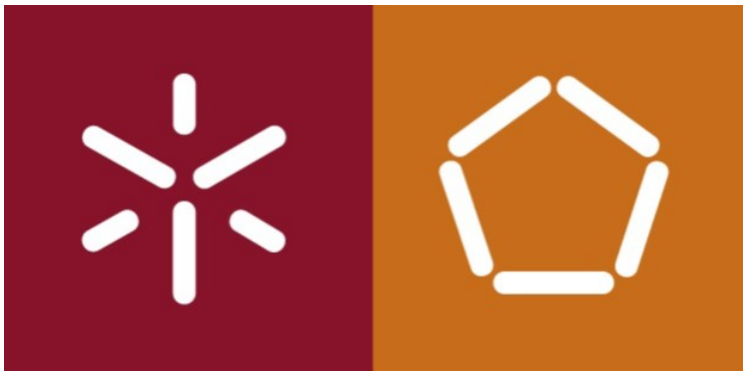

Universidade do Minho
Escola de Engenharia

Mestrado Integrado em Engenharia Informática

Comunicações por Computador

Relatório do Trabalho Prático 2

A78890 Alexandre Reis da Costa
A75248 Ana Sofia Gomes Marques
A65277 Flávio Manuel Machado Martins
A79799 Gonçalo Nogueira Costeira

Grupo 8

24 Maio 2020

Conteúdo

1 Introdução

No âmbito da unidade curricular de Comunicações por Computador foi proposto, no segundo trabalho prático, que se implementasse uma rede Overlay de anonimização do originador.

Então, para impedir um rasto auditável ao servidor(TargetServer), cliente(Origin) em vez de comunicar diretamente com este, liga-se por TCP a uma rede de overlay de nós. O nó escolhido irá aleatoriamente ligar-se a um outro nó por UDP, que por sua vez irá transmitir o pedido original por TCP ao servidor. O percurso inverso é usado de igual modo nas respostas do servidor.

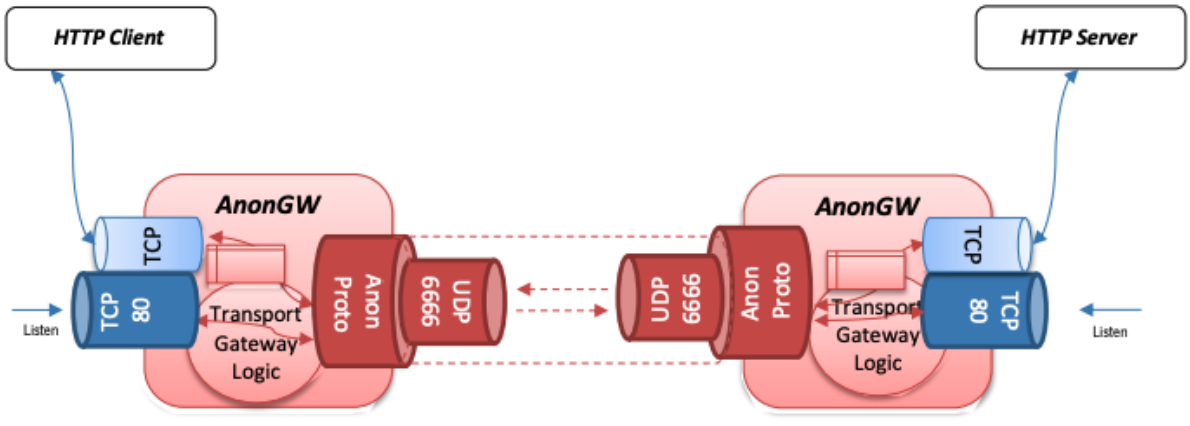

Figura 1: Maquete do enunciado.

