Faculty of Engineering of the University of Porto

Master in Informatics and Computing Engineering

Software Systems Architecture

Designing the Internet

Homework 01

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.
This project proposes a possible architecture for the Internet, as if we were in the 1970s and the Internet did not yet exist. The goal is to design a system that ensures fast, reliable, and scalable communication between devices, regardless of their location.
In this report, the necessary infrastructure is presented first, followed by the software that runs on these computers, and finally, the main challenges the proposed architecture must deal with.

Hardware and Infrastructure

The internet connects computers globally so that they can communicate and exchange data with each other. For these computers to be connected, there needs to exist some actual physical links, like cables, or even wireless links, like wi-fi. To achieve all the goals we have set for this system, we will need a significant number of *specialized* computers distributed across the world, organized in a hierarchical structure, that need constant administration and maintenance.

The structure we would need for our system to work would be hierarchical, taking advantage of the oldest stable hierarchical division of the world: continents, countries, regions, cities and local spaces. All across this hierarchy there would exist special computers called *routers*, whose job is specifically to guarantee that communication between different devices on the network is as available and fast as possible, organized in what we can later see as a tree like structure, with some redundancy included for fault tolerance. There would be a set of top-level routers in each continent, and going top-down in this layered structure we would have a set of routers by country, then by region, then by city and finally by local space.

By having this kind of infrastructure, we will be able to achieve some useful properties with our system such as speed, fault tolerance, topology changes and more.

Finally, we have a set of devices that will be constantly plugged in and out of the network – the devices used to run software, owned by people or other entities. These devices will be able to connect to the internet by establishing a connection to their local area router, which will make it identifiable in the global network, and can easily leave the network by disconnecting from the local router.

