Trabalho Prático 3: Streaming de áudio e vídeo a pedido e em tempo real*

Luís Carlos Sousa Magalhães[1 [PG47415]], António Jorge Nande Rodrigues[2,3 [PG47030]], and Francisco Correia Franco[3 [PG47187]]

[1] University of Minho, Braga, PT
[2] University of Minho, Braga, PT
[3] University of Minho, Braga, PT

Resumo No âmbito da unidade curricular de "Engenharia de Serviço em Rede", a equipa docente lançou o reto de desenvolver um serviço "Over the Top"para entrega de conteúdo multimédia, com o intuito de evitar perda de qualidade durante o serviço. Assim sendo, sobre uma topologia "underlay"construímos uma abordagem mais eficiente, criando uma aplicação que vai correr nos nós da rede overlay. De forma a alcançar o objetivo final mais facilmente, esquematizou-se todo o trabalho em cinco etapas distintas, que serão abordadas ao longo deste relatório.

- Etapa 0: Preparação das atividades
- Etapa 1: Construção da topologia overlay
- Etapa 2: Construção das rotas para os fluxos
- Etapa 3: Teste dos percursos no overlay
- Etapa 4: Streaming

Keywords: OTT · Server · Client · TCP · UDP · Stream.

1 Arquitetura da solução

* Supported by University of Minho.

