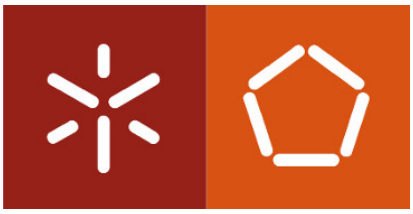

Universidade do Minho

Mestrado em Engenharia Informática

Engenharia de Serviços em Rede

TP2 - Serviço *Over the Top* para entrega de multimédia

Grupo 59

Duarte Parente (PG53791) José Martins (PG53968)

José Moreira (PG53963)

Ano Letivo 2023/2024

Índice

Capítulo 1

Introdução

O presente relatório destina-se a documentar, de maneira abrangente, as diversas etapas e soluções adotadas durante a execução do segundo e último trabalho prático da Unidade Curricular de **Engenharia e Serviços em Rede**. Este trabalho concentrou-se no fascinante domínio do *Streaming Over the Top* (OTT), um tema crucial e contemporâneo no cenário da Engenharia de Redes.

A Unidade Curricular de Engenharia e Serviços em Rede visa proporcionar aos estudantes uma compreensão aprofundada dos princípios fundamentais, protocolos e práticas relacionadas à construção, implementação e manutenção de redes de computadores. No contexto específico deste trabalho prático, a atenção foi direcionada para o OTT, uma abordagem inovadora para a entrega de conteúdo audiovisual através da internet, contornando as tradicionais infraestruturas de distribuição.

O *Streaming Over the Top* tem-se tornado cada vez mais prevalente, impulsionado pelo aumento da largura de banda disponível e pela crescente procura por acesso instantâneo a uma variedade de conteúdos multimédia. Este trabalho prático ofereceu uma oportunidade única para explorar as complexidades envolvidas na implementação de soluções de OTT, compreendendo desde os princípios teóricos até à aplicação prática desses conhecimentos.

Ao longo deste relatório, detalhar-se-á as várias fases do projeto, desde a conceção da arquitetura até à implementação efetiva, destacando os desafios encontrados e as estratégias adotadas para superá-los. Além disso, serão apresentadas análises críticas dos resultados obtidos, considerações sobre possíveis melhorias e reflexões sobre as implicações mais amplas desse trabalho prático no contexto da Engenharia de Serviços em Rede. Este relatório visa, assim, proporcionar uma visão abrangente do processo de desenvolvimento, oferecendo *insights* valiosos para futuros projetos e estudos relacionados ao campo do *Streaming Over the Top*.

Capítulo 2

Arquitetura da Solução

Seguindo a ordem de implementação sugerida pelo corpo docente, foi-se construindo a nossa arquitetura conforme as necessidades que se foram prevendo à medida que se equacionava a solução.

Sendo que estamos perante uma comunicação *peer-to-peer*, a princípio, não fazia sentido dividir as categorias entre cliente e servidor. Criou-se, assim, uma classe mais abstrata que, à *priori*, não assume nenhuma função. Assim, a respetiva função de cada nodo é atribuída no momento de inicialização, ou seja, pelo terminal. Independentemente da sua função, esta passa a conhecer os seus nodos vizinhos, estabelecendo conexão com eles, a partir de **TCP**, mandando uma mensagem para estes e pondo-se à escuta, à espera de mensagens dos seus vizinhos, também.

Desta forma, o sistema encontra-se constituído por quatro tipos de nodos: **cliente**, **servidor**, **RP** e, por fim, **nodo intermédio**, sendo que cada um tem as suas responsabilidades e limitações em relação ao que pode fazer na transmissão e pedidos de vídeo. Assim, detalhando cada uma das classes mais a fundo, o *rendezvous point* tem informação sobre o caminho mais curto para um servidor, com base na latência, sendo esta a diferença em relação aos outros nodos. Um cliente pode fazer um pedido de uma transmissão de vídeo e um servidor pode transmitir o mesmo, quando assim requisitado. Os nodos intermédios apenas herdam as propriedades normais dos nodos *"default"*.