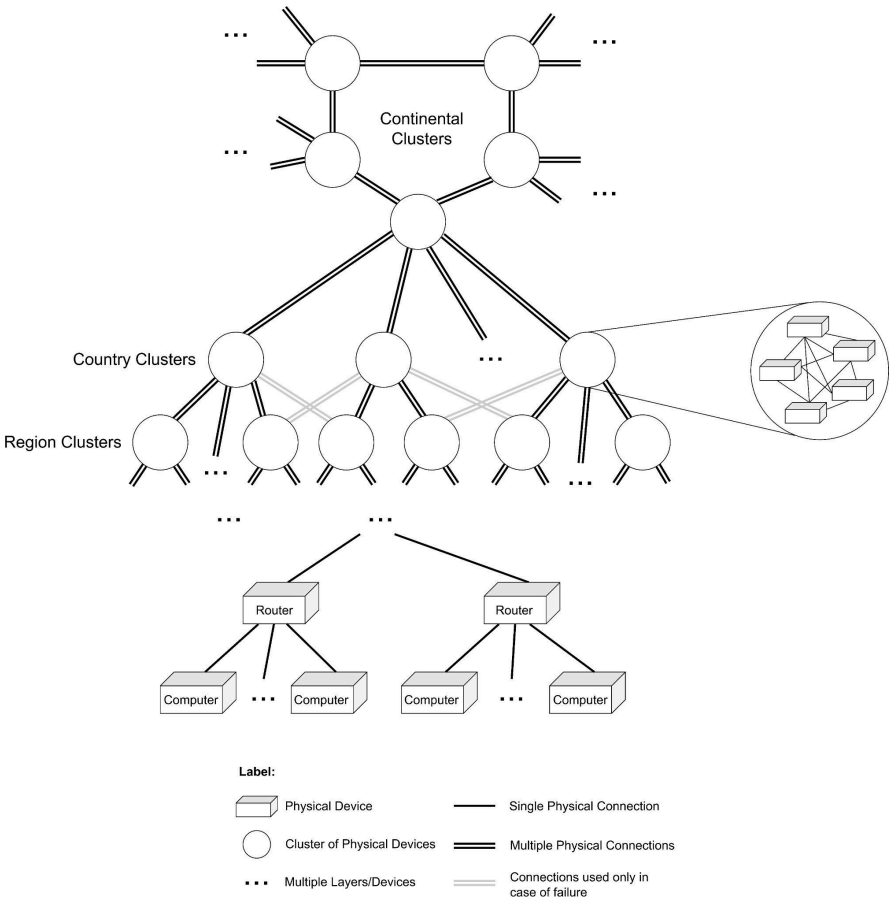

Figure 1.1 - Diagram of the Physical Structure

Software and Message Transmission

Identifying machines in the network

It is crucial to have a way of identifying machines that are part of the network, being the topology tree's leaves (the end users' devices) or the nodes in the intermediary layers (routers, etc.).

We envision that, when connected, each machine would be given a unique 64-bit address that identifies its position in the topology. The structure of each address would be as follows:

3 bits	6 bits	7 bits	10 bits	22 bits	16 bits
Continent	Country	Region	Municipality	Local Router	Machine
8 continents	64 countries/cont.	128 regions/country	1 024 municip./region	4 194 304 routers/ municip.	65 536 machines/ local

Figure 2.1 - Address Structure

When connected to the network, a machine would "ask" the one immediately above it in the topology for an available address among the possible sub-addresses for the topology sub-tree that it controls, as will be further detailed in the document.

Message Structure

When a machine, identified by their address, “wants” to pass some information to another machine, also identified by their address, it would have to structure it into one or more 512-bit messages according to a predetermined standard. Each message would have the following structure:

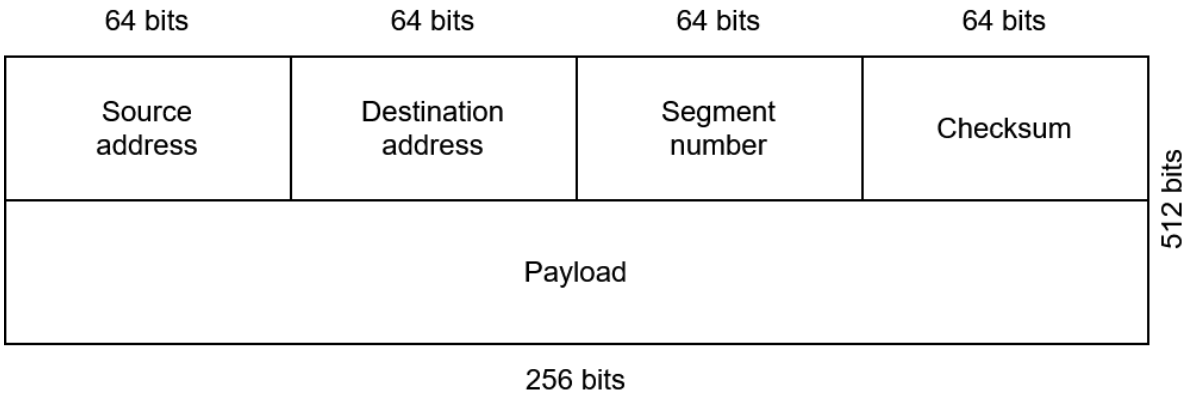

Figure 2.2 - Message Structure

The checksum is generated based on the other fields. The segment number is useful when the information to be transmitted needs to be divided into more than one message. The number indicates the position of the message’s payload in terms of the original information.

Transmission

Having the information structured according to what was established above, the main concern is the process of transmitting a message.

The sending machine would start by sending the message to their parent in the topology tree (e.g., a PC would send it to its local router). In the first moment, the parent machine would check the message’s checksum field against the other fields. If the checksum is correct, a positive response is sent back to the origin, indicating that the message was transmitted onwards. In case of not checking out, a negative message is instead sent, so that the origin can “decide” whether to try again. Then the machine shall look at the message’s destination and pass it on to the node above or a node below it on the topology, repeating the check/acknowledgment process every step of the way, until the message reaches its destination.

