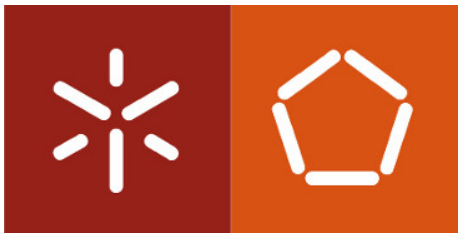
Rede Overlay de anonimização do originador

Comunicação por computadores

PL nr. 5
Grupo nr. 4

a84577	José Pedro Silva
a85700	Pedro Costa
a84485	Tiago Magalhães



Mestrado Integrado em Engenharia Informática
Universidade do Minho

1 Introdução

Este projeto foi desenvolvido no âmbito da Unidade Curricular de Comunicação por Computadores e tem como objetivo a implementação de uma rede overlay de anonimização do originador. Neste trabalho existem quatro entidades principais, sendo estas, o cliente, o servidor de destino e os 2 AnonGW que permitem a anonimização da conexão. São usados ainda dois protocolos de transporte, sendo estes, TCP e UDP, estando o TCP presente na comunicação de dados entre o cliente e o AnonGW nº1 e também entre o AnonGW nº2 e o servidor de destino. Por sua vez, o UDP é utilizado para a comunicação de dados entre os AnonGW.

2 Arquitetura da solução

2.1 AnonGW

Esta classe é onde é iniciado o programa começando por dar parse dos argumentos, neste caso será o **target server**,**porto do target server** e os **IP's das máquinas usadas para os AnonGW**. Após isto lança duas thread, sendo estas, **TCPListener** e **UDPListener** (estas duas classes serão abordadas com maior detalhe nas secções seguintes).

2.2 TCP Listener

Esta classe é responsável por abrir uma conexão TCP entre o cliente e o AnonGW, bem como gerar a chave e a cifra do cliente.Após isto é inicializada uma thread que irá garantir a conexão UDP entre os AnonGW e a cifragem da mensagem.

2.2.1 Estrutura do TCP Listener

```
/* Server socket que faz ligação cliente AnonGW */
private ServerSocket ss;

/* Lista de endereços IP dos AnonGW */
private InetAddress[] peers;

/* Porta à qual se liga o TCP (80)*/
private int port;

/* Map que guarda o socket correnspondente a cada sessão,
com a finalidade de saber o cliente ao qual
devemos enviar a resposta */
private Map<Session, Socket> tcp_sockets;

/* Hashtable que guarda os pacotes de cada sessão */
private Hashtable<Session, List<PDU>> map;

/* IP do target server  */
private InetAddress targetAddress;

/* Instância auxiliar para escolher aleatóriamente o primeiro AnonGW */
private Random random;
```

2.3 UDP Listener

Esta classe é responsável por receber pacotes UDP e inicilizar uma thread que irá potencialmente estabelecer conexão com o target server, ordenar os pacotes que recebe do AnonGW, cujo cliente comunicou, receber a resposta do servidor e por fim enviar a resposta para o AnonGW antecessor. Outro cenário possível para a classe UDPListener trata-se de quando recebemos os pacotes do ficheiro enviados pelo AnonGW com que nos encontramos a comunicar, no qual iremos adiciona-los também a uma lista para futuramente os ordenar e, de seguida, enviá-los para o cliente.

2.3.1 Estrutura do UDP Listener

```
/* Socket UDP entre os AnonGW */
private DatagramSocket udp_socket;

/* Array para armazenar informação  */
private byte[] receivedData;

/* Pacote */
private PDU packet;

/* Porta para qual irá enviar os dados */
private int redirectToPort;

/* Endereço IP para qual irá enviar os dados */
private InetAddress redirectToAdress;

/*  Map que guarda o socket correnspondente a cada sessão,
com a finalidade de saber o cliente ao qual
devemos enviar a resposta */
private Map<Session, Socket> tcp_sockets;

/* Hashtable que guarda os pacotes de cada sessão */
private Hashtable<Session,List<PDU>> pdu_map;
```

2.4 Session

Classe que representa uma sessão, onde, são guardados os endereços IP do cliente, target server e AnonGW a conectar.

2.4.1 Estrutura da Session

```
/* Endereço IP do cliente */
private InetAddress host;

/* Endereço IP do target server */
private InetAddress target;

/* Endereço IP do AnonGW a conectar */
private InetAddress randomPeer;
```

2.5 PDU

Protocolo que permite cumprir os requisitos propostos, este é composto por:

2.5.1 Estrutura do PDU

```
/* Indentifica ordem do pacote */
private int seq_Number;

/* Flag que indica se é o último pacote */
private int isLast;

/* Flag que indica que é a resposta ao cliente */
private int isAnswer;

/* Endereço a quem irá responder */
private String answerTo; // InetAdress

/* Informação do pacote */
private byte[] data;
```

3 Cifragem

Para cifrar a mensagem utilizamos criptografia simétrica, com o intuito de garantir a confidencialidade e a integridade. O algoritmo usado foi o DES (Data Encryption Standard). Quando o cliente estabelece ligação TCP ao AnonGW é gerada a sua chave e cifra.

```
/* Gera chave */
KeyGenerator keygenerator = KeyGenerator.getInstance("DES");
SecretKey chaveDES = keygenerator.generateKey();
System.out.println("Key used: " + Arrays.toString(chaveDES.getEncoded()));

/* Cria cifra*/
Cipher cifraDES = Cipher.getInstance("DES/ECB/PKCS5Padding");

/* Inicializa cifra */
cifraDES.init(Cipher.ENCRYPT_MODE, chaveDES);
```

