ESR - PL2 - Grupo 20

Gonçalo Gonçalves[1[PG55945]], Henrique Vaz[1[PG55947]], and Mike Pinto[1[PG55987]]

[1] Universidade do Minho, Braga, Portugal

Abstract. Este trabalho prático, desenvolvido no âmbito da disciplina de Engenharia de Serviços em Rede na Universidade do Minho, propõe-se a conceber e prototipar um serviço Over the Top (OTT) destinado à entrega eficiente de multimédia. Utilizando o emulador CORE como plataforma de teste, o projeto visa criar uma rede overlay para a transmissão em tempo real de vídeo com Over the Top na otimização de recursos e na melhoria da qualidade de experiência do utilizador.

Keywords: Streaming de vídeo · Rede Overlay

1 Introdução

Este relatório apresenta o design, implementação e avaliação de um protótipo de serviço Over-the-Top (OTT) para entrega de conteúdos multimédia, com foco na otimização de recursos e melhoria da experiência do utilizador. A solução proposta utiliza uma rede overlay na camada de aplicação para superar as limitações inerentes das infraestruturas IP na gestão de streaming multimédia em tempo real. Com recurso ao emulador CORE, foi desenvolvida uma topologia de Rede de Entrega de Conteúdos (CDN), integrando redes underlay e overlay para uma gestão eficiente da entrega de vídeos de servidores para clientes.

O projeto incluiu a criação de nós na rede overlay, estruturados dinamicamente em árvores de distribuição, otimizando a distribuição de conteúdos. Os nós utilizam protocolos de transporte (TCP/UDP) para manter conectividade robusta, garantindo baixa latência e perdas mínimas. A solução também priorizou a otimização dinâmica de rotas, monitorização da rede e mecanismos de recuperação de falhas para manter um streaming de alta qualidade, mesmo em condições de rede variáveis.

As avaliações experimentais demonstraram a capacidade do sistema de se adaptar dinamicamente a mudanças na rede, minimizar o uso de largura de banda e oferecer acesso contínuo aos conteúdos multimédia. Os resultados destacam o potencial das soluções baseadas em overlay para resolver desafios de escalabilidade e desempenho em sistemas de entrega de conteúdos em tempo real. Trabalhos futuros

incluem a extensão do protótipo para suportar múltiplos servidores e a implementação de estratégias avançadas de tolerância a falhas.

2 Arquitetura da solução

A solução desenvolvida é composta por diferentes tipos de nós que desempenham papéis específicos para garantir o funcionamento eficiente e escalável do sistema de entrega de conteúdos multimédia Over-the-Top (OTT). A seguir, descreve-se a função de cada um desses nós dentro da rede:

2.1 Servidor de Conteúdos

O servidor de conteúdos é a origem tanto dos fluxos de multimédia (video) como dos fluxos que permitirão calcular a latência em todos os nós (enviando um timestamp) permitindo então a criação e manutenção da àrvore de distribuição. O servidor é responsável por:

- Servir múltiplos vídeos simultaneamente.
- Iniciar o processo de geração as árvores de distribuição com base na latência (enviando o primeiro timestamp que será depois propagado por toda a rede de overlay).
- Enviar os fluxos de vídeo para nós da rede overlay que são seus vizinhos.

2.2 Nós da Rede Overlay (Overlay Nodes)

Os nós da rede overlay são responsáveis pela formação da estrutura lógica que liga o servidor aos pontos de presença (PoPs). Formam uma rede de distribuição baseada em árvores multicast, encaminhando os fluxos de dados de forma eficiente. As suas principais funções incluem:

- Construir e manter tabelas de rotas dinamicas para os diferentes conteudos.
- Reencaminhar pacotes de dados para outros nós ou para os PoPs.
- Reencaminhar pacotes de timestamps oriundos do servidor para os seus vizinhos (nós ou PoPs).
- Enviar mensagens periodicas para o nó ou servidor de quem estão a receber a *stream* de forma a possibilitar a gestão de perdas de nós/ligações.
- Garantir a redundância, proporcionando caminhos alternativos em caso de falhas na rede.