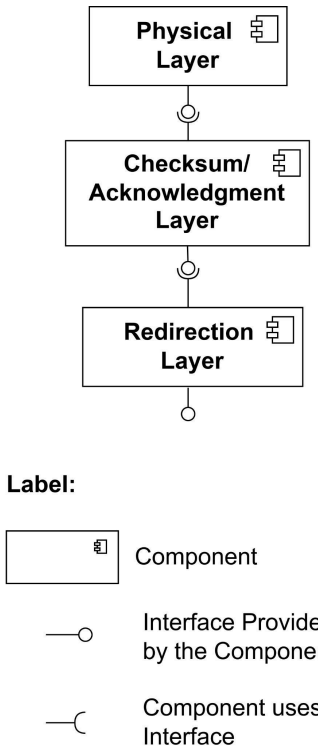

Figure 2.3 - Component Diagram

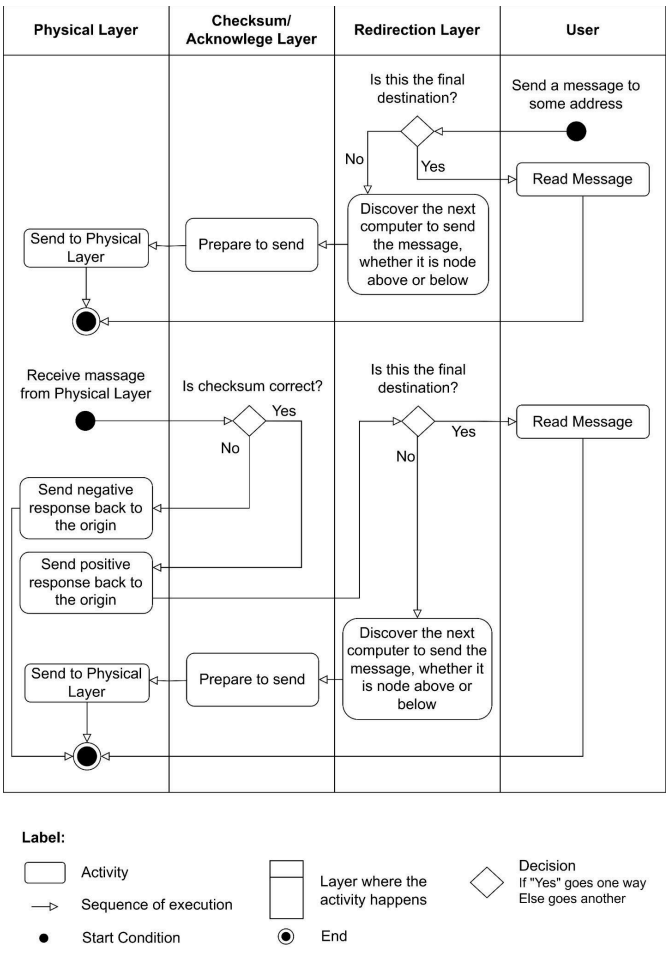

Figure 2.4 - Activity Diagram

Fault Tolerance

With millions of users in mind, potentially with multiple devices each, the network must be robust enough for handling occasional failures whether these are caused on purpose or by lack of maintenance.

In order to achieve this, this architecture should implement 2 main features: redundant hierarchical Links and backup routing nodes.

Redundant Hierarchical links

It is well established in our architecture that it should be composed of multiple subnetworks of subnetworks until we reach the bottom most hierarchical level where we have personal machines communicating with each other via a specialized machine assigned with the task of correctly transferring packets between each device. This enables communication with any other node in the network providing perfect conditions in a world where nothing goes wrong. However, in case of failure of the higher level subnetwork this entire section would become offline. By keeping a link to more than one higher ranked router and therefore adding redundant links, the network will be able to adapt until it is back up.

Backup Nodes for Routing

Despite our efforts to make the network as resilient as possible, each subnetwork still has a single point of failure with a critical role in packet transferring, the routers.
By introducing backup routing nodes or implementing a cluster approach, we minimize the risk of this issue while also contributing to load balancing to be talked about further in the following section.