A mensagem é encriptada e desincriptada consoante o fluxo da mensagem, isto é, a mensagem é desincriptada na comunicação com as extremidades (cliente e servidor) e encriptada nos intermediários (AnonGW).

```
decriptCipher.init(Cipher.DECRYPT_MODE, secretKey);
encriptCipher.init(Cipher.ENCRYPT_MODE, secretKey);

 /* Desincriptar */
 byte[] decryptedData = decriptCipher.doFinal(send.getData());

/* Encriptar */
byte[] encryptedData = encriptCipher.doFinal(Arrays.copyOfRange(data, 0, count));
```

4 Testes e resultados

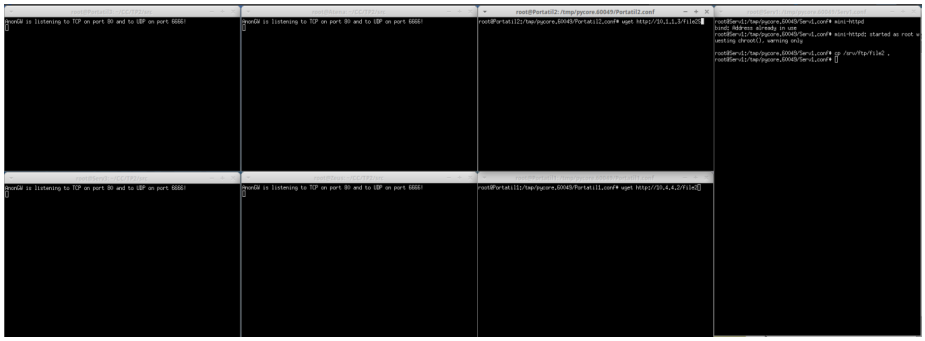


Figura 1: Cenário base sem pacotes a serem enviados.

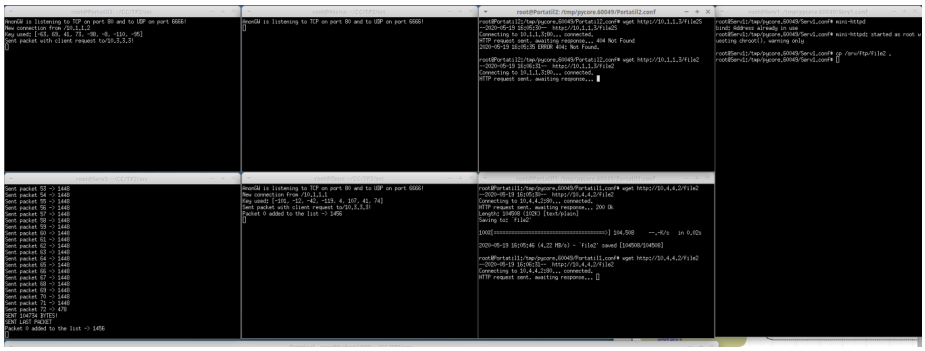


Figura 2: Cenário em que a informação está a circular.

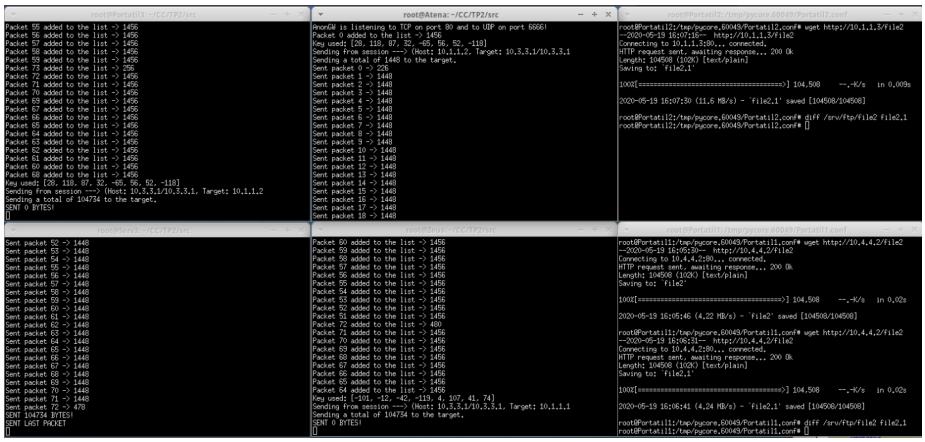


Figura 3: Cenário em que a informação chegou ao destino sem perdas.

Nota: *Devido às imagens não serem facilmente visiveis no pdf, decidimos portanto, adicioná-las na pasta imagens do projeto.*

5 Bibliotecas

As bibliotecas usadas foram:

- javax.crypto para a cifragem;
- java.net para o *sockets* UDP e TCP;

6 Conclusões e trabalho futuro

Com a realização deste trabalho, conseguimos perceber melhor na prática as diferenças entre TCP e UDP. Percebemos que, apesar das limitações do último, ao mesmo tempo fornece-nos um grande controlo e flexibilidade. Aprofundamos também o nosso conhecimento na área da encriptação assim como no transporte de dados. Neste trabalho todos os requisitos necessários foram alcançados, no entanto, numa perspetiva futura gostaríamos de fazer melhorias na implementação, isto é uso de um algoritmo mais poderoso de encriptação e controlo de perdas de pacotes. Adicionalmente, idealizamos ainda encriptar a chave de encriptação em Base64, cois