O *rendezvous point* é estático. Foi assim escolhido devido, essencialmente, à estabilidade que este traz à rede, sendo que, uma vez definido, permanece inalterado. Outra das grandes vantagens reside na facilidade de gestão deste método, sendo que se reduz bastante o *overhead* da descoberta dinâmica dele. Assim, o RP tem sempre o conhecimento necessário da sua rede, sem ter que, antes, perceber qual o nodo associado a esta função.

A comunicação entre os nodos presentes na topologia é feita por **TCP** a não ser que seja para a transmissão de um vídeo, neste caso a comunicação entre eles é feita por UDP de forma a termos uma transmissão continua de dados, sem precisar de todas as burocracias associadas ao **TCP**.

Capítulo 3

Especificação dos protocolos

Para a realização deste trabalho prático, recorreu-se aos conhecidos protocolos **TCP** e **UDP**, sendo o primeiro responsável pela troca de mensagens entre nodos e o segundo pela transmissão de stream.

O cliente vai ter a possibilidade de requisitar um vídeo, e esperar pela sua receção. O pedido do vídeo insere-se nas mensagens **TCP**, pois queremos garantir que há passagem de informação o menos falível possível. Quanto à receção de vídeo, como não poderia deixar de ser, é feita por **UDP**.

O servidor vai disponibilizar o vídeo para *stream* quando assim for requisitado. Este irá transmitir o vídeo também, naturalmente, por um canal UDP que irá chegar ao cliente.

O *rendezvous point* pode ser visto como o cérebro da operação, este vai entrar em ação sempre que o cliente pedir um vídeo que não esteja nos seus nodos mais próximos antes do **RP**. O **RP** tem na sua posse o caminho para o cliente, sendo que o sabe pois esta vai no corpo da mensagem, e tem uma lista com os servidores mais rápidos de aceder(menor latência). Este irá portanto comandar a abertura de canais UDP entre os nodos que irão ser precisos para estabelecer a conexão entre o servidor e o cliente de modo a transmitir o vídeo.

Dentro da mensagem **TCP** vai a informação sobre qual é o tipo de pedido da mensagem e como este vai ser tratado.

O identificador **C** está associado ao inicio das conexões entre os nodos, é o que garante que os nodos estão ligados entre eles antes de se poder transmitir mais informação. Já o **RJ** tem um comportamento bastante parecido mas este ocorre quando um cliente que já esteve "vivo"e foi "morto"é ligado de novo. Para informar que o nodo foi "morto"é transmitida uma mensagem **DC**, esta vai fazer com que quem a receba, remova o nodo da lista para a qual os seus vizinhos mandam informação e as ligações entre estes.

A mensagem com o identificador **LT** e **RT** estão fortemente relacionadas. Sendo que o a primeira vai fazer a ligação do **RP** aos servidores, deste modo vai descobrir os caminhos para os servidores com menor latência a partir da resposta (**RT**) do servidor para o **RP**. Facilitando assim a posterior transmissão de vídeo.

A mensagem com o identificador **RS** está associada ao pedido de vídeo, esta faz o pedido em si despoletando também a criação de canais **UDP** e a transmissão do vídeo. Estes processos são

todos tratados depois de estar garantido que existem comunicações entre todos os nodos e que o **RP** sabe o caminho com menor latência para um dos servidores.

Nas mensagens relativas ao controlo de vídeo, tem-se cinco tipos diferentes destas, sendo estas o **PLAY**, o **PAUSE**, o **PLAY_AGAIN** e **QUIT**. O nome destes pacotes é auto-explicativo, sendo que o *play* começa a transmissão tanto para o cliente como para o servidor, se outro cliente clicar no *play* depois do vídeo do servidor já ter começado o vídeo irá aparecer no mesmo momento para os dois. O pause para a transmissão de *frames* do nodo anterior a este para o nodo cliente. O *quit* vai encerrar a transmissão e a conexão entre o cliente e o seu vizinho. Podendo voltar a ligar mais a frente de novo.

De maneira a identificar que o um dos nodos está a fazer *stream*, este vai espalhar uma mensagem pela rede a dar a todos os nodos a informação que este está a transmitir, esta mensagem tem o identificador **SON**. Da mesma forma acontece quando este fecha a *stream*, espalhando assim a mensagem por toda a rede com o identificador **SOFF**.