Topology changes

Due to the scalable and distributed nature of the proposed architecture it predicts quite a dynamic topology, allowing the free addition and removal of nodes and links, to match the introduction of new users and organizations to the network.
In order to avoid causing disturbance to the users this process needs to be as seamless as possible, forcing the architecture to support these topology changes.

Addition

When a node joins a suitable network, it establishes a connection with the router responsible for the region being added in. It is the router's responsibility to give an available unique identifier that belongs to the subnetwork following the criteria specified in the software and message transmission section. Once the device's address has been established the router includes it in its routing tables, in order to correctly send a packet addressed to it.

Removal

When a node is removed from the network, whether due to failure, maintenance, or deliberate disconnection, its absence must be quickly detected to prevent data from being routed to an non-existent destination. The router responsible for the node should recognize its inactivity through mechanisms such as heartbeat signals or failed acknowledgments. Once confirmed, the router updates its routing table to remove the node's entry. To ensure smooth traffic flow, any active connections involving the removed node should be rerouted through alternative paths whenever possible. If the removed node plays a critical role, such as acting as an intermediary between subnetworks, backup nodes or redundant links should take over its function seamlessly.

The ability to seamlessly integrate new nodes allows the network to scale well as more users and organizations join, while the rapid detection and rerouting mechanisms for removed nodes prevent disruptions in communication. This adaptability, combined with redundancy and self-organizing principles, ensures that the network remains resilient, fault-tolerant, and capable of sustaining continuous operation even in the face of constant structural changes.

Load Balancing

As mentioned in the description, it is expected that the Internet will be used by millions of people. Therefore, efficient distribution of data traffic is essential. Congestion, slow

transmission speeds and bottlenecks, leading to inefficient communication, should be avoided.

Hierarchical load distribution in a tree-based network

Since the network is structured as a hierarchical tree, load balancing must be managed at different levels:

1. Local level - A home computer routes traffic through the home router.
2. City level - City clusters handle traffic between homes and local organizations.
3. Region level - A regional cluster is responsible for handling traffic between cities.
4. Country level - National clusters manage inter-region and national data flows.
5. Continental level - Top-level clusters manage continental communications.

Each node (personal router, city cluster, continental hub, etc.) is responsible for balancing the traffic among its children while maintaining optimal performance. Each cluster autonomously handles load balancing, providing encapsulation and abstraction of routing mechanisms at each hierarchical level. Nodes within a cluster continuously communicate with each other to assess availability, monitor traffic loads and adjust routes dynamically, ensuring a rapid adaptation to changes in the network.

Note that each cluster is spread throughout a city (city cluster), country (national cluster) or many countries (top-level continental cluster) rather than concentrated in a single physical place. This geographical distribution avoids localized failures (power outages, natural disasters), ensuring that problems in one area do not impact the whole cluster.

At the continental level, clusters must also communicate with each other to ensure global connectivity.

Adaptive path selection with exception handling

To balance traffic efficiently, routing nodes must collect real-time metrics, such as:

- Packet queue length
- Latency
- Failure rates
- Bandwidth

Each node dynamically adjusts preferred routes based on these. Normally, traffic follows the tree structure upwards and downwards. However, in an extremely unlikely case of failure at any level (e.g. a country cluster goes offline for some legal reason), child nodes can temporarily connect to an alternative parent node (e.g. other country), ensuring a continuous connection. This exception handling reroute allows the network to be reliable even when central assets fail.

Conclusion

In the attempt to design the architecture of an internet-like network from scratch with millions of users in mind, this report aimed at creating the most scalable and dynamic structure possible.

Our architecture consists of a hierarchical tree structure with specialized nodes capable of correctly redirecting packets - the smallest unit of information - to the correct node according to the unique identifier specified in the packet. Besides the basics, our architecture also provides fault tolerance, by adding additional routes to higher ranked subnetworks and by clustering critical components, which in turn, alongside the distributed process of routing packets will contribute to load balancing avoiding any unwanted disturbance to the overall network.

All in all, we believe this architecture will strive and prove to be effective in the handling of millions of users' worth of packet transfers with ease and efficiency.