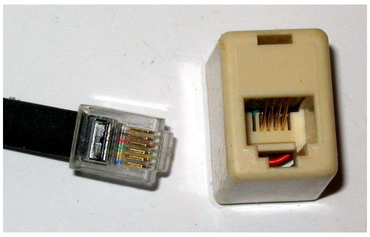
2.2 Analog ↔ Digital

Once we decide what bits to transmit, we need to find a way of making those bits available to another machine at a distance. First, we need to decide what medium to use and what physical properties we can control and monitor. Popular choices include using voltage on copper wires, sending light over optical fiber and manipulating the electromagnetic field using antennas.

> **Exercise 2.5** **Ponder**:
>
> - Can we make a copper cable as long as we desire?
> - What medium do wifi connexions use?
> - Do we need optical fibers to perform light-based communication?

If we opt for copper wires, we can put several in parallel and send more data at the same clock speed. This, however, comes at the price of additional complexity and cost than serial designs. Voltage interferences in the data lines may happen for a number of reasons, including physical phenomena such as electromagnetic induction in the wire. Notwithstanding, cables compliant with modern data transmission standards (*e.g.*, IEEE 802.3) often employ twisted pairs of cables and shielding among other strategies to guarantee bit errors below 1 in 10^{12} bits.

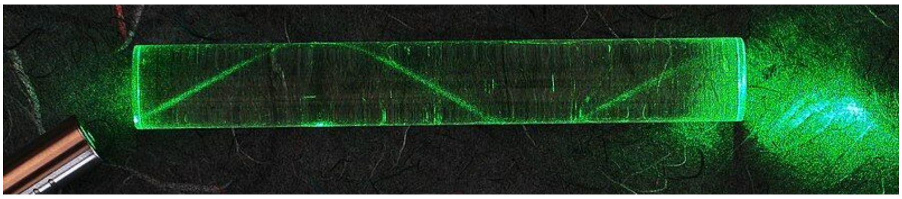
> **Exercise 2.6** **Consider and identify** the following wire connectors:

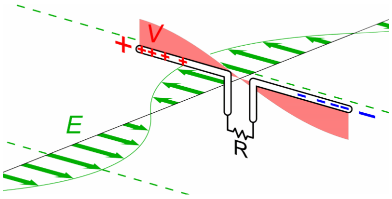





Have you ever been diving in a swimming pool, looked up and the water was mirror-like? Optical fiber cables work of this phenomenon, called refraction. Light that enters the fiber through one end stays inside until it exits through the other end. The coating of the fiber is important to support its integrity, but that coating does not participate in the "bouncing" of light when advancing through the fiber.

In a single-mode optical fiber, the sensor at the receiving end detects the presence or absence of a single wavelength range. These cables are relatively thin and cheap, but carry a single signal. Another option is to transmit multiple wavelengths (*e.g.*, using several laser sources) at the same time, through the same fiber. At the receiving end, a prism is used to divide the beam of light back into its monochrome components, which are sensed separately. In this way, multi-modal optical fiber optical fiber allows multiplexing several signals, greatly increasing the effective transmission rate.

> **Exercise 2.7** When a beam of light enters an optical fiber, it does so with a certain angle of attack. This angle affects the time it takes to reach the other end. If we project a cone of light, light enters the fiber at multiple angles of attack. What happens at the other end if we **turn this cone of light on and off** very quickly? Is it relevant whether we use mono-modal or multi-modal beams?



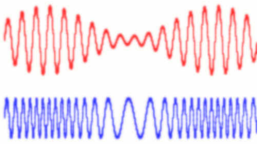


Even though there are many classes of antennas, their main purpose is to detect electromagnetic fields and/or to create them. The Maxwell-Faraday equation tells us that *changes* in the electric field $\vec{E}$ create a perpendicular magnetic field $\vec{B}$, and the other way around.

A dipole antenna like the one to the right uses this concept by applying an alternating voltage V, which generates electromagnetic waves that propagate outwards the dipole axis.

When we emit some energy E from a point in all directions, that energy is distributed around a growing sphere. Since the area of a sphere of radius r is $4\pi r^2$, the amount of energy that reaches a point at distance r will be in the ballpark of E/r^2.

