Introducion to the course

COMPUTER NETWORKS A.Y. 24/25

Leonardo Maccari, DAIS: Ca' Foscari University of Venice,
leonardo.maccari@unive.it

Venice, fall 2024

Sect. 1 Introduction to the Course

Welcome to the Course

- My name is Leonardo Maccari, I am an associate professor at Ca'Foscari and I am your professor for this course.
- Computer engineer (ing. informatico) I have been working on networks and network security for the past 15 years.
- How to reach me for tutoring: I generally make tutoring before the lesson on Wednesday (09:15-10:15), but you **must** write to two days before to request tutoring. We can also agree on some other time.

. . . in the previous episodes. . .

- This course does not requires previous theoretical knowledge
- It does require practice with code
- I will show you snippets of source code that implement the concepts we go through, and ask you to make some exercises.
- This requires basic knowledge of the C language and possibly Python language

Schedule

- Tuesday: 12:15-13:45 (Delta 0A)
- Wednesday: 12:15-13:45 (Delta 2A)
- During the lessons we are going to make exercises following the book material
- I strongly suggest you bring your laptop, because some of the exercises can be accessed using a web platform (more on this later on)

Evaluation: part 1

- A written exam, split in two parts:
  - multiple choice questions: 15 points
  - open questions: 15 points.
- This will be made on Moodle, in lab.
- If you don't achieve at least 3 points in the first test, you can not access the second.
- The written exam will give you a grade 18-30 and you can register directly

Evaluation: part 2

- An oral exam.
- Those that achieve more or equal to 25 at the written test, can make an oral to improve their grade.
- At the oral you can take a positive or negative grade, and if you do really bad, you can fail too.
- So don't come to the oral test "just because". The oral is intended for those that believe that they studied enough to improve on 25.

Material

- I have selected two books for this course:
  - O. Bonaventure, “Computer Networking : Principles, Protocols and Practice” third edition (CNP3). This is a totally different book, it is open source (you can download it for free), it is interactive (there are exercises you can make on the web page) and it uses a different concept than the classical Tanenbaum. This book is the one I have built my classes on.
  - A.Tanenbaum, D. Wetherall, “Reti di Calcolatori”, 6a ed., Pearson. This is one of the most popular and used books worldwide for teaching networks. It is very good and you can find it in the library and you can find the Italian version too.

A bit more about the book

- The book can be downloaded in pdf or ebook from `https://www.computer-networking.info/`
- It was written mostly by Oliver Bonaventure, a very well known professor in the networking area
- The book is released with Creative Commons license, which means it can be shared, modified, improved.

A bit more about the book, and the course

- The first part describes the theoretical foundations of this domain as well as the main algorithms involved.
- What are the problems that you need to solve when you deal with communications? what are the algorithm you will use?
- The books touches topics that are very general and can be applied to any communication system, not necessarily the Internet
- This will be mirrored in the first lessons of the course.

A bit more about the book, and the course

- The second part contains a detailed explanation of how these problems were solved in the design of the Internet.
- This means studying the main *Internet protocols* including HTTP, DNS, TLS, TCP, UDP, IPv6, BGP... and standards like Ethernet and WiFi.
- These are the building blocks of the technology on which the Internet is made of.

Book Exercises

- The third part of the book is all about exercises
- Some of them are open questions, some multiple choice, some are interactive.
- We will review some of them during the lessons and you can make them on your own to consolidate the knowledge and skills you gain during the classes.
- To complete the interactive exercises you have to sign-in to the course, **and** to the ingenious platform.
- In ingenious just choose the “Reti di Calcolatori - University of Venice (23/24)” course
- To enroll you need a password: **Eijae1ai**
- There is nothing difficult about it, so just register.

Book Status

- Compared to other books this is a pretty *young* one, and it uses an original approach, imagined for CS students
- It can be accessed for free.
- It is an ongoing effort, with room for improvement, as it is a community-based effort.
- I strongly suggest you do try to improve it: if you find a typo or a sentence you think it can improve, open a github issue to the authors and they will evaluate it. If you don't feel sure about it, discuss it with me beforehand.

Feedback

- This is the second time I hold this course
- Compared to the previous years I have changed all the material and the approach
- I hope you (and also I) will enjoy the course
- . . . but there is always room for improving, so feel free to send constructive feedback.

Practical Steps

- Register to the ingenious platform: `https://inginious.org/`
- Search among the available courses for: *Reti di Calcolatori - University of Venice*
- You should see a long list of exercises, we will use some of them during the course.