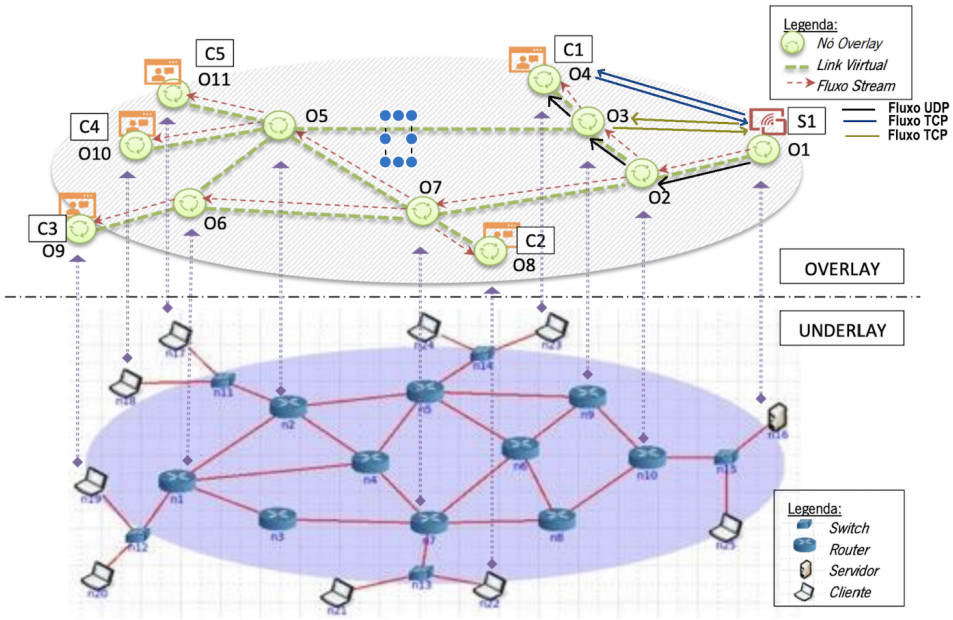

Figura 1. Visão geral de um serviço OTT sobre uma infraestrutura IP

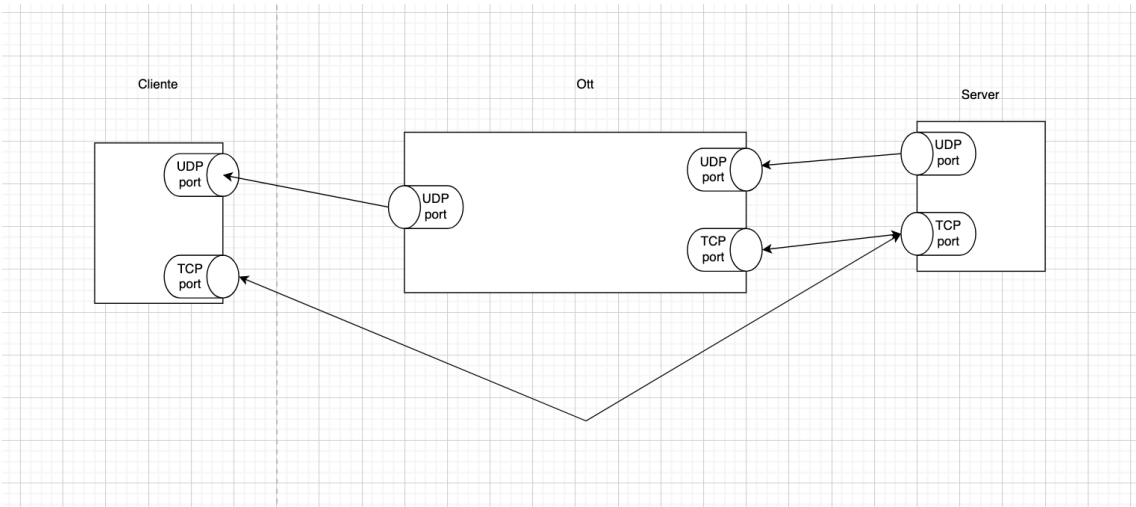

Figura 2. Arquitetura do sistema

2 Especificação dos protocolos

Para conseguir alcançar a solução final foi necessário utilizar e adaptar dois protocolos diferentes, TCP e UDP, ambos com funções distintas no sistema.
Inicialmente iremos abordar o TCP, responsável por todas as comunicações de "sincronização"dentro da rede. De seguida iremos abordar o protocolo UDP, usado, exclusivamente, no envio do conteúdo multimédia.

2.1 TCP

A escolha deste protocolo foi baseada em certas caraterísticas que se mostraram fundamentais à estrutura do projeto, tais como ser orientado à conexão, a sua confiabilidade, entrega ordenada e ser full duplex, permitindo a transferência simultânea em ambas as direções. Assim temos a garantia que todas as comunicações de "sincronização"entre clientes, ott's e servidor não falham, garantindo o funcionamento correto do sistema. Como comunicações de "sincronização"consideramos todas aquelas que são feitas na rede mas que englobam o stream de multimédia. Assim, este protocolo é utilizado para estabelecer dois tipo de conexão.

Cliente - Servidor Tipos de mensagens enviadas pelo cliente:

- Client - informando o server que este cliente está ativo e obrigando-o a calcular novas rotas para os clientes ativos
- Q - informa o servidor que quer encerrar conexão

Tipos de mensagens enviadas pelo servidor:

- reciever - envia ao cliente qual o ip do nodo que lhe vai fornecer o serviço
- -1 - informa o cliente que não se encontra na lista de ips fidedignos
- Q - proíbe o cliente de enviar informação

Ott - Servidor Tipos de mensagens enviadas pelo ott:

- ott - informa o servidor que está ativo e este calcula de novo as rotas incluíndo este ott nas devidas rotas
- Q - informa o servidor que quer encerrar conexão

Tipos de mensagens enviadas pelo servidor:

- data - lista com os vizinhos para os quais o ott deve transmitir o conteúdo recebido
- -1 - informa o cliente que não se encontra na lista de ips fidedignos
- Q - faz com que o ott deixe de receber o conteúdo enviado

2.2 UDP

Este protocolo foi o usado na transmissão de conteúdo multimédia, devido às caraterísticas do serviço de stream pretendido. Devido a este necessitar de fornecer conteúdo em tempo real aos seus utilizadores, optamos por este protocolo devido a mostrar uma menor reação aos efeitos de uma alta latência de rede, suportar broadcast e multicast e por fim não perder tempo com a criação ou destruição de conexões.

Este protocolo é usado tanto pelo cliente, como pelos otts e servidor, de forma a estabelecer as conexões de transmissão. Podemos assistir assim a dois tipo de conexão:
Servidor - Ott: o servidor envia o conteúdo multimédia ao respetivo ott que fica assim na posse do mesmo.
Ott - Ott/Cliente: o ott que recebeu o conteúdo na conexão anterior vai enviá-lo para o respetivo destino, sendo este um ott ou um cliente (destino final do fluxo).

3 Implementação

3.1 Linguagem de programação

Depois de um pequeno debate entre os três membros do grupo chegou-se ao consenso de recorrer à linguagem de programação, Python. Apesar de não representar a linguagem com que o grupo tem maior à vontade, desafiamo-nos a construir a aplicação baseada nela devido à grande facilidade em encontrar soluções às adversidades que possam surgir durante a implementação do serviço e à evolução do conhecimento que proporciona.

3.2 Estratégia de construção do overlay

A estratégia de construção do overlay definida foi uma abordagem baseada em 3 aplicações.
A primeira, um controlador, o servidor, que tem o conhecimento de todo o overlay, aquando da leitura dos ficheiros de configuração, no seu arranque. A segunda, uma aplicação intermediária ao fluxo da transmissão e que possibilita a escalabilidade da mesma, o ott, que envia mensagem de registo, a identificar-se, e vai recebendo como resposta ips, quando este nó é necessário à transmissão de conteúdo. Por fim, o cliente, executa o mesmo processo, que o ott, no envio da mensagem de registo e identificação, pedindo posteriormente o serviço. Quando qualquer nó quer abandonar a execução do programa informa o servidor, que o retira assim da lista de nós ativos, deixando de contar com a presença para de transmissão de conteúdo, até novo registo.