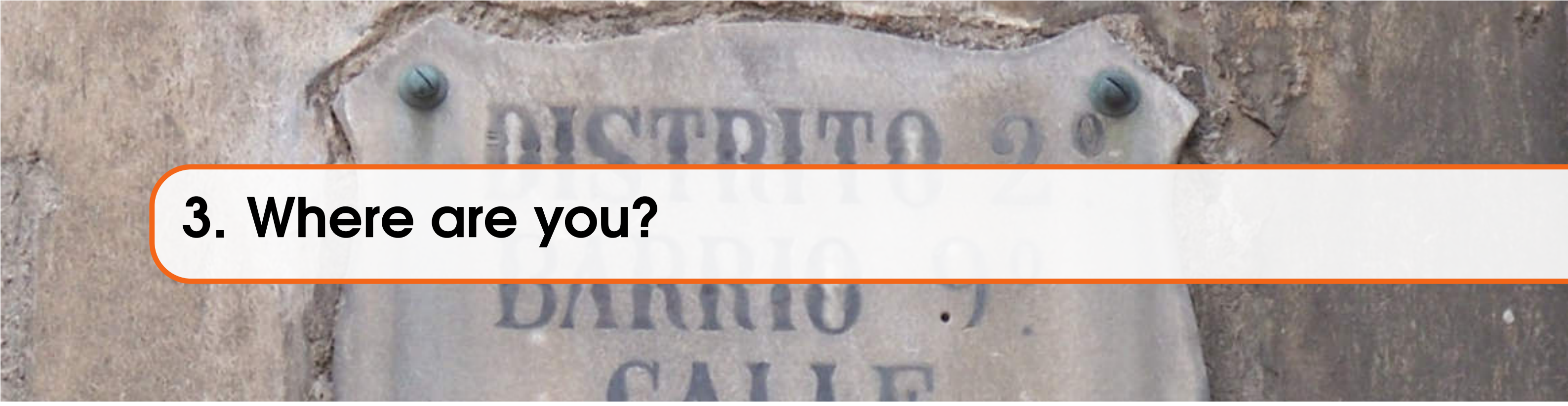
This is sometimes referred to as the inverse square law, and is the reason why currents inducted in the receiver's antenna are usually between nanovols (10^{-9} V) and picovolts (10^{-12} V). Surprisingly, this is enough for modern detectors to read the data signal.



Regardless of the method chosen to let the data travel, the veil between analog signals and digital data must still be pierced. Once way of doing this is modulation: a base sinusoidal signal called carrier is produced, and the data are encoded by modifying this carrier. For instance, once can change the amplitude of the carrier (ASK), its frequency (FSK), or even the amplitude and the phase at the same time (QAM).

Exercise 2.8 The figure above displays an example of amplitude modulation and of frequency modulation. Which is which? For each case: is that **modulation transmitting an analog or a digital signal**?

Exercise 2.9 What's the **difference between bandwidth and transmission speed**? Why do you think they are interchangeably used in non-technical contexts?



3. Where are you?

Connecting two computers makes them much more useful than two isolated machines. However, their true potential is only unlocked when they can be connected to *lots* of other computers.

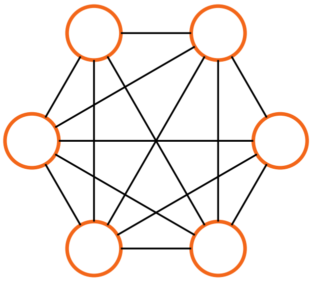
Practical limitations including cost and geographic location make it impossible to directly connect every pair of computers. Instead, devices are distributed across a graph of interconnected networks, which may extend even beyond the scale of a planet.

Whether locally within physical reach or in another continent, an addressing system is needed to identify, locate and route messages so that they reach their intended peers, just like we do with physical locations.

3.1 Please contact me

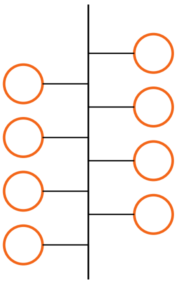
When only 2 devices are involved, one may use a point-to-point connection. In this type of connection, there is only "this side" and "the other side" of the cable. Moreover, when a device wants to communicate with the other end, there is only one link (*e.g.*, one cable) to choose from, so there is no possible confusion. Since we can operate this connection with just 2 network cards and 1 cable, point-to-point connections are viable for $N = 2$ devices.