License

- Whenever the slides carries an explicit copyright notice, you should consider it.
- For instance if slides have an explicit Creative Commons license, then you can re-use the material as the license says
- For material I create from scratch, I use CC licenses that allow you to at least share the material with others
- For material I re-use, then the license of the original work may apply (for instance, material I take from the CNP3 book)
- When there is no license, then I probably re-used some material non-CC. I can do it during lessons because I am a professor in his teaching role, but you can't. So don't re-share it.

Sect. 2 Part II: introduction to the Internet

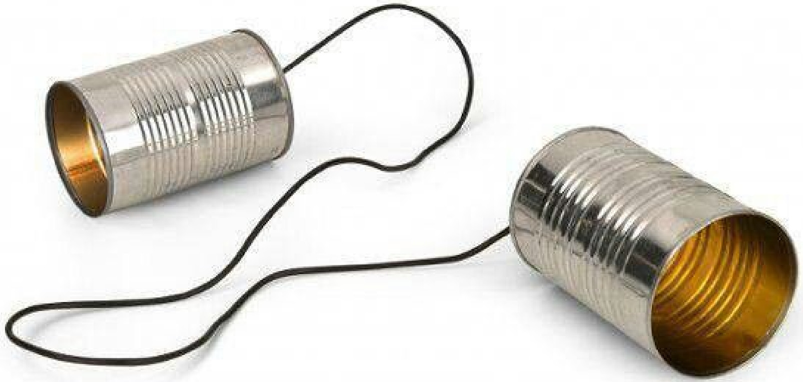
The rest of the lesson

- Today we will make a brief and informal introduction to the Internet
- Only the last part will be part of the exam program, the rest is informative but helps you understand how we arrived here.

Before the Internet we Know

- Traditionally, communication networks were *circuit-switched*.
- There was a physical circuit (i.e. a cable) that was connecting point A with point B in space.
- Electromagnetic signals were physically traveling from A to B, that is, an electron leaves A and the same electron reaches B (well, oversimplified. . .)

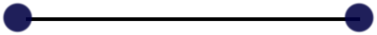
An analog circuit-switched network equivalent

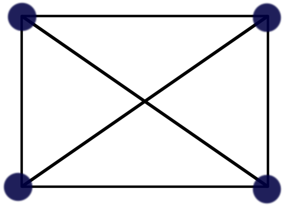


The problem of Multiplexing

- Many people, means many circuits, but. . .
- A circuit can not be physically present between any couple of telephones: if you have 1000 telephones, each telephone needs 999 cables. . .
- The need for switching emerged.
- A switch is a device that aggregates lines. Switches can be composed to make hierarchical networks.

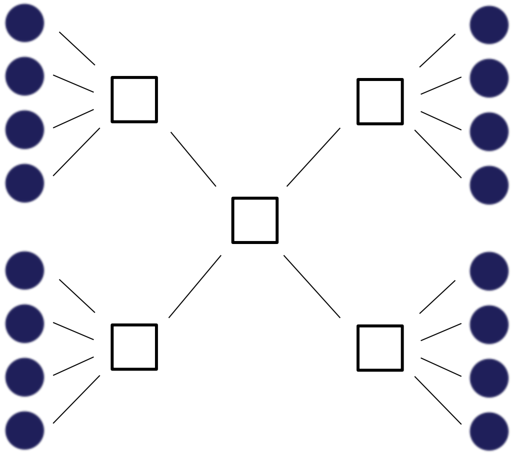
2 endpoints: 1 cable





N endpoints: N*(N-1)/2 cables

Hierarchical Networks



Madam, would you connect me to. . .

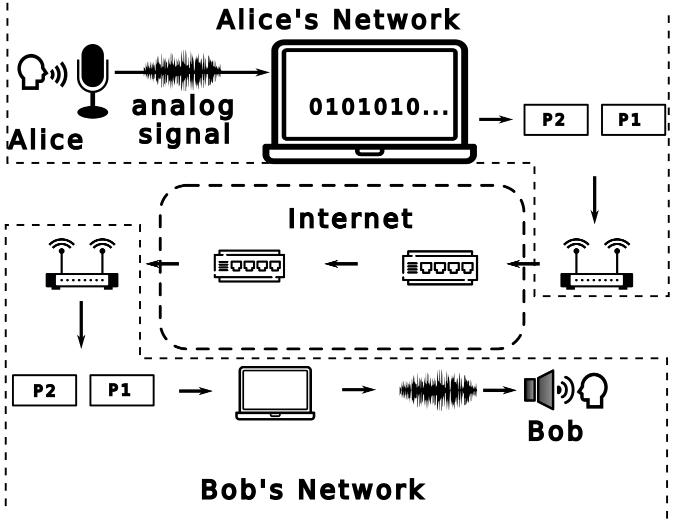
Hierarchical Networks

- When A calls B, a series of circuits need to be connected by a series of operators.
- To achieve this, a hierarchy is needed. A calls his local operator, which calls the regional operator, which calls the national operator, then down local towards B.
- Even when human operators were replaced by machines, the hierarchical architecture was preserved.
- A hierarchical architecture (like a tree-shaped network) is intrinsically fragile to failures, if the central node in the previous topology fails, the network is disconnected
- In the 60s, telecommunication engineers started to design a different kind of network.