As mensagens de **ACK** são enviadas como o próprio nome indica, como a confirmação que a mensagem correu como devido.

Por fim a mensagem de **MR**, faz um pedido de *streaming* a um nodo que já esteja a *streamar*, aproveitando assim a informação que já tem sobre ele não precisando de percorrer o percurso todo ate ao servidor para o obter.

Segue-se um resumo num formato tabular dos pacotes TCP usados para troca de mensagens entre os nodos da topologia.

Identificador	**Descrição**
`C`	Conexão inicial
`RJ`	Conexão de um nodo adicional
`DC`	Desconexão de um nodo
`RT`	Resposta ao Teste de Latência
`RS`	Pedido de Vídeo (Floading)
`ACK1`	Acknowledgement a pedido de vídeo (SV - RP)
`ACK2`	Acknowledgement a pedido de vídeo (RP - CL)
`SETUP`	Pedido de Vídeo (RP - SV)
`PLAY`	Pedido de ínicio de stream
`PAUSE`	Pedido de pausa de stream
`PLAY_AGAIN`	Pedido de retoma de stream
`QUIT`	Pedido de cancelamento de stream
`SON`	Anúncio de inicio de stream
`SOFF`	Anúncio de fim de stream
`MR`	Pedido de vídeo a um nodo já em transmissão
`MUL`	Resposta a um pedido de a um nodo já em transmissão

Tabela 3.1: Descrição dos pacotes TCP

De seguida, apresenta-se um diagrama referente ao processo de troca de mensagens, em **flood**, aquando do pedido de stream, por parte de um cliente num momento em que não existem já fluxos de stream para um mesmo pedido. Deste modo, consegue-se observar as principais interações e os pacotes TCP usados entre os diferentes membros envolvidos, bem como pedidos iniciais, respostas a pedidos e por fim o envio de stream.

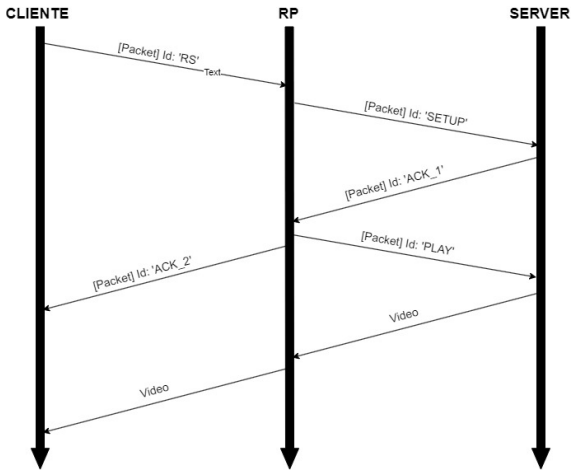

Figura 1: Troca de mensagens em pedido de stream

Capítulo 4

Implementação

A implementação foi feita tendo em conta todas as necessidades do projeto e todas as métricas acima indicadas. Inicialmente, como já foi dito, foi criado um nodo mais geral com as capacidades de ter funções extra em relação aos nodos intermédios, cumprindo assim funções de cliente, servidor e RP.

Ao iniciar o sistema estes nodos são todos iniciados como "ouvintes", à espera da mensagem **TCP** inicial para estabelecerem conexão entre eles. Isto dá ao sistema tempo de abrir todos os nodos antes de começar a enviar mensagens, permitindo a prevenção do erro de um dos nodos mandar uma mensagem para um vizinho seu que ainda não esteja à "escuta".

Depois de confirmada a receção de mensagens entre eles, é tempo de descobrir a menor latência com os servidores. Isto é trabalho do RP, este deve fazer um *flood* à procura de servidores e iniciar um *timer* para ter contabilizar o tempo que demoram as suas mensagens a chegar e a voltar ao remetente, assim, quando precisar de pedir um vídeo a um servidor este irá saber qual o caminho a percorrer que terá menor latência. Estes testes são realizados a cada 500 segundos para manter a lista de caminhos com menor latência atualizada.