If we want to connect $N > 2$ devices, a naïve option is to connect every device with each other in a *complete graph*. Now every device needs to handle $N - 1$ cables to the other devices, but communicating with any other device is as easy as choosing the correct cable and establishing a point-to-point connection.



The main issue with this approach is that the total number of connections is $N \cdot (N-1)/2 = O(N^2)$ (more on that in your trusted *Discrete Math* course). This means that, in order to have $N = 10$ devices connected in this way, you would need for instance 40 cables *and* 90 network cards. For $N = 1000$ devices, there would be half a million point-to-point connections, which is already quite unmanageable.

> $O(N^2)$ is notation for a quadratic complexity bound. This is *not* the same as exponential and should never be confused. (read more...)



There is one trick up our sleeves: data buses, where one or more data lines that are simultaneously connected to multiple devices. This retains the simplicity of point-to-point connections, because each device needs to handle only 1 cable and can use it to communicate with all other devices. The cost-effectiveness is also retained, since only N network cards for N devices ($O(N)$).

The main advantage of data buses is also their main weakness: all devices send and receive data using the same data line, but they cannot do it at the same time because there is only one line. If two or more do, there is a collision that makes data transmission impossible while it lasts.

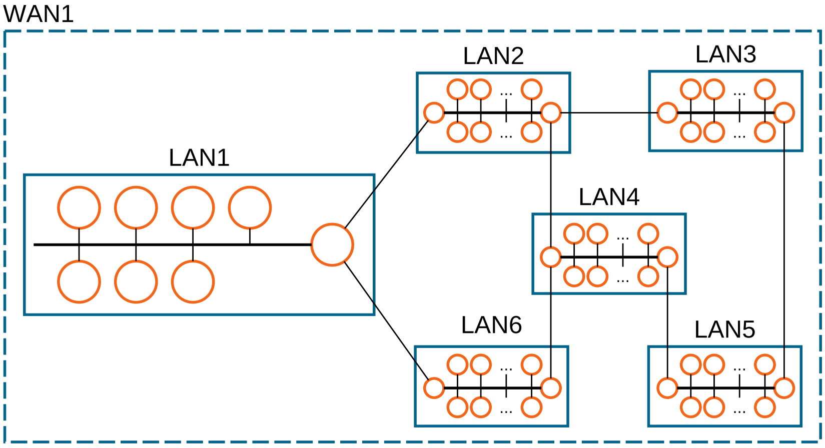
> As computer networking was developed, people experimented with many alternatives to the bus topology. One curious example is Token Ring, where devices are

> connected around a circle, and messages are passed around in a single direction.

In practice, it is not trivial to implement a bus technology where new devices can be added over time. Historically, this was sometimes done by physically piercing the main bus cable, and adding a new cable towards the new computer being connected. Later on, this was simplified with devices called hubs. They worked similarly, but they offered pre-built ports and removable connectors. Nowadays, hubs have been replaced by switches. Switches offer the functionality of a bus, connecting every device with each other, and also prevent collisions. For obvious reasons, structuring connections around a switch is called a star topology.

Exercise 3.1 What is **needed in a switch** that is not needed in a hub?

Bus topologies work great at local scales, and are extensively employed in local area networks (LANs). However, it is not cost-effective, (sometimes not even physically possible) when devices are far apart. Instead, devices can be organized in separate LANs, each one conceptually a bus, and then these separate LANs can be connected using cheaper, feasible point-to-point connections. Considered together, these devices would then form a wide-area network (WAN).



3.2 I need to find you

Once again, the main advantage of data buses is also their main weakness: all devices send and receive data using the same data line. Even if collisions don't happen, every message sent into the bus is received by all devices. However, what we actually want is for our message to reach its destination, and only its destination. Let's call this the *find-my-device problem*.

The find-my-device problem is not unique of buses or LANs. On the contrary, it becomes very important when we consider WANs, where not all devices are directly interconnected. In this case, we need to make sure we can route our messages between different LANs, and that they reach their destination.

