**Faculty of Engineering of the University of Porto**

**Master in Informatics and Computing Engineering**

**Software Systems Architecture**

# Designing the Internet — Part 2

Homework 02

**Team 32**

José Francisco Veiga up202108753@up.pt

Marco Vilas Boas up202108774@up.pt

Pedro Lima up202108806@up.pt

Pedro Januário up202108768@up.pt

Pedro Marcelino up202108754@up.pt


February 2025

# Introduction

The Internet is a vast network of interconnected computers that enables global communication and information sharing among millions of people. It provides a fast and efficient way to exchange data and resources across the world, allowing people and machines to connect, collaborate, and access resources stored on the other side of the globe.

After proposing a possible architecture for the Internet in a previous document, as if we were in the 1970s and it did not yet exist, we now assess our proposal against the real design of the Internet's essence, analyzing in the paper *Extracting the Essential Simplicity of the Internet*, by James McCauley, Scott Shenker, and George Varghese.

# Q1 — Design Choices

The authors started by reducing the Internet into a small set of design options, which included a service model, a four layer architecture and three mechanisms - routing, reliability, resolution. Given these weren't enough to justify the Internet's success and longevity, the authors gave 4 more key reasons, being two of them:

**Modesty -** The decision to adopt a **best-effort service model** played a crucial role in the Internet's early **expansion** . By setting minimal infrastructure requirements, the Internet became accessible to a wide variety of network types, including those with limited resources and less sophisticated technology. Despite creating some less quality transference of packets and losing the ability to ensure reliability throughout the whole network it allowed a seamless integration of new nodes with minimal problems enhancing global growth of the network.
*"Being modest in performance requirements allowed the Internet to be ambitious in the speed and scope of its deployment."*

**Rough consensus and running code**: instead of adapting the "formal design committees" which were standard in the telephony world, the Internet chose a pragmatic approach where small teams tested ideas and implemented working prototypes, and then the community would choose which ones to adopt. This strengthened the philosophy of a decentralized, "communal ownership" Internet which connected everyone equally, encouraging innovation and adaptability.

# Q2 — Understanding the "Why"

Understanding why certain design choices were made in the development of the Internet helps us appreciate its success. Instead of focusing on how the Internet works, the authors look at the reasons behind the decisions made while creating the internet. The following examples highlight two important design choices, one about the Internet's service model and another about routing, where the why gave us additional insight into the internet design.

## Example 1: The Best-Effort Service Model

Why did the Internet's designers choose a simple "best-effort" service model instead of a more advanced one? They had to decide between creating a more technologically ambitious network that supports all application requirements or a simpler network that connects many different packet networks with only a set of more modest guarantees. They chose the second option, using a "best-effort service model" that does not promise reliable delivery or specific speeds. This decision made the Internet grow quickly because it did not require expensive infrastructure changes. Instead of relying on the network for strict performance, applications were designed to adjust to different levels of service. This flexibility, in turn, allowed the Internet to expand rapidly and support many different technologies.

## Example 2: Preventing Infinite Loops in Routing

Why is avoiding loops in routing so important? Routing is how the Internet decides where to send data packets. One key problem is avoiding loops, where packets could keep traveling in circles instead of reaching their destination. The internet prevents data packets from getting stuck in endless loops by using “link-state” routing, “distance-vector” routing algorithm to find the shortest-path routes to each destination and a "time-to-live" (TTL) field in each data packet. The reason this is important is that without careful design, routing loops could cause major congestion, slow down the entire network and even prevent the data from reaching its destination. By understanding why routing must prevent loops, we see that Internet engineers needed to find a balance between flexibility and efficiency in how routers communicate.

# Q3 — Decoding the Title

The title of the article, “Extracting the Essential Simplicity of the Internet”, denotes the intent of the authors’ to forget about complexities in mechanisms that are not essential to the core design of the Internet — “Looking past inessential complexities to explain the Internet’s simple yet daring design” — and often obfuscate one’s understanding of how it works — “its beauty is buried beneath an avalanche of implementation details” —, while its **essence** is rather **simple**.

Although the Internet is often perceived as a “soup of protocols such as IP, TCP, BGP, OSPF, DVRP, DNS, STP, and ARP”, the authors consider that those protocols have more to do with the “how”, while their aim is to lay bare the “why” of the Internet.

By doing so, the authors try to highlight the Internet's conceptual simplicity and the foresight of its original designers, who made decisions that allowed it to scale and adapt over decades despite massive changes in technology, applications, and usage.

# Q4 — Comparing Designs

Comparing our own design with the actual Internet design, we can notice some obvious differences that, considering the system we are trying to design, demonstrate that the Internet is in fact a system better than our own.

The first difference we noticed is that the Internet's design is rather simple, compared to ours which is a bit more complex. Considering that this system should be scalable and distributed globally, we should favor a system that is simple and ‘easy’ to implement and get working. In our system, we have a rather strict organization of the network, with defined rules for redundancy and fault tolerance (e.g. multiple links and clusters of routers), while the Internet just defines a set of rules for communication and routing, without explicitly needing a certain structure.

Another note about structure that we can point out on our design is the fact that we hang too closely to geographical definitions for continents, countries and so on. This may not be desirable at all in some cases, for example: some countries may have no strictly defined borders, because of wars and other political conflicts, while there may be entire regions that may be regarded by some as a country and by others as less. This would cause trouble in our hierarchy. Another pain point would be, for example, on the routing part, if a host decided to move its servers to another place in the world – we would need a way to know that this happened and correct the computers trying to communicate with them.

We also noticed that the Internet is way more modular and layered – as mentioned a few times in the article, the layers 1 through 4 need only to implement a common interface for communication with other layers, while being agnostic to the algorithms and code running inside the actual layer. This allows for different choices of implementations regarding each situation, and makes it very easy to scale and distribute this system, as one only needs to deploy more machines that implement these communication layers and protocols. In contrast, our system has a very strict routing strategy (following the hierarchy) and also a strict protocol that, while it does guarantee a lot of things, makes it harder to customize communication.

Another interesting thing we noticed is that, in our design, we tried to guarantee some aspects such as fault tolerance and load distribution by default in the system, through the redundant topology, error checking in the packets and so on. On the other hand, the Internet, to our surprise, assumes that ‘failures are normal’ and that, therefore, it is not the responsibility of the system to handle these. Instead, the Internet just implements a very simple, few guarantees base for communication, and then expects the protocols that we place on top of it to handle failures as we wish, or not handle them at all! Indeed, as we see today, for example, we have two main transport layer protocols, TCP and UDP. While the first tries to implement a stable, robust, fault tolerant (that is, handles packet-loss) connection, the latter does very little to prevent this – in fact, it is not even desirable, since, by not doing that, we can make the communication faster, which is great for real-time activities such as online gaming and streaming.

# Conclusion

This homework provided insight into the amazing architecture of the Internet, which is both simple and elegant. Its design cleverly hides complex details behind interfaces, making it appear almost magical to many people. While exploring its design, a comparison was made between our approach and the real implementation, allowing flaws and weaknesses to be identified in our design.