2.3 Pontos de Presença (PoPs)

Os pontos de presença (PoPs) são os nós de ligação entre a rede overlay e os clientes finais. Diferente dos nós de overlay e dos servidores de conteúdo os PoPs não têm vizinhos, recebem mensagens dos nós de overlay para calculo da latência mas apenas atualizam as suas tabelas, não enviando estes pacotes para mais ninguém. Quando um cliente pede informação da latência é enviada a latência do servidor de conteúdos

até ao PoP junto com um novo timestamp para que o cliente consiga calcular a latência em toda a rede. As suas principais funções incluem:

- Determinar as melhores rotas na rede overlay baseadas na latência.
- Monitorizar as condições de rede e ajustar as conexões com base nas métricas observadas.
- Servir como ponto de acesso principal para os clientes, garantindo uma entrega rápida e eficiente dos conteúdos.
- Enviar mensagens periodicas para o nó de quem estão a receber a *stream* de forma a possibilitar a gestão de perdas de nós/ligações.

2.4 Clientes

Os clientes (oClient) são os destinatários finais dos conteúdos multimédia. Cada cliente é responsável por solicitar e consumir o conteúdo oferecido pela rede. As suas principais características são:

- Estabelecer conexões com o PoP mais eficiente com base na latência.
- Solicitar o video pretendido ao PoP.
- Reproduzir os conteúdos recebidos, assegurando uma experiência de qualidade para o utilizador final.
- Adaptar-se dinamicamente a mudanças na rede, mudando de PoP se necessário.

2.5 Bootstrapper

O bootstrapper desempenha um papel fundamental na inicialização da rede overlay. É responsável por fornecer aos nós da rede as informações necessárias para que estabeleçam conexões com seus vizinhos. As suas funções incluem:

- Receber pedidos de nós que desejam ingressar na rede.
- Consultar um ficheiro de configuração (JSON) para determinar os vizinhos do nó solicitante com base no seu endereço IP.
- Enviar as informações dos vizinhos para o nó solicitante, permitindo a formação da rede overlay.
- Impedir o arranque de nós mal configurados, retornando uma mensagem de erro caso não existam vizinhos definidos.

3 Especificação dos protocolos

O protocolo TCP, reconhecido pelo seu serviço orientado à conexão, destaca-se por assegurar a entrega ordenada e confiável dos dados, mediante a retransmissão de pacotes perdidos, a deteção e a correção de erros. Este é especialmente indicado para aplicações que são sensíveis à perda de dados e exigem garantias de entrega. No contexto deste projeto, o TCP é usado exclusivamente para as mensagens de controlo entre os PoPs, nós do overlay e servidor de conteúdos. Em contraste, todas as transmissões de vídeo adotam o protocolo UDP. Esta escolha é motivada pela natureza do streaming de vídeo, onde a perda ocasional de pacotes não é considerada um problema crítico. Isto alinha-se com as características do UDP, que oferece uma menor latência e é mais tolerante à perda de pacotes, sendo uma escolha mais apropriada para fluxos contínuos de dados, como é o nosso caso. Além disso, todas as comunicações entre os clientes e os pontos de presença acontecem em UDP como indicado no enunciado deste projeto. Esta abordagem de usar diferentes protocolos para diferentes finalidades é uma pratica comum em projetos de redes e sistemas, onde a escolha de protocolo é adaptada ás necessidades específicas de cada tipo de comunicação. O TCP é utilizado onde a confiabilidade é crucial, enquanto o UDP é adotado para otimizar a eficiência em transmissões de dados menos sensíveis à perda.