Surprisingly, not even point-to-point connections are safe from the find-my-device problem. Assuming devices A and B are connected that way, two different programs may want to exchange information at the same time, *e.g.*, some alarm control software and a music streaming app. In this scenario, you want the client and server of the alarm monitoring service to communicate with each other but not with the music streaming service, and vice-versa.

The most common solution to this problem is to use identifiers that let us differentiate between devices or even between running processes inside an operating system (OS). Some of these identifiers are referred to as addresses, precisely because they are used to find and reach a remote device.

Addresses are typically *numerical* identifiers, that is, just a number. As such, they can be expressed in different bases, including decimal and hexadecimal. These numbers are drawn from a predefined set, and the choice of that set determines the maximum number of elements we can uniquely identify.

> Addresses expressed as `d8:43:ae:61:ed:f1` or `192.168.1.1` (as it will be seen later in the course) are also numbers we could have expressed as `237785200061937` and `3232235777`, respectively (which is not as convenient).

Exercise 3.2 **How many elements can be uniquely identify** (at most) with an address we express using 5 bits? How about 10? How about 20? Express the general solution mathematically.

Exercise 3.3 Suppose there are 1000 machines in a LAN. **How many address bits are needed**? How about 800 machines? And 1100? Express the general solution mathematically.

> In 2019, we run out of Internet (v4) addresses. Whoops!

3.3 How do I get there?

In many scenarios, multiple addresses are needed to perform the desired end-to-end communication. For instance, we may need to identify a single device within a LAN, and that LAN within a WAN. Sometimes, we will need to identify a process (a running program) within that device. This is not unlike someone paying you a visit in person:

Joseph Edgar Foreman
123 South Fake St.
Adams County
Ohio

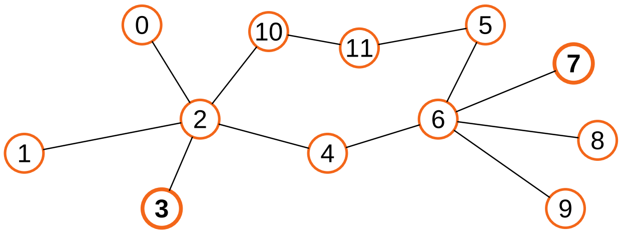
Process #1717403
Device #51688798
LAN 3141 (Data Engineering Labs)
WAN 442 (UAB)

Whatever addressing system we use, it must contain enough information to locate and reach the destination, *i.e.*, to route our messages through the graph of connections. The process of routing involves multiple steps or "jumps" across networks until the destination LAN is reached.

The name router refers to a type of network device whose job is to forward these messages across networks. These devices must contain enough information so that, when handed a message with a destination address, they can decide what network interface to use. In turn, non-router devices only need to be configured so that they can reach the next router.

> At home, your router typically also plays the role of a switch, but those are different concepts.

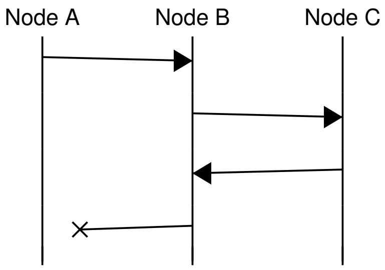
Exercise 3.4 This figure represents a handful of devices and their physical connections. These devices can communicate only by sending messages through those connections. Notwithstanding, ③ wants to send a message to ⑦.