Keywords: UDP · TCP · Anonimização · Rede · Overlay · TargetServer· Origin

2 Arquitetura da solução

A arquitetura que o nosso grupo optou por seguir, pode ser traduzido pela maquete abaixo apresentada:

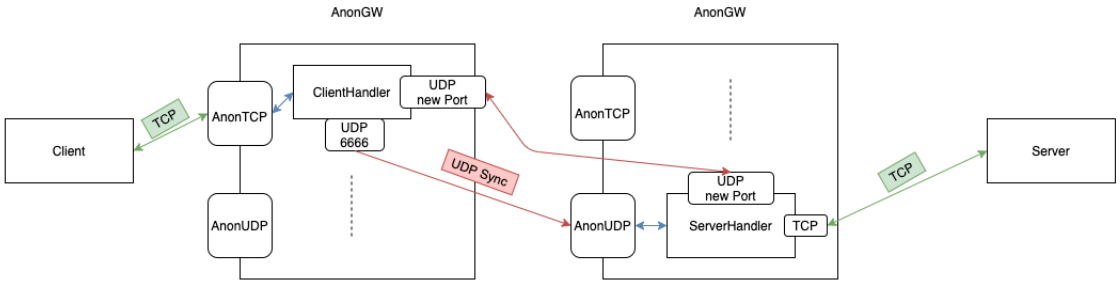

Figura 2: Maquete da nossa arquitetura.

O nosso AnonGW tem duas main threads. A thread AnonTCP que recebe os pedidos dos clients via TCP e cria uma thread para cada pedido (ClientHandler) reencaminhando-os encriptados via UDP. E a thread AnonUDP que recebe o connect dos outros AnonGW na porta 6666 cria uma thread (ServerHandler) na porta enviada na mensagem de conexão e reecaminha o pedido desincriptado via TCP para o servidor.

A resposta do servidor percorre o caminho contrario aproveitando as conexões já estabelecidas via TCP e as portas reservadas para cada pedido via UDP.

3 Especificação

3.1 Formato das mensagens protocolares (PDU's)

De forma a simplificar as conexões as nossas mensagens são bastante simples.

Port -> A conexão é estabelecida enviando a porta ao AnonGW escolhido via UDP na porta 6666;

Done -> A conexão é fechada quando o AnonGW que recebeu a resposta do servidor envia um "done"para o AnonGW que espera a resposta;

3.2 Interacções

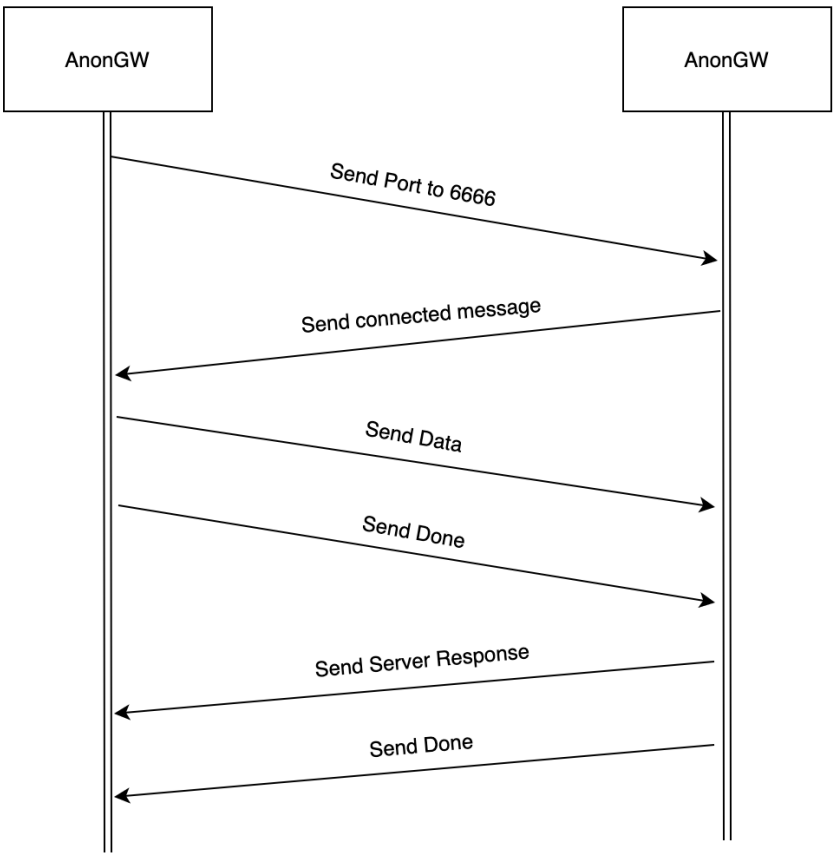

Figura 3: Protocolo AnonGW - AnonGW.