O cliente está então em condições de pedir um vídeo a sua topologia. Este irá procurar entre os seus vizinhos e estes, se não o tiverem, irão ter um comportamento igual ao do nodo anterior e mandar o mesmo pedido aos vizinhos. Eventualmente, se o vídeo não for encontrado primeiro, a mensagem atingirá o RP, que irá verificar na sua lista qual o servidor mais rápido de alcançar, estabelece então conexões UDP para trás até ao cliente e para a frente até ao servidor, mandando também o pedido ao servidor por este mesmo caminho mas usando o protocolo **TCP**.

O servidor irá então responder, transmitindo o vídeo pedido por UDP. Este é encriptado pelo servidor para ser mandado, sendo apenas desencriptando no cliente. Deste modo o cliente irá ser servido pelo servidor mas, de uma forma transparente ao cliente, irá passar por nodos intermédios para ter esta conexão.

O acima indicado apenas acontece se o cliente não encontrar nenhum nodo a transmitir o vídeo antes do RP. Se este encontrar algum antes disso vai aproveitar a *stream* desse nodo e usa-lo para o cliente. Se chegar ao RP e este também já estiver a transmitir também aproveita a *stream* deste de uma forma semelhante ao descrito.

Capítulo 5

Limitações da Solução

Este capítulo visa detalhar os diversos fatores que impuseram restrições ao desenvolvimento do nosso projeto, bem como as razões subjacentes a essas limitações. É crucial salientar que as restrições identificadas estão diretamente relacionadas com uma etapa adicional proposta no enunciado do trabalho prático, sendo que tais desafios contribuíram para moldar a abordagem e escopo do projeto.

A principal limitação que merece destaque é a praticamente inexistente tolerância a falhas no nosso sistema. A complexidade inerente à implementação de mecanismos robustos de tolerância a falhas revelou-se um desafio considerável, comprometendo a capacidade do sistema de lidar eficazmente com situações inesperadas, como falhas de hardware, interrupções de rede ou outros eventos adversos. Este défice na tolerância a falhas pode impactar negativamente a fiabilidade e disponibilidade do sistema em ambientes operacionais reais.

Outra limitação notável está relacionada com a impossibilidade de adicionar nodos de forma dinâmica ao sistema. A rigidez nesse aspecto implica que a escala do sistema é estática e não pode ser ajustada conforme as necessidades ou exigências do momento. Esta característica pode limitar a capacidade do sistema de se adaptar a variações na carga de trabalho ou no acréscimo do número de utilizadores, comprometendo a eficiência e flexibilidade do sistema.

A ausência da capacidade de adicionar nodos de forma dinâmica também tem implicações na escalabilidade do sistema, uma vez que não é possível expandir a infraestrutura para acomodar um aumento repentino na procura. Isso pode resultar em sobrecarga e degradação do desempenho, afetando diretamente a qualidade do serviço oferecido aos utilizadores finais.

Por fim, finalizando este capítulo, reforça-se a ideia de que o projeto desenvolvido apresenta as suas expectáveis falhas, falhas essas que comprometem, obviamente, a aplicação da solução a contextos realistas. Por outro lado, afirma-se a importância de combater as várias dificuldades enfrentadas, reforçando a necessidade de adição de tarefas ao capítulo referente ao trabalho futuro.

Capítulo 6

Testes e Resultados

Neste capítulo, serão apresentados os detalhes dos testes realizados no âmbito do projeto, visando avaliar o desempenho, a eficácia e a robustez da solução implementada. A execução de testes sistemáticos é fundamental para validar as escolhas arquiteturais, identificar possíveis falhas e garantir a conformidade com os requisitos estabelecidos.

6.1 Metodologia de Testes

A metodologia adotada compreendeu uma série de testes abrangentes, abordando diferentes aspetos do sistema, em diferentes fases de desenvolvimento do mesmo. Isto permitiu a correção simultânea do código com a escrita do mesmo.

Os vários testes realizados permitiram que o projeto fosse avaliado em vários parâmetros, desde controlo de qualidade a controlo de erros. Com isto, passou a ser possível identificar, mais facilmente, as falhas do projeto, podendo, assim, reforçar a ideia de trabalho futuro a implementar.