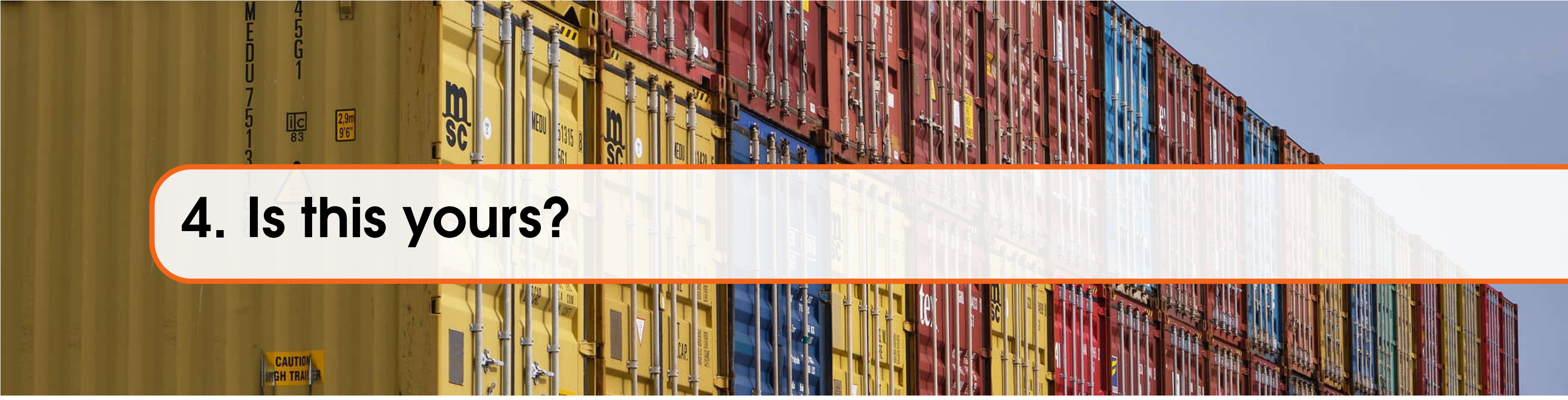
- Can you identify any devices likely to be acting as a switch? And as a router?

- What's the **minimum number of messages** sent so that ③ can contact ⑦, and ⑦ can reply to ③? List them in the order they are produced.
- What information needs to be included in those messages so that the communication ③ ↔ ⑦ may take place?

Swimlane plots like the one below are used to represent the interactions (*e.g.*, message exchange) between actors (*e.g.*, network devices). In them, the vertical axis represents time, which is invaluable to identify cause-effect relations.



Exercise 3.5 Based on your answer to Ex.3.4, **produce a swimlane diagram** that represents the message exchanges. For each message (*e.g.*, above the arrows), include the addressing information present in it.



4. Is this yours?

Information travels through this graph of networks in the form of packets, which contain a small amount of information and meta-information.

Their format must be agreed upon by both ends of the communication, so that those packets can be automatically produced and interpreted.

When packets travel through one or more networks, they can get lost or reordered. If a continuous stream of data must flow between two computers, packets must contain mechanisms to detect losses and restore the proper order.

4.1 I have a delivery for you

It is not feasible to establish physical connections between each pair of computers that want to communicate. Instead, relatively few connections are shared to carry messages between many pairs of devices.

If a single device transmits data continuously for a long time, that connection is blocked and may become a bottleneck that prevents other devices to communicate. To avoid this problem, connections limit the maximum amount of data that can be sent without interruption. As a result, devices are forced to send data in discrete bursts called packets. The exchange of these type of messages across one or more networks is called packet switching.

> The concept of packet switching (crucial to the creation of the Internet) was developed in the 1960's (during the cold war) by the US military to have a communications network that would work in case of a nuclear strike.
>
> The problem with the one they had was that, if one station was damaged, the whole system stopped working. However, by using packet switching, any connected subgraph of the network would keep working.
>
> The reasoning for creating a packet-switched network was that they could still retaliate their enemy in case of an attack, which would make them a less attractive target.
>
> Leonard Kleinrock, one of the pioneers of internetworking, disputes the authorship of packet switching, which is often attributed to Paul Baran and Donald Davies.

Packets need to be small so that connections are not blocked for too long. At the same time, we want packets to contain as much data as possible, because each packets contain overhead data (such as the addresses discussed in Section 3) that need to be sent in addition to the user's payload data. Networks set the maximum packet length using a parameter called Maximum Transmission Unit (MTU), *e.g.*, 1500 bytes in Ethernet.

Packet transmission across a data link must be done carefully, particularly when that link is a bus shared by multiple devices. Each packet must be individually distinguishable from the rest, which can be done via at least three strategies:

1. *Fixed length*: the length of all packets is the same. The sender and the receiver must have agreeded to this value in advance.
2. *Explicit length*: the length of the packet is included in the packet, *e.g.*, using the first byte of the packet. The sender and the receiver must agree on how this information is included and how to interpret it.
3. *Escape sequences*: Certain binary escape sequences within the data have a special meaning. For instance, one could define the byte `11110000` (`0xf0`) to mean "end of

packet": these bits must appear after each packet, and only then. Then the sender could start transmitting the contents of a packet, followed by `0xf0`. Both parties must agree on what escape sequences to use, what they mean, and how to send data equal to those sequences (*e.g.*, it should be possible to send `0xf0f0f0` without triggering an "end of packet" until all bytes are transmitted).