O AnonGW que recebeu o pedido via TCP começa por enviar um pedido ao outro AnonGW via UDP na porta 6666 com a porta que os dois devem comunicar para tratar daquele pedido. De seguida o segundo AnonGW envia um mensagem de confirmação e abre essa porta. Posto isto o primeiro AnonGW envia o pedido e no fim envia um "done"que indica o fim do pedido.

Depois do segundo AnonGW receber a resposta do servidor via TCP reencaminha essa resposta de volta para o primeiro AnonGW via UDP na porta anteriormente definida. No fim envia um "done"assinalando que a resposta foi toda enviada. Fechando assim as conexões.

4 Implementação

O nosso projeto consiste em 6 classes.

AnonGW -> Esta class tem o método main que recebe os parametros de configuração do AnonGW e inicia as main threads.

AnonGWTCP -> Esta class serve para controlar os pedidos dos clients via TCP. A cada pedido cria uma thread da class ClientHandler.

AnonGWUDP -> Esta class serve para receber os pedidos de conexão via UDP de outros AnonGW na porta 6666, esta conexão controla se os pedidos são enviados dos ips conhecidos. De seguida cria uma thread da class ServerHandler para tratar o pedido.

ClientHandler -> Esta classe controla a thread do AnonGW que trata o pedido do client e reencaminha para outro AnonGW. Para incialização da mesma é dado uma comunicação TCP ao cliente e um endereço do irmão com que se deve esconder do servidor. De inicio é criado uma ligação UDP com o endereço onde está o AnonGW com que deve estabelecer comunicação e envia a porta dessa ligação para esse endereço na porta 6666. De seguida fica a aguardar o sinal de comunicação que irá receber já na nova porta vindo do AnonGW do lado do servidor.
Em seguida é lido via TCP o pedido do cliente e reencaminhado via UDP para o AnonGW com que já foi estabelecida ligação, no final do pedido é enviado um "done"sinalizando que o pedido está totalmente enviado.
Por fim fica a aguardar a resposta do AnonGW que vem via UDP e é enviada para o cliente via TCP.

ServerHandler -> Esta classe serve para controlar a thread do AnonGW ligado do lado do servidor, que recebe o pedido via UDP de outro AnonGW e envia o pedido ao servidor, obtendo a resposta e enviando de volta para o mesmo AnonGW. Para inicializar a mesma é necessario criar um socket(ligado ao AnonGW do lado do cliente), é ainda recebido uma porta, onde os anons vao estabelecer a conexão UDP, o endereço do servidor protegido pelo AnonGW e ainda o endereço do AnonGW do lado do cliente. Aquando do inicio dum ServerHandler é enviado um sinal ao AnonGW do lado do cliente informando da conexão via UDP por esta thread. Então inicia uma espera pelos pedido do anon que está do lado do cliente, este pedido é obtido via UDP no socket previamente iniciado. O final do request é reconhecido pelo envio de um "done"do lado do cliente. Após receção total do pedido este é enviado então ao servidor protegido por este anon via TCP. De seguida recebemos a resposta do Servidor via TCP e fazemos o envio, da mesma forma que enviamos o sinal previamente, para o AnonGW do lado do cliente. Após o envio da resposta ao pedido é enviado também um sinal "done"que indicará o fim da resposta ao AnonGW do lado do cliente.

AESencrp -> Esta class serve apenas para encriptar e desincriptar os pedidos.

5 Testes e Resultados

Os testes e resultados podem ser consultados em anexo.

6 Conclusão

Em modo de conclusão, o trabalho cumpriu maioria dos objectivos básicos com sucesso apesar de algumas dificuldades no que toca à decisão da arquitectura a adoptar. A compreensão de protocolos já existentes e a respectiva adaptação para a implementação também se revelou uma dificuldade para o grupo.

Contudo, o grupo reconhece que o trabalho possui falhas e que poderia ser melhorado e optimizado em varios aspectos. Este projecto revelou-se crucial para a consolidação de matérias relativas à camada de transporte e de aplicação da pilha protocolar.