Packet Switches

- The idea of packet switched networks emerged.
- In a packet-switched network there is no stable physical circuit (not even temporary for the duration of the call) between A and B.
- Every information is encoded in digital format (0/1 bits) and included in a "packet"
- A packet is just a standard format to organize bits.

Packet-Switched Network



Packet Switched Networks

- Voice is translated into a digital stream of bits with a microphone and a sound card
- Bits are grouped into **packets** by the OS of the PC
- Each packet is delivered to Alice's router
- Router in the Internet forwards each packet towards Bob's network
- Packets reach Bob's router, then Bob's PC, they are reordered, a digital stream is reformed and they are sent to a speaker.
- The speaker converts the digital stream to an analog sound.

Packet-Switched Networks: Advantages

- Each link could use a different technology, and in fact they do: you use wireless in your home, copper from your router and optic fiber between Internet routers. The network grows easily because there is no “legacy” to be preserved. With a circuit-based network instead, the radio technology must be compatible throughout the path from A to B.
- Once data is digitalized, it could carry any kind of information: voice, video, data. . . A circuit-based network brings analog information, Packet-Switched networks are more versatile.
- Routing is dynamic. If there is a failure, routers will find a way to “route around” the failure and find a valid path to the destination (if there is one). Packet-Switched networks are more robust and self-healing.

Internet is Decentralized

- The consequence is that the Internet is more robust and does not assume the existence of a global coordinator
  - It can extend without a planned hierarchy
  - If something breaks, it auto-repairs
  - Thus, it is way more resilient than old-style hierarchical analog circuit-switched networks.

Inter-operation

- So we have a network that is made of different technologies, without a global coordination (every router takes its own decision) and with no hierarchy.
- How can this actually work? How can hardware and software vendors produce products that are somehow magically inter-operable?
- Thanks to the use of **layering** plus **standards and protocols**.

Layers, Standards & Protocols

Layers

- The sub-system that implements networking in any computer is split in **layers**.
- Each layer takes care of a specific task. All together the layers make the **network stack**.

Standards and Protocols

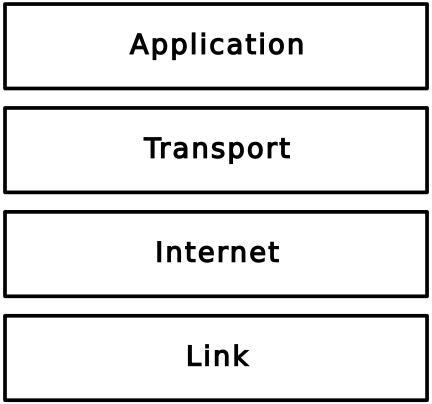
For each layer there is one (or more than one) standard and/or protocol that every layer in (almost every) node must respect. A standard or a protocol is a technical document that describes the details of communication. Every host participating in the communication must respect it. Examples:

- IEEE 802.11 (popularized as Wi-Fi) is a standard
- HTTP is a protocol your browser uses to receive a website from a server.

Two products from two vendors can communicate as long as they respect them.

Internet Layers: The TCP/IP Stack

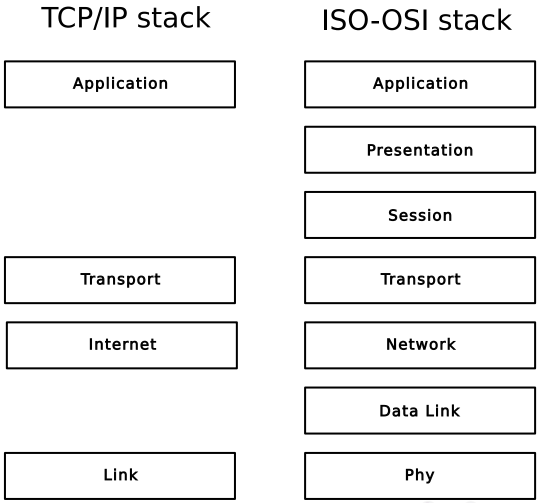
- Application Layer: describes the behaviour of applications
- Transport Layer: makes transmissions reliable and multiplexes connections
- Internet Layer: makes different physical networks interoperable, provides addresses and routing.
- Link Layer: prescribes how physical communication should work (wireless, wired, access to the media. . .)



