Trabalho Prático 2
Comunicações Por Computador

Hugo Cunha (84656), Maria Pires (a86268), and Susana Marques (84167)

Universidade do Minho
Departamento de Informática

Resumo: O segundo trabalho prático, no âmbito da unidade curricular Comunicações por Computador tem como objetivo o desenho e implementação de um gateway de transporte para uma rede de anonimização designado por AnonGW. Posto isto, será necessário desenvolver um protocolo que funcione sobre UDP e garanta a entrega ordenada e confidencial dos dados de uma ou mais conexões de transporte TCP.

Keywords: UDP · TCP · Fiabilidade · Controlo de fluxo · Controlo de congestão

1 Introdução

Este trabalho iniciou-se pela implementação das componentes TCP (cliente/servidor) e estruturas de dados internas. De seguida, foi desenhado e posteriormente implementado o protocolo Anon Protocol com os respetivos PDU, e, finalmente, implementamos as componentes UDP e Anon Protocol. Todos os dados enviados pelo cliente na conexão TCP são recebidos pelo primeiro AnonGW e imediatamente enviados pelo túnel UDP anonimizador para um segundo AnonGW, que por sua vez os faz chegar numa nova conexão TCP ao TargetServer. Todo o programa foi implementado recorrendo à linguagem Java e testado na topologia CORE, fornecida pela equipa docente.

2 Arquitetura e Implementação da solução

2.1 Inicio da ligação

O cliente inicia a conexão com o AnonGW através de um *http request*. Após isto, é criada um sessão onde é selecionado aleatoriamente um segundo AnonGW, para onde serão encaminhados os dados através de uma ligação UDP. O segundo AnonGW inicia uma sessão com as informações de controlo recebidas no PDU da ligação UDP e estabelece uma ligação com o servidor final.

2.2 Formato das mensagens protocolares (PDU)

O PDU está associado a uma sessão, que corresponde a uma ligação entre o cliente e o servidor. Este é composto por um número de sequência único com base aleatória, por uma flag usada para vários propósitos como, por exemplo, identificar ACKS, trocar chaves, entre outros.

Também contém um timestamp que refere ao momento em que foi criado ou modificado, um corpo (dados), e padding para balançar o tamanho.

Estes campos podem ser usados livremente para diversos propósitos, por exemplo, um ACK pode ser representado por um PDU sem dados com apenas o id de sessão, o número de sequência e a flag, ou seja um pacote para trocar chaves tem um campo específico para enviar a chave sem precisar do id de sessão ou do número de sequência.

2.3 Controlo de Perdas e Ordenação dos pacotes

Para ordenar os pacotes e controlar as perdas usamos uma estrutura auxiliar designada de Sliding Window que é essencialmente uma queue com uma base dinâmica.

À medida que se recebe os pacotes é enviado de volta um ACKS e os pacotes recebidos são inseridos na Sliding Window, tendo como índice o seu número de sequência. De seguida é removido um array de pacotes seguidos do início da Sliding Window. Este segmento pode ser imediatamente processado passando, através de uma queue, para a Thread responsável pelo output para o Socket TCP.

À medida que é feita a remoção da queue de envio de pacotes UDP, pacotes de dados são colocados numa instância da Sliding Window e são enviados por UDP.

Ao receber ACKS os pacotes associados a esse número de sequência na Sliding Window são removidos e da próxima vez que for lido um pacote da fila de espera é extraído um array de pacotes da Sliding Window que ainda não foram removidos e que originaram um timeout. Estes pacotes são inseridos outra vez na fila de espera e a base da Sliding Window é reajustada para não considerar os pacotes nulos que estejam no início.