3.1 Formato das mensagens protocolares

De forma a simplificar a diferenciação entre diferentes tipos de mensagens foram escolhidas portas fixas para determinados tipos de mensagens, sendo estas:

- Streaming Port = 12346: Porta onde serão enviados e recebidos todos os dados da stream em UDP.
- Control Port = 13333: Porta onde serão enviadas e recebidas todas as mensagens de controlo (UDP entre cliente e PoP e TCP entre os restantes nós).
- Heartbeat Port = 22222: Porta onde serão enviadas mensagens períodicas para testar se as ligações/nós estão “vivos” (UDP entre cliente e PoP e TCP entre os restantes nós).
- Timestamp Port = 13334: Porta onde serão enviados e recebidos os pacotes de timestamp entre os nós de overlay, PoPs e servidores de conteúdos (TCP).
- LatencyRequest Port= 13335: Porta onde serão enviados pedidos e respostas de informação de latência entre os clientes e os PoPs (UDP).

Pacotes de vídeo

Os pacotes de video serão sempre enviados/recebidos na porta 12346 tal como referido acima, estes pacotes têm o seguinte conteúdo:

- video_id: Id do video.
- packet_id: Número de sequência do pacote.
- frame_size: Tamanho do frame em bytes.
- chunk: Dados do frame.

Pacotes de controlo

Os pacotes de controlo serão sempre enviados/recebidos na porta 13333, estes pacotes têm a função de pedir o começo/fim da stream de um dado video para um dado cliente, podendo ter origem no cliente (quando este pretende trocar para um PoP com melhor latência pede ao PoP a que está ligado para terminar a stream antes de mudar para o novo), podem também ter origem nos nós do overlay/PoPs (quando há um melhor caminho para a stream pedem ao nó do qual estão a receber atualmente para terminar antes de pedir ao novo nó para começar a transmitir). Quando um nó recebe informação para terminar uma stream verifica sempre se têm mais algum cliente interessado nessa stream, caso não tenha o pacote é propagado na direção do servidor para que não existam fluxos desnecessários. A falha de um cliente a enviar Heartbeats também faz com que os nós verifiquem se têm mais clientes interessados na stream, parando de enviar para o cliente que está incontactável.

Estes pacotes têm o seguinte formato:

- mensagem: "START_STREAM" ou "STOP_STREAM".
- video: Id do video que quer receber ou parar de receber.

Pacotes de Heartbeat

Estes pacotes têm como objetivo a possibilidade de parar a stream quando algum cliente ou nó deixa de estar contactável. Se os nós que estão a enviar a stream (nós overlay/PoP/Servidor de conteúdos) não receberem um heartbeat dos seus clientes no tempo especificado então removem-nos da sua lista de clientes ativos.

Estes pacotes têm o seguinte formato:

- mensagem: "HEARTBEAT"

Pacotes de timestamp

Estes pacotes têm origem no servidor. O seu propósito é permitirem o cálculo da latência entre todos os nós (excepto entre os clientes e os PoPs). O servidor de conteúdos envia pacotes deste tipo periodicamente para os seus vizinhos que por

sua vez os propagam imediatamente para os seus vizinhos até que cheguem aos PoPs. Junto destas mensagens é também enviada a informaçãao de todos os videos disponiveis para que posteriormente possam existir pedidos sobre os mesmos.

Estes pacotes têm o seguinte formato:

- timestamp: Timestamp original criado pelo servidor.
- video_list: Lista dos Ids dos videos disponíveis.

Pacotes de Latency Request

Como um cliente se pode querer ligar a qualquer momento é necessário que os PoPs disponibilizem uma forma dinâmica para o cálculo da latência desde os clientes até ao servidor de conteúdos, para isso as mensagens deste tipo irão conter informação da latência do PoP até ao servidor de conteúdos e também um timestamp gerado pelo PoP, permitindo dessa forma o cálculo da latência em toda a rede.

Estes pacotes têm o seguinte formato:

- mensagem: "LATENCY_REQUEST"
- latency_sent: Latência do PoP até ao servidor de conteúdos.
- sent_time: Timestamp gerado pelo PoP.
- available_videos: Lista dos vídeos disponíveis.

3.2 Interações

O sistema desenvolvido permite um número de possíveis interações entre os diversos componentes integrados na topologia da rede, sendo algumas das mais pertinentes as que iremos descrever.

Quando arrancam, todos os nós de overlay e servidores de conteúdo têm obrigatoriamente de obter a informação dos seus vizinhos do bootstrapper.

