



**Universidade do Minho**  
Escola de Engenharia

UNIVERSIDADE DO MINHO  
MESTRADO INTEGRADO EM ENGENHARIA DE TELECOMUNICAÇÕES E  
INFORMÁTICA

PROJETO DE TELECOMUNICAÇÕES E INFORMÁTICA I  
**REDE OVERLAY P2P DE ANONIMIZAÇÃO**  
RELATÓRIO

**DOCENTES:**

PROF. ANTÓNIO COSTA

PROF. HELENA RODRIGUES

10 de novembro de 2020

## 1 Membros



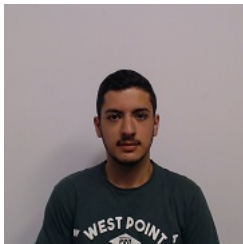
Hugo Machado (a80362@alunos.uminho.pt)

---



Micael Vieira (a85659@alunos.uminho.pt)

---



Pedro Correia (a83666@alunos.uminho.pt)

---



Pedro Silva (a77854@alunos.uminho.pt)

---

# Índice

<b>1</b>	<b>Membros</b>	<b>1</b>
<b>2</b>	<b>Introdução</b>	<b>5</b>
<b>3</b>	<b>Descrição do projeto</b>	<b>6</b>
3.1	Objectivos . . . . .	6
3.2	Fundamentos . . . . .	7
3.2.1	Redes P2P . . . . .	7
3.2.2	Onion Routing . . . . .	7
3.2.3	Proxy e VPN . . . . .	8
3.3	Planeamento . . . . .	10
<b>4</b>	<b>Desenvolvimento</b>	<b>12</b>
4.1	Arquitectura . . . . .	12
4.1.1	Infraestrutura . . . . .	12
4.1.2	Protocolo de comunicação . . . . .	13
4.2	Tecnologias . . . . .	15
4.3	Funcionamento . . . . .	16
4.3.1	Fase inicial de construção( <i>BootStraping</i> ) . . . . .	16
4.3.2	Encaminhamento interno . . . . .	16
4.3.3	Manutenção da rede . . . . .	16
<b>5</b>	<b>Conclusão</b>	<b>17</b>

## Índice de Imagens


1	Exemplo de <i>Onion routing</i> . . . . .	8
2	Diagrama de Gantt do projeto (Fase 1) . . . . .	10
3	Diagrama de Gantt do projeto (Fase 2) . . . . .	11
4	Diagrama de Gantt do projeto (Fase 2) . . . . .	11
5	Diagrama de Gantt do projeto (Fase 2) . . . . .	11
6	Diagrama da arquitetura da rede . . . . .	13

Índice de Tabelas

1      Tabela de ferramentas . . . . . 15

## 2. Introdução

O presente relatório reflete o desenvolvimento de uma Rede *overlay* P2P (*peer-to-peer*) de Anonimização, realizado no âmbito da Unidade Curricular Projeto de Telecomunicações e Informática I.

Neste documento iremos incidir num dos temas **que muitos** aram evitar, nomeadamente a proteção da privacidade dos utilizadores na **rede tradicional**.

É do conhecimento geral que é uma das áreas mais negligenciadas em aplicações distribuídas e também das mais controversas. Isto porque ao mesmo tempo que se pretende proteger a privacidade dos utilizadores, também se procura proteger **os xdfdzn<gr<zxhfserviços** disponibilizados de usos maliciosos e o grupo de utilizadores que usufruem do mesmo serviço.

Os servidores que disponibilizam esses serviços possuem ficheiros de log que relatam todas as comunicações dos utilizadores, podendo muitas vezes possuir informação que estes não gostariam que o provedor tivesse conhecimento.

Iremos, portanto, desenvolver uma solução que permite a proteção da privacidade dos utilizadores, mas que irá dificultar a auditoria com o intuito de detetar atividades maliciosas.

## 3. Descrição do projeto

### 3.1. Objectivos

O objetivo deste projeto[1] será desenvolver um serviço, recorrendo a infraestruturas e protocolos de rede já existentes, que pretende anonimizar o cliente. Ao invés do cliente comunicar e estabelecer uma ligação HTTP diretamente com o servidor, expondo assim os seus dados pessoais como endereço de IP, porta, sistema operativo, *browser* a ser utilizado, será utilizada uma rede P2P (*peer to peer*) como intermediário entre o cliente e o servidor, camuflando a verdadeira origem da conexão, mantendo assim a privacidade do cliente.