2.4 Confidencialidade dos dados

A sessão é iniciada com geração e troca de chaves RSA de 1024 bits entre os dois anonGW, isto permite a existência de possibilidade de estabelecer uma comunicação altamente segura incluindo estabelecer algum nível de Autenticação de Origem. Dada a natureza algoritmicamente pesada deste tipo de encriptação os pacotes em si, são encriptados com uma chave simétrica AES de 128 bits gerada no inicio da sessão que posteriormente é cifrada com a chave publica do remetente e é colocada numa estrutura Container, com o objectivo de encapsular a mensagem e a chave cifradas para serialização obtendo assim uma encriptação segundo um modelo de "Envelope Digital". Um fluxo de pacotes entre o Cliente

e o Servidor são registados numa sessão de forma apenas suficientemente identificativa para manter o canal funcional. Nesta constam um id único partilhado por todos os gateways do canal, no entanto cada instância só tem um socket TCP e um endereço de AnonGW indiscriminados de forma apenas ser possivel descobrir as duas máquinas adjacentes. O id de sessão é cifrado com o pacote.

2.5 Multiplexagem do canal

Cada AnonGW tem uma Thread de um servidor TCP e um servidor UDP. O servidor TCP permite que o AnonGW estableça conexões com múltiplos clientes na porta 80 associando a cada um uma sessão com os seus próprios Threads para processar os dados, nomeadamente um TCPServerWorker, cujo trabalho é enviar os dados pelo canal através de um DatagramSocket partilhado pelo AnonGW inteiro. O servidor UDP é único para todas as sessões, está constantemente a receber e processar pacotes num DatagramSocket aberto no port 6666. O processamento que o DatagramSocket faz consiste em desserializar, desencriptar e encaminhar os pacotes UDP recebidos para os Threads responsáveis pela sessão. Desserializa os bytes em Containers, desencripta a chave do Container com a sua chave privada e o PacketUDP com essa chave, devolve pacotes de ACK para o remetente com o número de sequência respetivo ou processa a troca de chaves dependendo da flag no pacote recebido, finalmente desmultiplexa os PacketUDP ao colocá-los na queue da sua respetiva sessão.

3 Detalhes da Implementação

Para a implementação do AnonGW usamos várias bibliotecas de suporte:

- **javax.crypto e java.security:** Cifrar em RSA-1024 e AES-128 e gestão das chaves criptográficas;
- **java.util.concurrent:** Estruturas de dados concorrentes para partilha de dados entre Threads;
- **java.util.Base64:** Codificação de elementos atómicos segundo o standard de texto;
- **java.net:** Sockets TCP e UDP, Endereços IP;
- **java.time:** Gestão e geração de Timestamps.

Tendo em conta o mínimo do MTU do protocolos IP previmos que o tamanho máximo dos dados deveria de estar à volta dos 1024B. Como não é implementada o controlo de congestão estabelecemos que o tempo ideal para o Timeout deve estar por volta dos 300ms para minimizar as tentativas de reenvio caso a performance se degrade. O tamanho ideal para o buffer da Sliding Window foi estabelecido como 300 pacotes garantindo espaço suficiente para enviar um ficheiro de 200KB sem causar colisões e um ficheiro de 1MB mantendo pouca variação na velocidade de download.

O PDU contém um campo de flag com diversos usos, representada por um *Integer* que pode representar os seguintes valores:

- **1:** Pacote de dados normal
- **-9:** Termino forçado de Threads
- **328:** Request de chave pública
- **329:** Reply de chave pública
- **200:** ACK
- **201:** ACK por enviar

4 Testes e Resultados

À medida que fomos desenvolvendo o protocolo foram efetuados diversos testes para garantir que o resultado final corresponderia com o pretendido, apresentamos de seguida o resultado de alguns testes realizados após finalizar a implementação do protocolo.