Após isso, o servidor de conteúdos envia para os seus vizinhos um timestamp que será propagado por toda a rede até chegar aos PoPs.

Quando um cliente pretende ver uma stream é enviado um pedido para todos os PoPs para ser feita a medição da latência, é daí também que recebe a informação dos videos que têm disponíveis.

Após ter informação de latência até aos PoPs, o cliente escolhe o melhor PoP e pede para ser iniciada a stream, este pedido será propagado pelo PoP no caso de não ter já nenhum cliente a consumir essa stream, o mesmo é feito pelos restantes nós até chegar ao servidor de conteúdo para que seja simulado o multicast no nível aplicacional.

A partir do momento em que há streams ativas há também a propagação de pacotes de heartbeat no sentido dos clientes para o servidor de conteúdos, para que seja possível parar streams para clientes/nós que tenham falhas.

4 Implementação

Neste capítulo, será detalhada a implementação dos principais tipos de nós que compõem a solução:

Clientes (oClient): Os destinatários finais do conteúdo multimédia, responsáveis pela receção e reprodução dos fluxos enviados pela rede.

Pontos de Presença (PoPs): Interfaces estratégicas entre a rede overlay e os clientes, encarregados de otimizar a entrega.

Nós da Rede Overlay (Overlay Nodes): Elementos que estruturam a camada lógica da rede, responsáveis pelo encaminhamento eficiente de dados.

Servidores (Servers): Origem dos conteúdos multimédia e núcleo central para a geração de árvores de distribuição na rede

Bootstrapper: Um nó especial, utilizado para inicializar a rede overlay, facilitando a entrada de novos nós e garantindo a conectividade inicial.

Para o desenvolvimento de todo o trabalho foi selecionada a linguagem de programação python.

4.1 Bootstrapper

O *bootstrapper* tem como principal propósito indicar aos nós da rede *overlay* quais são os seus vizinhos iniciais, permitindo a formação da estrutura lógica da rede. Para cumprir essa função, o *bootstrapper* cria uma *thread* dedicada para cada ligação recebida.

Quando um servidor ou nó *overlay* é iniciado, este envia um pedido ao *bootstrapper*. O *bootstrapper*, por sua vez, lê um ficheiro *JSON* previamente configurado, onde está definida a lista de vizinhos para cada nó. Com base no endereço IP do nó que fez o pedido, o *bootstrapper* procura no *JSON* pelos vizinhos correspondentes. Caso existam vizinhos configurados para o nó em questão, os seus endereços são enviados de volta.

Se, no entanto, não forem encontrados vizinhos, o *bootstrapper* retorna uma mensagem de erro. Nesse caso, o nó que fez o pedido é incapaz de arrancar, assegurando que apenas nós corretamente configurados possam integrar a rede overlay.

Para arrancar o bootstrapper:

python3 bootstrapper.py –file <json de vizinhos>

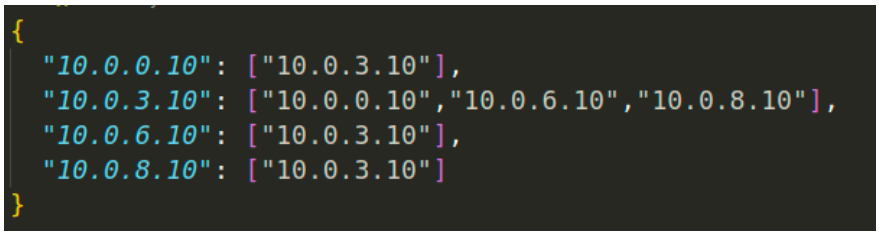

Fig. 1. Exemplo de um ficheiro com os vizinhos

4.2 Servidor de conteúdo

O servidor de conteúdos é o único tipo de nó que têm acesso direto aos ficheiros de video. Nestes servidores é criada uma thread para cada video existente para que os videos avancem mesmo sem clientes a consumirem. Há também uma thread responsável por enviar Timestamps para os vizinhos a cada 10 segundos (foi considerado o valor ideal para nem causar demasiado overhead na rede nem demorar demasiado tempo a recuperar de falhas). É também criada uma thread por cada ligação recebida para pedidos de video. O servidor mantém também uma lista de clientes para cada video pelo que consegue desligar a stream para qualquer cliente que falhe a enviar um heartbeat a cada 6 segundos.