3.3 Cliente

Foi criada uma aplicação própria a ser rodada nos end-systems destinados aos clientes.
Esta encarregava-se de registar o cliente no sistema, enviando uma mensagem TCP ao servidor e de requisitar, de seguida, o serviço fornecido pelo anterior, a stream.
De forma a possibilitar a receção e reprodução do conteúdo, a aplicação estabelece uma conexão UDP com a porta 9999, por onde, através de um ott ou do próprio servidor chega o conteúdo a ser reproduzido. Para que seja possível à aplicação suportar todas estas conexões são utilizadas threads. Assim, uma thread é criada para suportar a conexão TCP e outra é criada para suportar a comunicação UDP. Dentro da thread que suporta a comunicação TCP são criadas mais duas threads, uma que permite o cliente enviar para o servidor e outra que o permite receber do segundo. Já na thread UDP não existe a necessidade de criar threads adicionais uma vez que o cliente apenas recebe conteúdo, o conteúdo da stream, sendo assim apenas uma thread necessária para estabelecer a comunicação.

3.4 Ott

Aplicação inicializada nos routers existentes entre o servidor e os clientes. Responsável por estabelecer conexão TCP e UDP com o servidor e UDP com o cliente. Procedendo de igual modo ao cliente, o ott também faz uso de três threads para possibilitar a conexão TCP. O que difere estas duas aplicações é que o ott recebe por TCP os ips a quem deve enviar o conteúdo. Relativamente à comunicação UDP, também é criada uma thread responsável por receber conteúdo de um ott ou servidor e por enviar o mesmo para outro ott ou o cliente final.

3.5 Server

A aplicação server carateriza-se por ser o cerne do nosso trabalho, pois é ela a responsável por calcular todas as rotas e informar os devidos participantes das mesmas, a partir do conhecimento adquirido da topologia e fornecer o conteúdo para as mesmas, fazendo uso de duas threads, uma para correr as comunicações TCP e outra para as UDP. A nível das comunicações TCP são criadas mais threads, duas para cada cliente/ott que se conecte à rede, sendo uma para enviar e outra para receber mensagens. Mediante a primeira mensagem recebida, o servidor identifica se é cliente ou ott e procede de forma diferente para cada um dos casos, como já descrito na secção relativa ao protocolo. Por fim, a nível da comunicação UDP apenas é utilizada uma thread para envio do conteúdo da stream, que começa trabalhar a partir do momento em que o primeiro cliente se conecta à rede.

3.6 Rotas

Graças ao servidor ter conhecimento do overlay, é possível este ensinar as rotas de fluxo aos restantes nodos da topologia. Através das variáveis globais actives, activesCl, graph, new e routes, o servidor vai mantendo atualizado o estado do grafo representante da topologia e aplicando a função BFS ao grafo, indicando o nó de destino, o servidor vai calcular a rota para o fluxo pretendido, baseada no menor número de saltos da origem até ao destino, iniciando depois um fluxo de transmissão até ao mesmo e informando os intervenientes de forma a começar a começar a transmissão.