> It is also possible to to signal the *beginning* of a packet. In that case, a preceding sequence called preamble is used.

Exercise 4.1 All strategies described above are used nowadays in real scenarios, depending on the features, costs and trade-offs of each one.

Discuss **which of these strategies would be most appropriate** for each of these scenarios:

- A single manufacturer designs the hardware and software that communicates two endpoints. A single data line must be multiplexed for multiple control commands as well as multiple data streams. The priority is efficient power and buffer usage.
- Multiple devices share a bus. They occasionally send messages of different lengths, but not a huge volume of data. The priority is cost.
- Multiple devices share a bus. They continuously send messages of different lengths, trying to maximize the effective transmission rate through the channel. The priority is throughput.

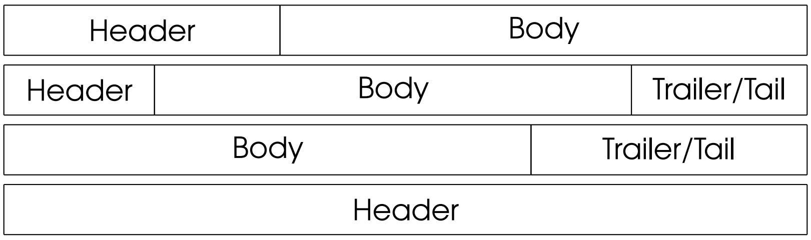
4.2 Please fill in the form

Regardless of the strategy employed, data packets typically comprise two parts:

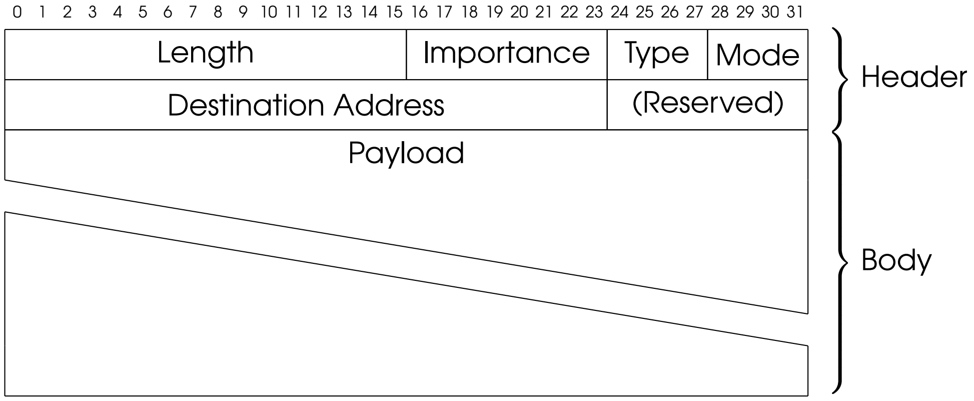
- Data payload: the information that needs to be transmitted, and
- Metadata: the meta-information needed to deliver the payload.

Length might be one of the metadata fields included in the packet, along with others that help identify their type and function. The location of payload and metadata, as well as the exact metadata fields included, their length and their meaning must be known by both ends of the communication. All these decisions constitute the packet format or packet structure.

One of the most common ways of arranging payload and metadata is with a header-body format. In this case, all meta-information is included first, followed by the data. Other formats may include a trailer (also known as tail, most often used for error detection and correction), in combination with the header, or replacing it. All of the following configurations are possible.



The format of the payload in a packet is normally user-defined. The format of the header, however, is strictly defined so that it can be easily produced and parsed. This format is often presented using a diagram that helps identify the individual bit and byte positions, like in the following (not-real) example:



Fields may also have fixed or variable lengths, which do not necessarily align with byte boundaries. Fields may also have length equal to 1 bit, in which case its is normally called a flag or flag bit.d

In diagrams like the one above, multiple lines (rows) are often used so that they can be easily printed and read. In that case, the meaning is the same as if all fields had been presented along the same line (row).