Os nós da rede P2P formam uma rede *overlay*, onde os pacotes provenientes do cliente serão reen-caminhados várias vezes pelos nós desta rede. Desta forma o servidor apenas conhece as informações dos nós encaminhadores, anonimizando o cliente.

A rede P2P poderia funcionar com apenas um nó, porém quantos mais *peers* existirem, maior será a segurança desta, sendo maior o numero de saltos que os pacotes dão antes de chegarem ao seu destino, mais difícil será descobrir a verdadeira origem deste.

## 3.2. Fundamentos

### 3.2.1. Redes P2P

Uma rede P2P[4] (*Peer-to-Peer*) trata-se de uma arquitetura de rede onde cada computador (denominado de nó) funciona simultaneamente como cliente e servidor, permitindo a partilha de dados sem a necessidade de um servidor central. Por não se basear em uma arquitetura cliente-servidor, onde apenas o servidor é responsável pela execução de todas as funções da rede, a rede P2P tem uma enorme vantagem por não depender de um servidor. Todos os nós estão interconectados permitindo o acesso a qualquer nó a partir de qualquer nó.

Na implementação de uma rede P2P podem ser seguidas três abordagens no que consta à estrutura da mesma:

- Rede P2P não-estruturada;

Este tipo de redes P2P não impõem uma determinada estrutura na rede *Overlay*, é sim formada por nós que formam aleatoriamente conexões entre si. Como não existe uma estrutura globalmente imposta, as redes não-estruturadas são fáceis de construir e são altamente robustas em situações em que um grande número de nós se conecta ou desconecta da rede frequentemente.

- Rede P2P estruturada;

Em redes P2P estruturadas, a rede *Overlay* é organizada numa topologia específica, e o protocolo implementado garante que qualquer nó possa procurar de forma eficiente um dado ficheiro/recurso que se encontre noutro nó. Porém para manter um encaminhamento de tráfego eficiente cada nó deve manter uma lista dos seus vizinhos.

- Rede P2P híbrida.

Em redes P2P híbridas são uma "mistura" do modelo P2P com o modelo cliente-servidor, e será esta a abordagem que iremos seguir na implementação da nossa rede *Overlay*. Um modelo comum de uma rede P2P híbrida é ter um servidor central que serve como ajuda para os nós se encontrarem uns aos outros. Atualmente este tipo de estrutura é a que apresenta um maior desempenho.

### 3.2.2. Onion Routing

*Onion routing*[3] é uma técnica para a comunicação anónima através de uma rede P2P. Numa rede *onion*, as mensagens são encapsuladas em camadas de encriptação.

No *onion routing* a conexão é mantida entre diferentes nós, quando um cliente quer, por exemplo, fazer um pedido HTTP a um servidor web, este pedido salta entre os vários nós e apenas quando atinge o



ultimo nó neste circuito é que este solicita o pedido HTTP ao servidor e a página *Web* desejada é enviada como resposta, utilizando a mesma rede de nós.

Quer o pedido que parte do cliente, quer a resposta do servidor são encriptadas com chaves diferentes, onde existe uma chave exclusiva para cada salto. O cliente tem acesso a todas as chaves mas o servidor apenas tem acesso às chaves específicas para a encriptação/descriptação daquele servidor.

Desta forma se existir, por exemplo, um *Man-In-The-Middle* e capturar o pacote entre dois nós, apenas terá acesso a uma mensagem encriptada e aos endereços IP destes dois nós, não conseguindo quer decifrar a mensagem, quer encontrar a verdadeira fonte do pedido, ou seja, o cliente permanece anónimo.