4 Testes e resultados

4.1 Topologia para testes e cenários de teste nessa topologia

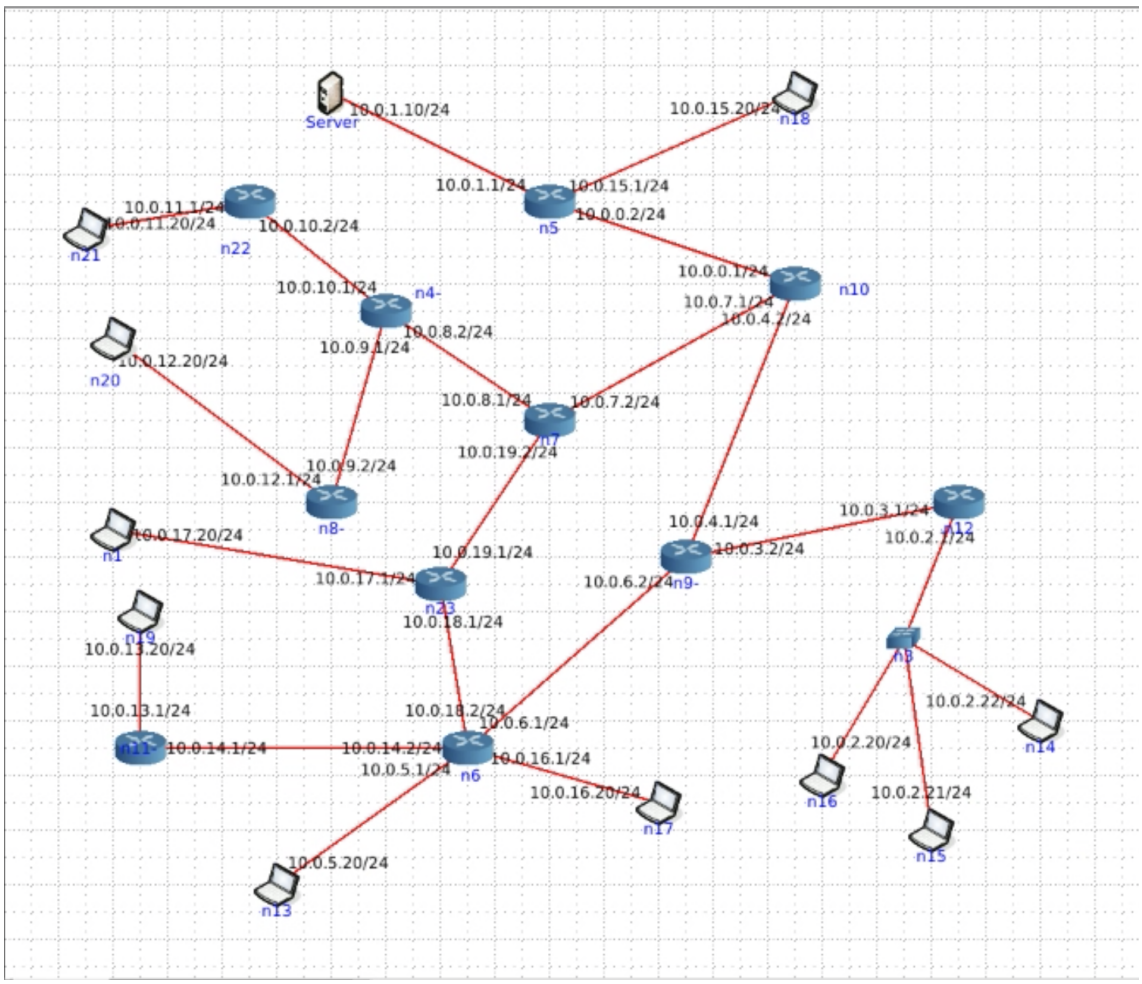

Figura 3. Teste de transmissão com um servidor, um OTT e um cliente ligado

Figura 4. Teste de transmissão com um servidor, um OTT e um cliente ligado

5 Apreciação crítica

5.1 Transmissão de conteúdo

Apesar de o nosso serviço permitir a transmissão de conteúdo, esta ficou um bocado aquém das nossas expectativas, uma vez que o pretendido pelo grupo era enviar conteúdo vídeo. Ressalvamos a descoberta de uma solução que permitia facilmente partilhar vídeos e até mesmo gravação direta da câmara entre o servidor e os clientes mas não nos foi possível a sua implementação uma vez que as bibliotecas por ela usada necessitavam de um espaço de armazenamento superior aquele disponível na Virtual Machine.

5.2 Segurança

Ao longo deste trabalho foi prezada a segurança dos dados em circulação, realizando-se codificação e descodificação das mensagens no momento de envio e receção das mesmas, respetivamente e procedendo à verificação dos ips que se conetavam ao sistema através da consulta da lista de ips fidedignos. Se o ip não se encontra-se registado então o servidor não estabelecia qualquer tipo de conexão com esse end-system.

6 Conclusões e trabalho futuro

O trabalho que nos foi proposto revelou-se um verdadeiro desafio. O nosso sistema mostrou-se capaz de satisfazer os requisitos pretendidos. Apesar da transmissão ter ficado aquém das nossas expectativas, este trabalho permitiu-nos consolidar inúmeros conceitos que relacionados com a área de redes de computadores e não só e também conhecer melhor python. A implementação final foi-se revelando ligeiramente diferente da implementação inicial que tínhamos idealizado, muito devido ao facto de apenas termos compreendido muitos dos pormenores pretendidos aquando do momento implementação. Consideramos que a versão final se mostra mais robusta e fiável em relação aos primeiros esboços por nós "traçados", justificando assim as nossas alterações a meio da construção. Para concluir, esperemos ser capazes de numa futura revisão do projecto corrigir e implementar todos os pormenores que não nos satisfizeram.