Exercise 4.2 The following packet contents were sniffed out of a network, expressed in hexadecimal. Assuming the packet has the format of the example above:

- **Indicate the value of each field** (length, importance, type, mode, destination address, and payload.
- Can you guess the meaning of the payload?

```
00 0c 00 f3 bb bb bb 00 68 65 6c 70
```

Exercise 4.3 The following code accepts two arguments: (1, `sys.argv[1]`) the path of an input file, and (2, `sys.argv[2]`) the path of an output file. The first 255 bytes of the contents of the input file are read, they are formatted as a packet, and the bytes of that packet are output to the output file.

- Describe the packet format used in the code, and its limitations.
- Extend the packet format so that
  - Packets can be longer than 255 bytes
  - The address of the destination device can be encoded
- Provide a byte diagram of the new format, as well as an explanation of the addressing system you are using.
- Modify the code so that it outputs packets of your proposed format.

snippets/simplepacketoutput.py

```python
import sys

# By default, read this script and write to the standard output
if len(sys.argv) != 3:
    sys.argv = [sys.argv[0], __file__, "/dev/stdout"]

with open(sys.argv[1], "rb") as input_file, \
        open(sys.argv[2], "wb") as output_file:
    # Read at most 255 bytes from the file
    payload: bytes = input_file.read(255)
    # The + operation concatenates bytes
    output_file.write(bytes(len(payload)) + payload)
```

Languages like Python can make a distinction between files containing text and files containing binary data. In the code above, files are open in binary mode using

> modes `'rb'` and `'wb'`.
>
> When open in binary mode, file bytes are directly represented by a `bytes` object, an array of integers between 0 and 255. In text mode, those bytes are processed (decoded) further by the `open` method, and returns a string that might contain special characters like accents or non-latin glyphs. (read more...)

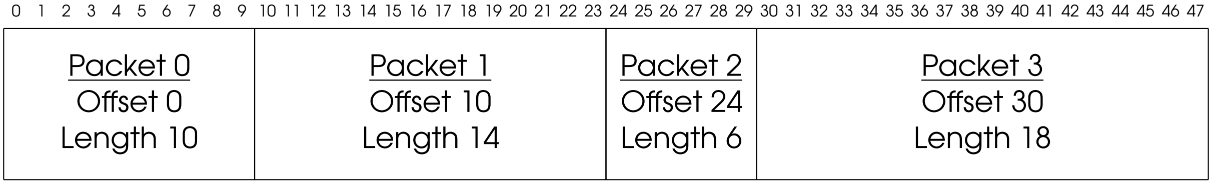
4.3 They keep coming

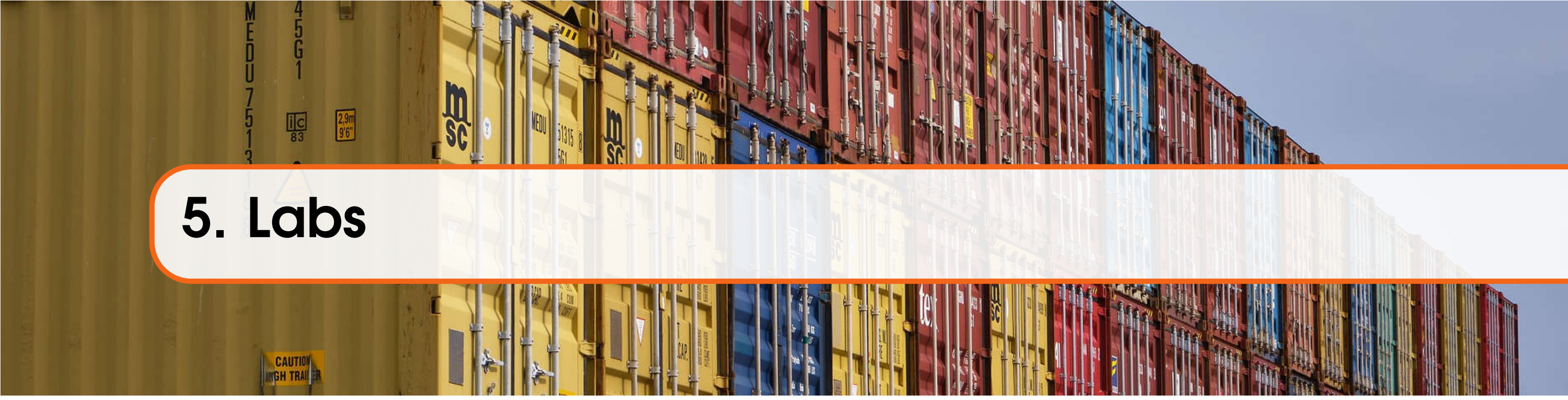
When developing applications that use networking capabilities, it is often useful to imagine communication as a stream, *i.e.*, as a conceptual pipe where we put our data (any amount of data) in one end, and it comes out at the other end. If we have this capability, then it becomes much easier to send files of any size, and to transmit a never-ending amount of audio or video. Unfortunately, all we have to simulate those streams are packets.