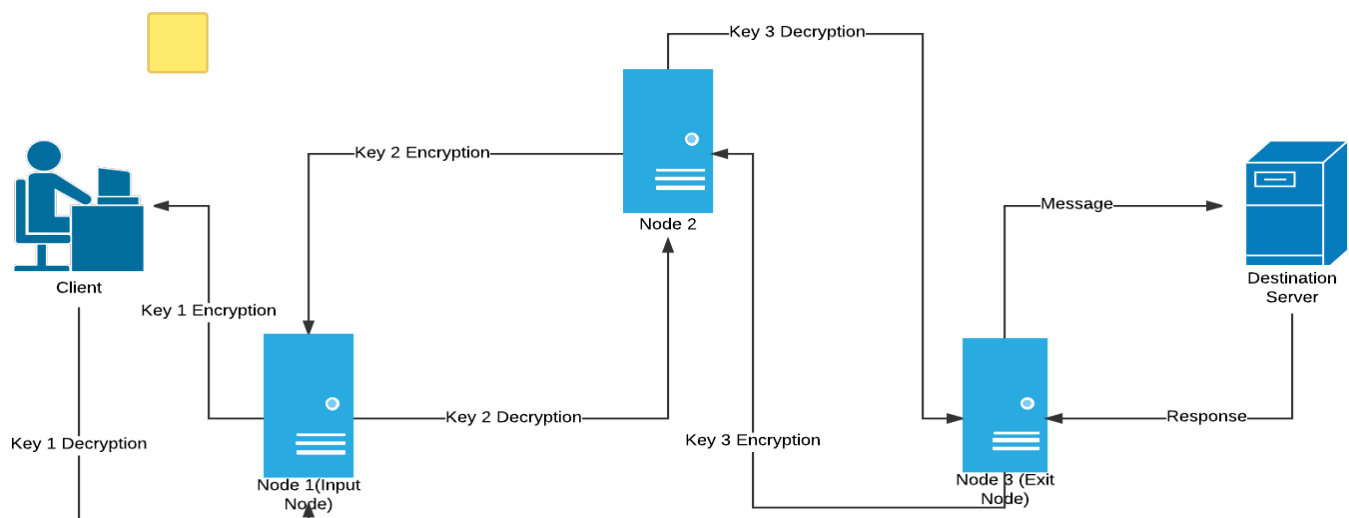


Figura 1: Exemplo de *Onion routing*

### 3.2.3. Proxy e VPN

Uma Proxy[6] é um servidor que age como intermediário para um cliente que pretendem comunicar com algum servidor, por exemplo, para carregar uma página *Web*. Quando uma máquina se liga a um Proxy "obedece" às regras definidas por este e todos os pedidos são feitos pelo próprio Proxy, que posteriormente os devolve ao cliente. Estando o cliente ligado à Internet através de um Proxy, o endereço IP deste não passa para além do Proxy, porém, normalmente, as comunicações entre o Proxy e os servidores não são encriptadas, e aí entram as VPN's.

Uma VPN[2] é um tipo de rede *Overlay*, que cria uma rede virtual por cima da já existente Internet, mas trata-se de uma rede privada. Quando um cliente estabelece uma ligação VPN é criado um canal de comunicação seguro, utilizando técnicas de encriptação e autenticação (por exemplo SSL/TLS). Quando

o cliente utiliza uma VPN, esta cria um canal de comunicação com outro nó que pode estar a quilómetros de distancia, e toda a atividade online do cliente será associada ao endereço IP deste nó e não ao seu endereço real.



### 3.3. Planeamento



A realização deste projeto está dividida em duas fases. Numa primeira fase, mais curta, é feito o levantamento das principais questões envolvidas e dos requisitos necessários. São decididas as tecnologias e os protocolos a dar uso, ou se necessário, implementar. Esta fase é o ponto de partida para a realização do projeto.

Na segunda fase já são desenvolvidos todos os estágios necessários para apresentar um produto final, é uma fase mais duradoura e desafiante.

Seguem-se os diagramas de Gantt para nos organizarmos ao longo do projeto:

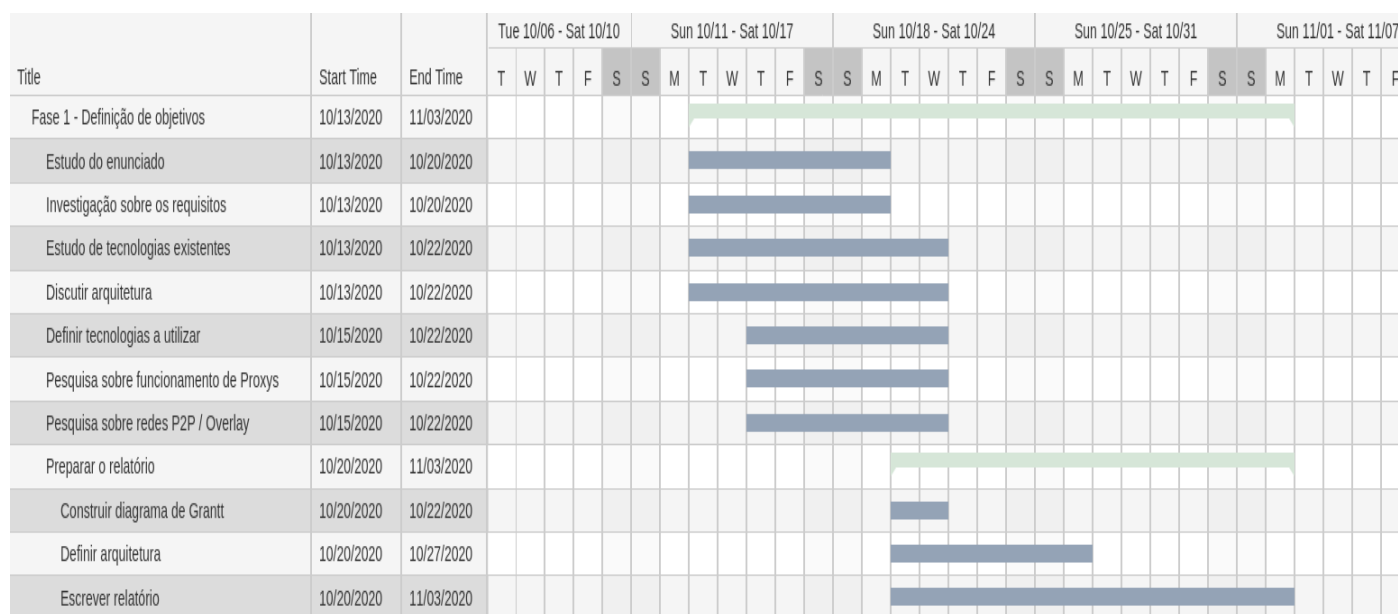


Figura 2: Diagrama de Gantt do projeto (Fase 1)

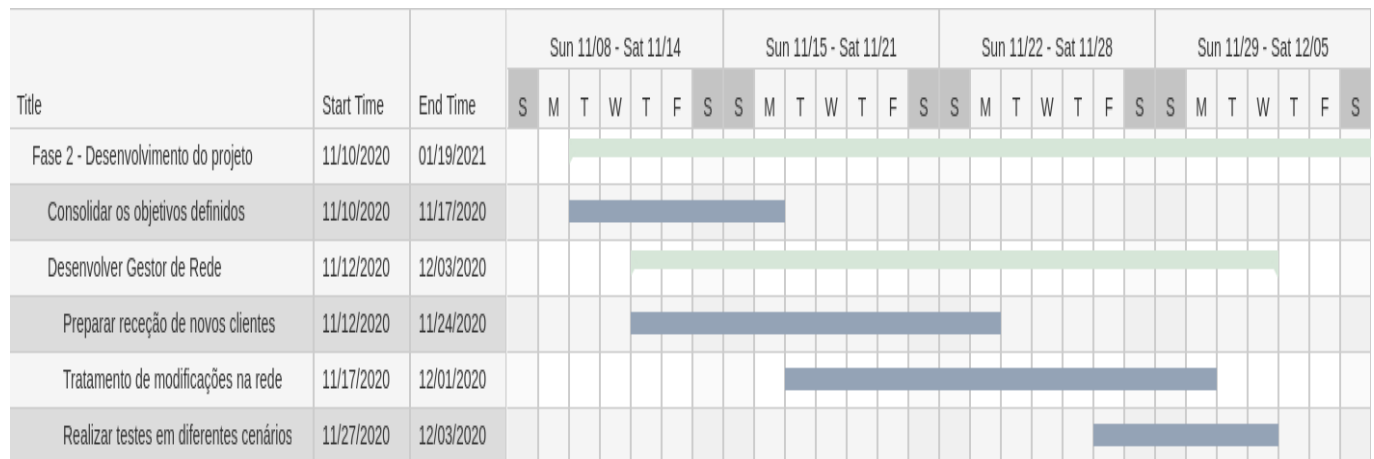


Figura 3: Diagrama de Gantt do projeto (Fase 2)