7 Anexos

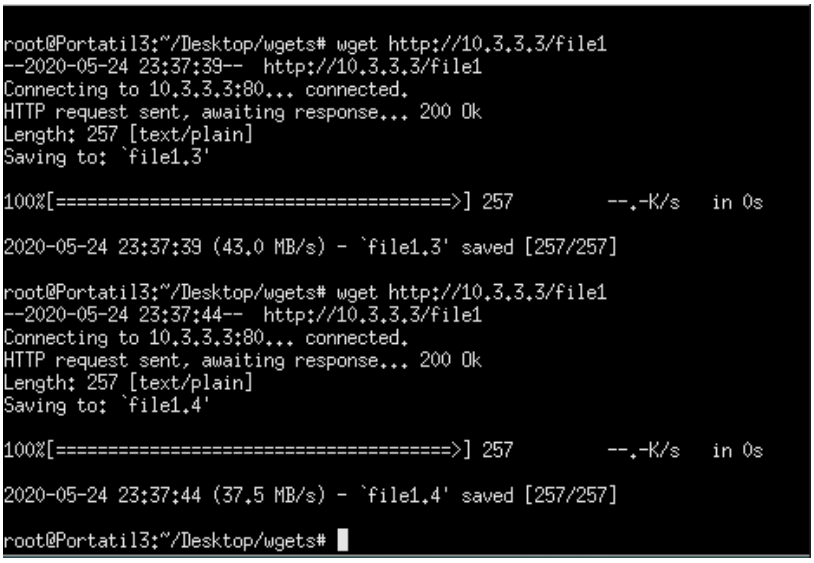

Figura 4: Transferência

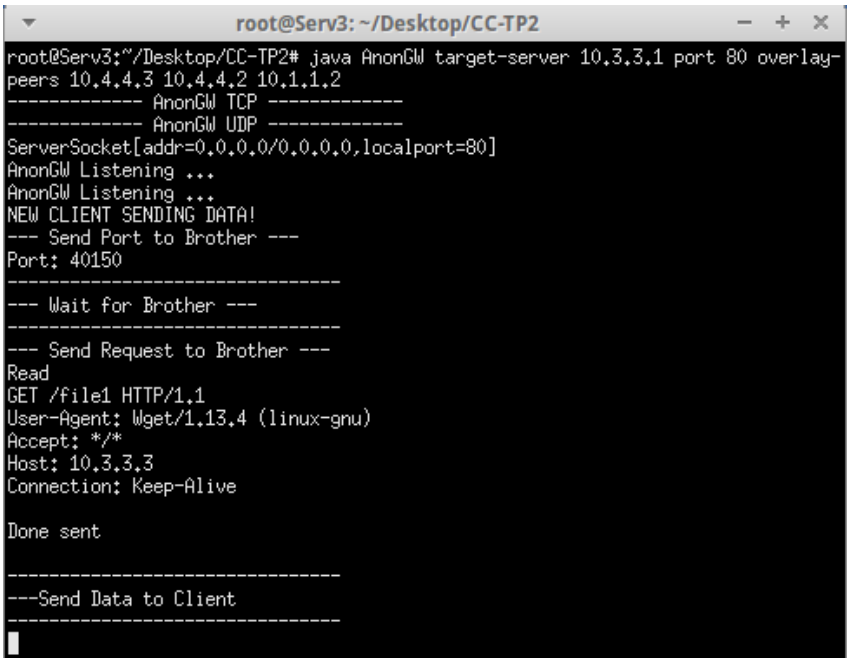

Figura 5: Client Side AnonGW