TCP/IP Vs ISO-OSI Stack

The ISO-OSI stack is another, more detailed reference model. It adds some layers:

- Data Link: How multiple terminals should access the same phy medium
- Session layer: How to maintain user sessions
- Presentation layer: how to show/visualize data



Which stack counts?

As a rule of thumb:

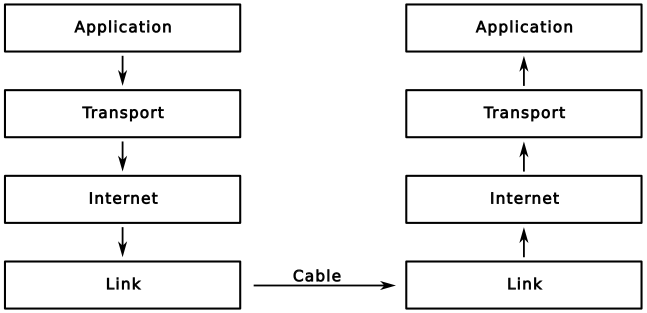
- The IETF, which is the organization that produces the so-called Internet Standards, the well-known RFC documents, uses the TCP/IP.
- RFCs deal with protocols that go from the Internet layer, up. Example: HTTP is an “application layer“ protocol.
- IEEE, or other entities that make low-level communication standards (WI-Fi, Ethernet etc.) use the ISO-OSI, as they generally don’t care of upper layers, but need more details in the low layers.

Network Stacks are the Rule

- The layered approach is a fundamental abstraction for Internet applications.
- Each layer is implemented in a separate piece of software.
- When programming for the Internet, developers need to care only about the layer in which they are operating.
- Ideally, every layer communicates only with the same layer in the other end of the communication
- There are well defined interfaces from one layer to another, and that's all developers have to know.

The Real Communication Path

- The abstraction is fundamental, but it is just an abstraction.
- In practice, the communication between two hosts passes through all the layers
- Transport/Internet/Link layers are implemented in the OS



Interfaces and Services

- Every layer offers a standard interface towards the upper and lower layers.
- With standard interface, i literally mean that when you are programming at the, let's say, application layer, you rely on the APIs that are offered by the Transport layer, and these APIs are standardized.
- We say that every layer offers a *service* to the upper layer.

Service Example

- We will see that the data packets can be lost, because the physical media is not perfect
- Some applications however require all data are delivered correctly, for instance, if you are downloading a binary application.
- Some other instead tolerate loss of data, for instance, a video call
- The transport layer then offers two services:
  - the *reliable* transfer of a stream of data (slower and more complicate)
  - the *unreliable* transfer of data packets (fast and simple, but lossy)
- That correspond to two different parameters in the API. Examples here and here.

Encapsulation (TCP/IP)

- Every layer *encapsulates* the data received by the upper layer.
- The Transport layer receives raw data from an application, splits it in *segments*, adds the TCP header and moves them down to IP.
- The IP layer takes the segment and adds another header, the IP header. This becomes a *packet* with two headers.
- The IP moves it down to the link layer, that adds another header, and then transmits it. We call it a *frame*.
- This process is called **encapsulation**, like when a letter is put in an envelope, and then in another one, and so on.
- When the packet is received the opposite process is followed. Every layer reads its own header, and passes the rest to the upper layer.

Security Consequences

1. Packet passes through intermediate routers which can read information
2. The source or the destination can not determine the path that will be followed.

Internet routers can spy on your traffic, and you don't know if your traffic will pass through routers in Italy, Europe, US, China or any other place you may not trust.

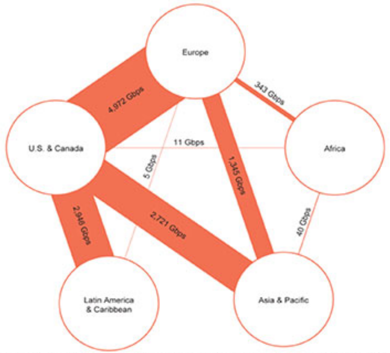
Interlude: The NSA vision of the Internet

TOP SECRET//SI//ORCON//NOFORN

(TS//SI//NF) Introduction

U.S. as World's Telecommunications Backbone

- Much of the world's communications flow through the U.S.
- A target's phone call, e-mail or chat will take the **cheapest** path, **not the physically most direct** path – you can't always predict the path.
- Your target's communications could easily be flowing into and through the U.S.



International Internet Regional Bandwidth Capacity in 2011

Source: Telegeography Research

TOP SECRET//SI//ORCON//NOFORN

Security enters the game

- This is why we need to enforce some kind of security measure in the protocols we use
- Early Internet engineers were too worried about making the internet *possible*, and overlooked the security implications
- After many years the Internet become commercial and widespread, and we started to review the old protocols and make some secure version of it.
- We are still in that process. . . .

License