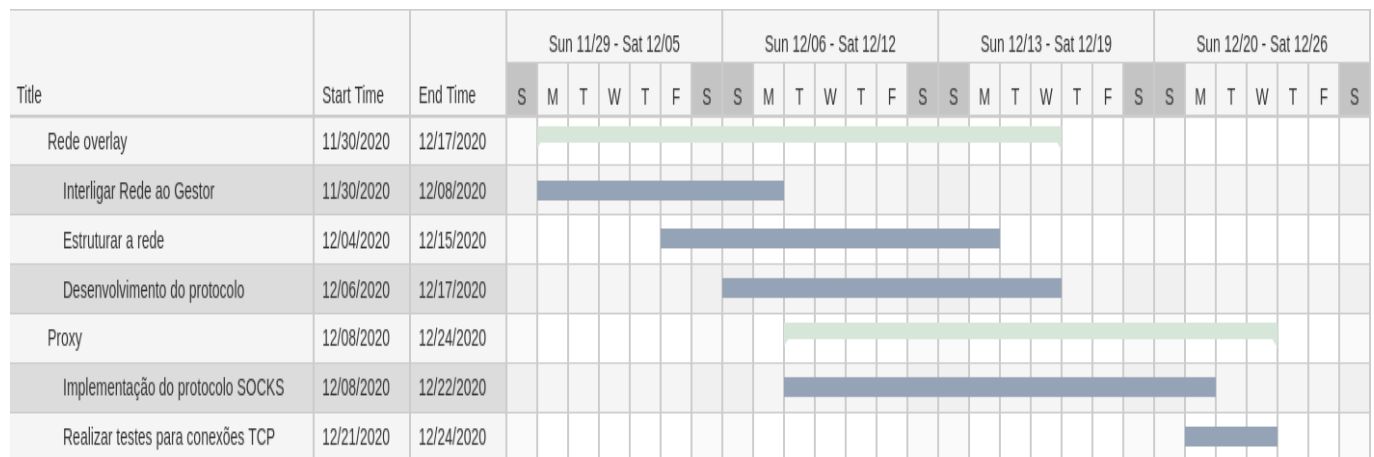


Figura 4: Diagrama de Gantt do projeto (Fase 2)

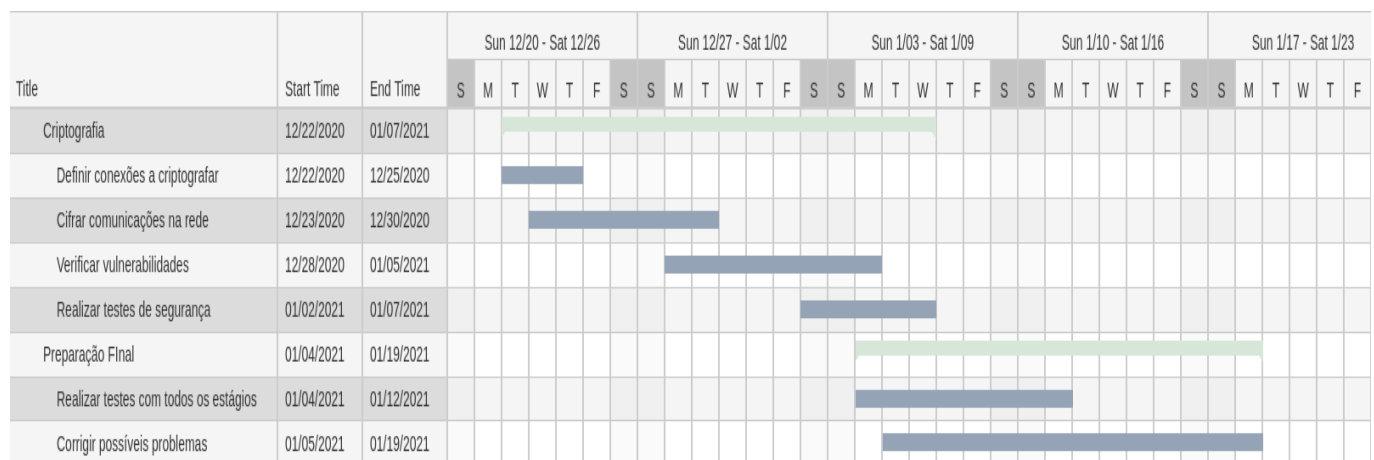


Figura 5: Diagrama de Gantt do projeto (Fase 2)

## 4. Desenvolvimento

### 4.1. Arquitectura

#### 4.1.1. Infraestrutura

Este projeto visa construir uma Rede *Overlay Peer-to-Peer* para Anonimização de tráfego *Web*.