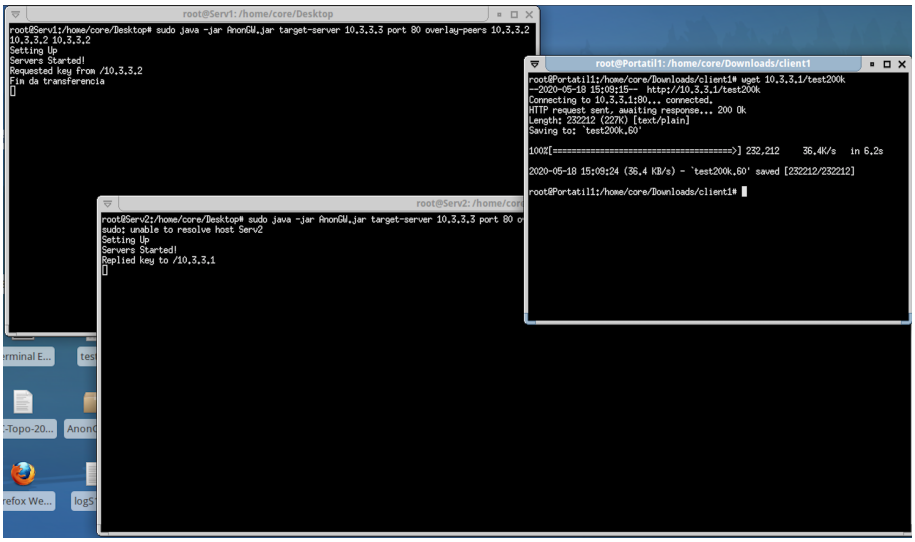

Fig. 1. Transferência de um ficheiro de 200k

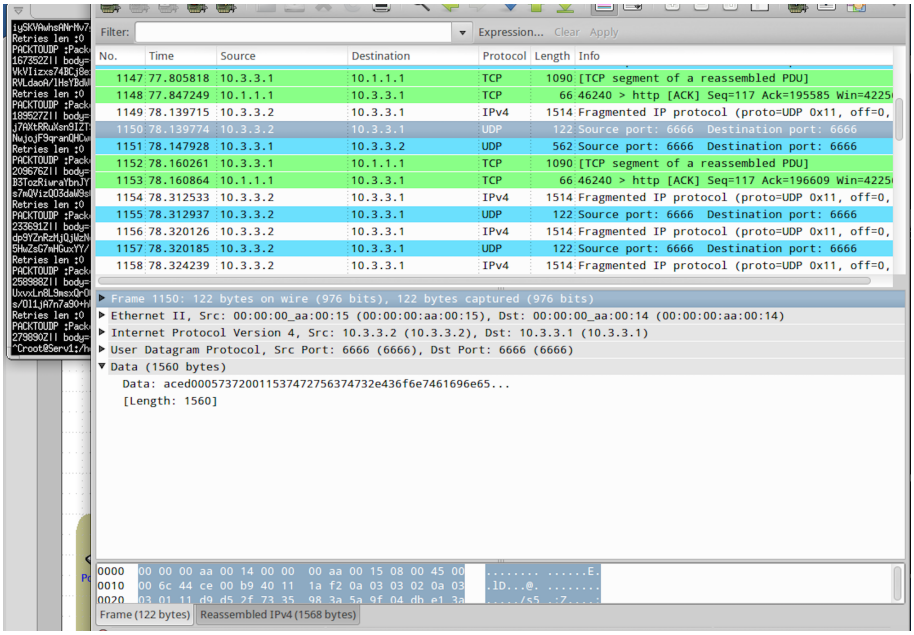

Fig. 2. Captação de pacotes através da ferramenta Wireshark

5 Conclusões

Podemos concluir que o protocolo desenvolvido é capaz de lidar com múltiplos clientes e é um protocolo fiável, que garante a entrega ordenada e íntegra dos dados enviados e a sua confidencialidade, com controlo de perdas. Todos os objetivos foram alcançados, com exceção da autenticação de origem, pois apesar da simplicidade de implementação a performance do protocolo é gravemente afetada, pelo que o grupo decidiu não incluir essa componente adicional.

Referências

1. Kurose, James F., Ross, Keith W.: Computer Networks - A Top-Down Approach. 6th edn. Peason Education, 2013