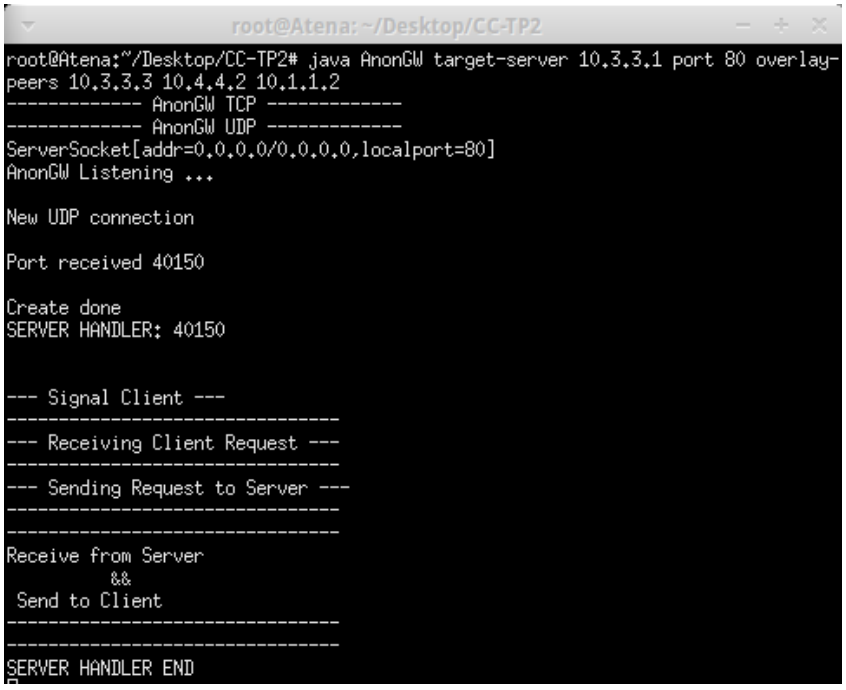

Figura 6: Server Side AnonGW

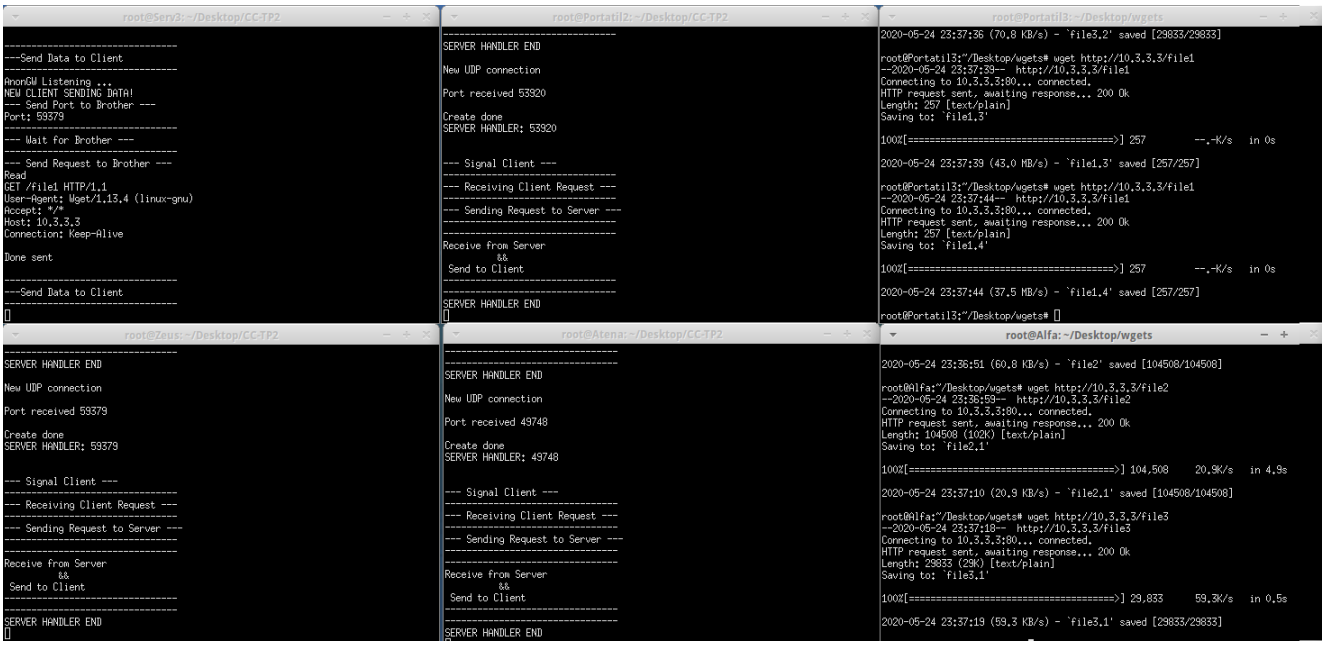

Figura 7: Execução em simultâneo