A primeira etapa partiu por definir que tipo estrutura utilizar, e, com algum estudo das tecnologias e estruturas existentes, concluímos em definir uma estrutura Híbrida, obtendo uma maior flexibilidade.

Esta nossa estrutura híbrida, possui uma abordagem diferente das outras estruturas híbridas, pois será na sua base uma rede descentralizada, ou seja, não existe qualquer tipo de hierarquia nem super nós. Mas terá uma vertente centralizada com um ou mais nós, não pertencentes à rede, com a função de gestores para a construção e manutenção da mesma. E para que esta rede seja o mais descentralizada possível, os nós gestores vão ter apenas um conhecimento parcial dos nós constituintes da rede, ou seja, se os nós estiverem corretamente colocados o gestor eliminará os seus dados relativos à máquina.

Para a seguinte etapa estipulamos uma rede *Peer-to-Peer* estruturada em forma de matriz, de forma a tornar o encaminhamento dinâmico. Isto é, será possível encaminhar os pacotes por diferentes rotas com o mesmo destino, com o mesmo ou diferente número de saltos. Desta forma, evitamos também a identificação dos múltiplos nós da rede.

Para que esta infraestrutura funcione corretamente, definimos que cada membro terá um número mínimo e máximo de ligações com os seus vizinhos, e sendo esta infraestrutura em forma de matriz, cada membro idealmente teria quatro ligações(ou seja, quatro vizinhos). Mas para o formato de desenvolvimento e testes, limitamos a nossa rede a uma inicialização facilitada, definindo assim três tipos de nós pertencentes à rede:

1. O "*Vertex Node*", este é o primeiro nó da matriz posicionado no canto superior esquerdo, será único e terá apenas dois vizinhos;
2. O "*Edge Node*", este é um nó de aresta da matriz, estes serão posicionados pela aresta esquerda e superior da matriz, e terão apenas três vizinhos;
3. O "*Middle Node*", este é um nó intermédio da matriz e possuirá quatro vizinhos.

Por fim, cada nó será um cliente do nosso serviço, podendo este ser um encaminhador ou nó de saída na rede.