6.2 Resultados Obtidos

Os resultados dos testes oferecem uma visão abrangente do desempenho do sistema em diferentes cenários. Os dados observados permitiram avaliar a eficácia das estratégias implementadas para lidar com os desafios identificados. Além disso, possibilitaram a identificação de áreas de melhoria e ajustes necessários para otimização o sistema e correção de erros críticos.

De seguida, apresenta-se duas imagens que ilustram a topologia final usada para testes, bem como o serviço de **streaming** possibilitado pela nossa solução.

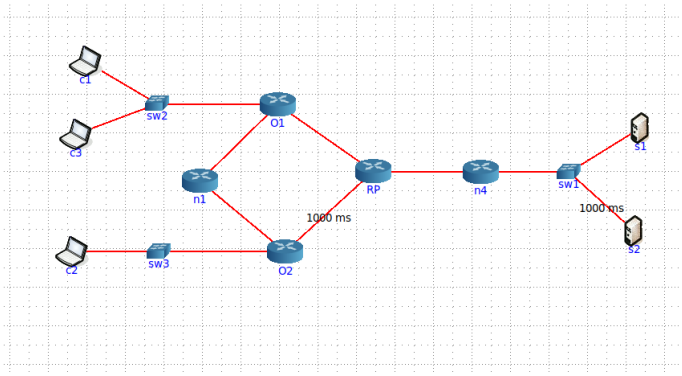

Figura 2: Topologia

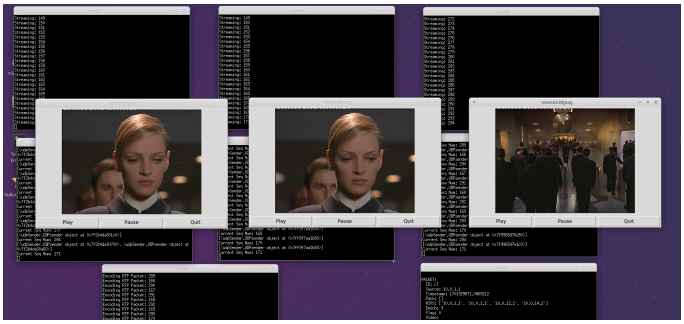

Figura 3: Serviço de streaming

Capítulo 7

Conclusões e Trabalho Futuro

Terminando o presente relatório, torna-se evidente a complexidade e os problemas envolvidas no desenvolvimento do projeto prático relacionado ao tema *Streaming Over the Top*, na Unidade Curricular de **Engenharia e Serviços em Rede**. O processo de conceção, implementação e teste proporcionou uma valiosa aprendizagem sobre os desafios inerentes à entrega de conteúdo audiovisual através da internet, destacando as limitações enfrentadas e as estratégias adotadas para superá-las.

A abordagem adotada na implementação do projeto revelou-se eficaz em muitos aspetos, permitindo a transmissão de conteúdo de forma eficiente e garantindo uma experiência satisfatória ao utilizador final. No entanto, as limitações identificadas, como a falta de tolerância a falhas e a incapacidade de adicionar nodos de forma dinâmica, apontam para áreas que necessitam de atenção e melhorias.

Os testes realizados desempenharam um papel crucial na validação do sistema, oferecendo *insights* significativos sobre o desempenho, resiliência e escalabilidade da solução. As análises dos resultados permitiram identificar pontos fortes, como a eficiência na transmissão de dados, ao mesmo tempo em que destacaram áreas que necessitam de melhorias, especialmente no que diz respeito à tolerância a falhas e escalabilidade.

Este projeto prático não apenas consolidou os conhecimentos teóricos adquiridos ao longo da Unidade Curricular, mas também proporcionou uma oportunidade valiosa para a aplicação prática desses conhecimentos num contexto relevante e desafiador. A compreensão das limitações enfrentadas e a reflexão sobre os resultados obtidos servem como base para futuras iterações do projeto e para o aprimoramento contínuo da solução.

À medida que avançamos na era digital, onde a entrega de conteúdo multimédia continua a desempenhar um papel central, a experiência obtida neste projeto contribuiu para uma visão mais abrangente sobre as necessidades e complexidades associadas ao *Streaming Over the Top*. Deste modo, encerra-se o presente relatório relativo ao segundo projeto desenvolvido no âmbito da Unidade Curricular de **Engenharia e Serviços em Rede**.