Packets often need to be forwarded through multiple networks. Routers are not perfect and are sometimes inoperative or improperly configured, so not all packets arrive to their destination. Moreover, different packets may be delivered through different routes that may be of different length, so packets may arrive out of order.

If we want to simulate a stream of data, packets must contain enough meta-information in them so that losses can be detected and the correct order can be restored. This introduces an overhead that is not suitable in all scenarios –*e.g.*, very low latency–, so some applications base their network communication in messages instead of streams.

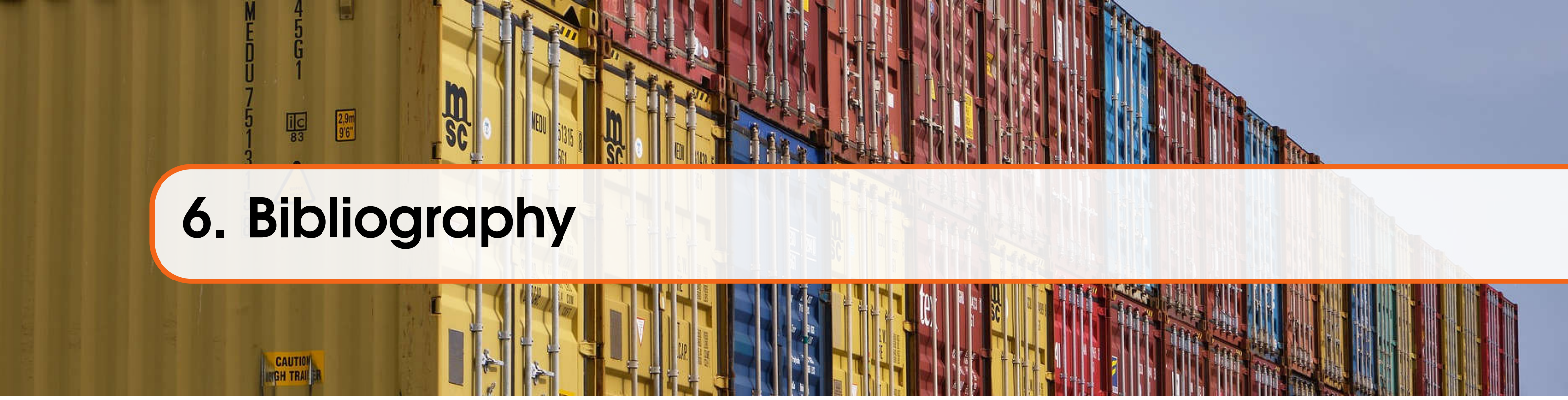
When streams are required, the concept of offset is used to reassemble multiple messages into a single stream. Each packet contains some bytes of the data stream, and the offset indicates the exact location in that stream.

> **Exercise 4.4** The following diagram describes a stream of 48 bytes produced by a source node, as well as the packets it was split into before transmission:
>
> 
>
>
> - Describe the minimum header that can be used to transmit this stream.
> - Assuming that the recipient receives Packet 2 first, followed by Packet 0 and other packets are lost: **what bytes of the stream are known** by the destination node, and what bytes are unknown?
> - How can the source node know that some packets were not delivered?
> - What would happen if a bogus message is received by the destination, with offset 7 bytes and length 4 bytes?



5. Labs

1. Platform setup, bit-byte with python, basis for submask calculator (extend snippets/bitwisemanipulation.py?)



6. Bibliography

Makefile does not compile this because there is nothing to do

Books

Articles

7. Index of Concepts