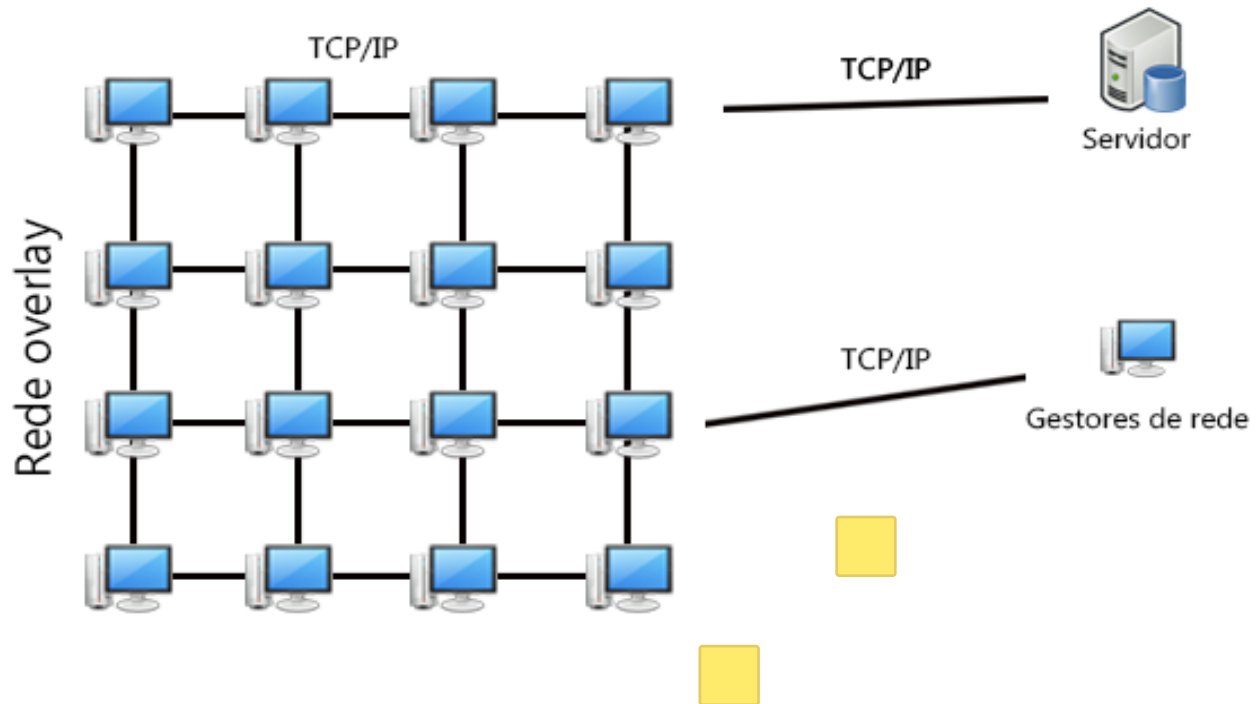


Figura 6: Diagrama da arquitetura da rede

#### 4.1.2. Protocolo de comunicação

Em relação ao protocolo de comunicação que será utilizado, apesar de não estar no seu formato final, já definimos os principais pacotes para a inicialização da topologia, encaminhamento e manutenção da rede *Overlay*.

Na inicialização:

- **Login** - Pacote direcionado do novo nó ao Gestor, para que este novo seja colocado na rede.
- **New Node** - Pacote direcionado do Gestor a um nó pertencente à rede, avisando o endereço IP e a posição do novo nó em relação a ele.
- **Connect** - Pacote direcionado do nó pertencente à rede para o novo nó, avisando da sua posição.

Depois no encaminhamento:

- **Request** - Este pacote tem um número de saltos pré-definido e a cada salto regista a direção percorrida, possui também o endereço IP do *WebServer* e o *Payload* correspondente.

- ***Reply*** - Pacote direcionado do nó de saída ao nó de origem, com novas direções para o mesmo destino, com a resposta do *WebServer*. Esta resposta poderá ser fragmentada.

Por fim, na manutenção:

- ***Patch up Request*** - Pacote direcionado do nó que perdeu um vizinho ao Gestor , avisando da posição do vizinho desconetado.

(Nota: as posições/direções transmitidas, são as posições relativas a dois nós, ou seja, se está em cima ou em baixo, direita ou esquerda)

Desta forma, todos estes pacotes serão devidamente identificados e todas estas ligações serão encriptadas. Como tecnologia criptográfica iremos utilizar certificados **x509**.



## 4.2. Tecnologias

No que diz respeito às tecnologias que utilizaremos no nosso projeto, como podemos visualizar no esquema da arquitetura, para a comunicação entre os *peers* da rede será utilizado o protocolo de comunicação TCP, tal como para a comunicação entre o *peer* de saída e o servidor. O nosso serviço necessita da confiabilidade da entrega dos pacotes, logo devido ao controlo de fluxo e deteção de erros que o protocolo TCP implementa será esta a melhor opção.

De forma a intercepar o tráfego proveniente do cliente e posteriormente encaminhá-lo para a nossa rede iremos utilizar o protocolo SOCKS, na sua última versão, SOCKS5[5]. Trata-se de um protocolo da camada aplicacional que estabelece a comunicação e troca pacotes entre cliente e servidor através de um proxy.

Para estabelecer a conexão com um cliente existe uma troca de pacotes, que consiste essencialmente nos seguintes passos:

- O cliente conecta-se através de um TCP *handshake*, e envia um pacote de "boas-vindas";
- O servidor escolhe um dos vários métodos de autenticação que o SOCKS5 fornece;
- O cliente envia um pedido de conexão;
- O servidor envia a sua resposta.

Uma vez com a conexão estabelecida o cliente poderá fazer pedidos HTTP através de um servidor proxy.

Uma vez com a conexão estabelecida o cliente poderá fazer pedidos HTTP através de um servidor proxy.

As ferramentas utilizadas estão descritas na Tabela 4.2, apresentada a seguir.

Ferramenta	Descrição
Core Network 6	Ferramenta para emulação de redes
Linguagens	C/C++ e <i>shell script</i>
Visual Studio Code	Editor de texto
Wireshark	Ferramenta usada para análise do tráfego na rede
Visual Paradigm	Ferramenta utilizada para desenvolver Diagrama de Gantt
Overleaf	Plataforma usada para escrita do relatório

Tabela 1: Tabela de ferramentas



## 4.3. Funcionamento

### 4.3.1. Fase inicial de construção(*BootStraping*)

Numa fase inicial, quando existe um **novο cliente** que se pretenda conectar à rede será tratado pelo gestor, pois é ele que será responsável por alocar os novos nós na rede, de forma a estruturá-la da forma que pretendemos. O gestor irá atribuir um identificador ao novo nó e se já existirem nós na rede, o novo cliente ficará conectado a estes. Caso seja o primeiro, irá ficar sozinho, e novos nós serão conectados a este.

### 4.3.2. Encaminhamento interno



A rede estará estruturada de forma a que cada nó no máximo esteja a ligado a 4 vizinhos da rede. Quando um cliente pretender aceder a um servidor, irá realizar 4 saltos dentro da rede, de forma aleatória, isto é, irá selecionar um vizinho a quem envia o pacote, este irá realizar o mesmo processo, até serem efetuados 4 saltos. O caminho que o pacote seguir vai sendo escrito no próprio pacote para que o último nó que efetua o pedido ao servidor, tenha noção de onde o pacote veio, de forma a que a resposta do servidor seja encaminhada para o nó principal que teve a intenção de realizar a conexão, não pelo mesmo caminho, mas por outro pseudo-aleatório, baseado no caminho principal que tomou.

### 4.3.3. Manutenção da rede



De forma a tirar o melhor partido do funcionamento do encaminhamento que se desenvolveu, a rede terá a funcionalidade de se auto ajustar quando um cliente se desconectar da rede. Como a rede irá manter uma estrutura em forma de matriz, com a saída de um nó central, o gestor será notificado pelos nós vizinhos do cliente que se desconectou, de forma a que um **próximo cliente** seja colocado estrategicamente naquele lugar.





## 5. Conclusão

Este projeto tem um caráter desafiante, uma vez que a construção de uma rede *overlay peer-to-peer* é algo complexo e que implica a conjuntura de diversas tecnologias, com diversos protocolos de comunicação a terem que ser utilizados e outros desenvolvidos para os nós se conhecerem e encaminharem tráfego entre eles.

Ao longo da primeira fase fomos cimentando ideias de como estruturar a nossa rede, de como encaminhar o tráfego do utilizador da rede para outro nó da mesma e de como edificar os protocolos necessários para que tal aconteça, tendo já sido obtido um esboço daquilo que será efetivamente ser utilizado para o produto final.

Contudo, temos plena noção que poderão surgir alterações com o decorrer do desenvolvimento da solução, podendo ser modificados alguns dos pacotes mencionados e outros adicionados.

## 5. Referências

- [1] *Enunciado PTII - Prof. António Costa e Prof. Helena Rodrigues.* (acedido em 13.10.2020).
- [2] *Everything You Need to Know About VPNs - Nick Mediati.* URL: <https://www.techsoup.org/Support/articles-and-how-tos/everything-you-need-to-know-about-vpns>. (acedido em 15.10.2020).
- [3] *Onion Routing - Computerphile.* URL: <https://www.youtube.com/watch?v=QRYzre4bf7I>. (acedido em 15.10.2020).
- [4] *Peer to Peer Overlay Networks: Structure, Routing and Maintenance - Wojciech Galuba e Sarunas Girdzijauskas.* URL: [https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9\\_1215](https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_1215). (acedido em 20.10.2020).
- [5] *SOCKS Protocol Version 5.* URL: <https://tools.ietf.org/html/rfc1928>. (acedido em 15.10.2020).
- [6] *What is a Proxy Server and How Does it Work? - Jeff Petters.* URL: <https://www.varonis.com/blog/what-is-a-proxy-server/>. (acedido em 15.10.2020).