Para arrancar o servidor de conteúdos:

python3 main.py –ip <IP do bootstrapper> –video video1.mp4 video2.mp4 ....

4.3 Nó de overlay

Os nós de overlay têm a função de reencaminhar as streams dos diferentes videos, podendo esta ser recebida tanto de outro nó de overlay como diretamente do servidor de conteúdos, para isso estes nós ficam à espera de receber timestamps, quando recebem atualizam uma tabela de latências para os nós que começaram a conexão, reencaminham o melhor timestamp para os seus vizinhos e escolhem a melhor opção para pedir a stream. Após isso ficam à espera de receberem pedidos de streams, ao receberem mais pedidos verificam sempre se já têm acesso à stream desse video de modo a evitar a criação de fluxos desnecessários na rede. Se um cliente de uma dada stream falhar o envio de um heartbeat a cada 6 segundos a stream é desligada para esse cliente. Enquanto o nó de overlay estiver a receber streams de outro nó envia-lhe heartbeats a cada 2 segundos para mostrar que está “vivo” e contactável. Quando o último cliente de uma dada stream é removido os nós de overlay enviam um pedido para terminar a stream para o nó do qual são clientes.

Para arrancar o nó de overlay:

python3 main.py –ip <IP do bootstrapper>

4.4 Point of presence

Os PoPs são o ponto de acesso dos clientes às streams existentes. Ao arrancarem os PoPs ficam à espera de receberem timestamps provenientes de nós de overlay que os têm como vizinhos, à medida que recebem estes timestamps vão criando uma tabela com latências para que, quando um cliente faça um pedido de medição lhes seja enviada a melhor latencia existente, ficando depois ao critério do cliente escolher o melhor PoP. Todas as interações com os clientes são em UDP. Existe também nos PoPs uma listagem de todos os clientes existentes para cada video, para que, no caso de não receberem heartbeats a cada 6s de todos os clientes a stream lhes seja parada. Enquanto o PoP estiver a receber streams de outro nó envia-lhe heartbeats a cada 2 segundos para mostrar que está "vivo" e contactável. Quando o último cliente de uma dada stream é removido os PoPs também enviam um pedido para terminar a stream para o nó do qual são clientes.

Para arrancar o PoP:

python3 main.py

4.5 Cliente

Os clientes são o ponto onde será visualizada a stream. Ao arrancar o cliente pede a todos os PoPs que conhece uma medição da latência que será usada para escolher o melhor PoP, estas medições serão depois feitas a cada 10 segundos de forma a manter a melhor qualidade possível. Assim que começa a receber uma stream o cliente envia heartbeats a cada 2s para mostrar ao nó que lhe está a enviar a stream que está "vivo" e contactável.

Para arrancar o cliente:

python3 main.py –ip <IP PoP 1> <IP PoP 2> ...

5 Testes e resultados

5.1 Topologia de testes

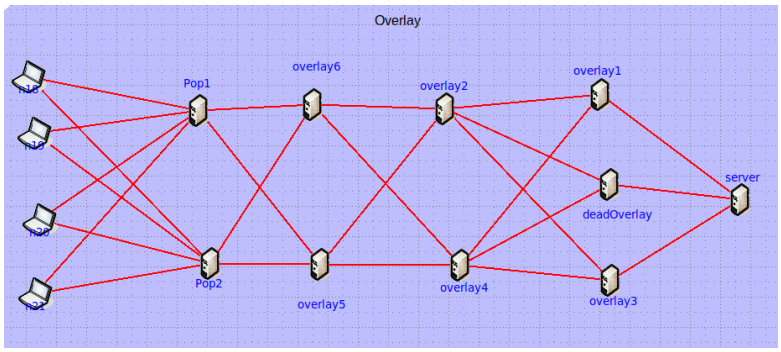

Fig. 2. Topologia overlay

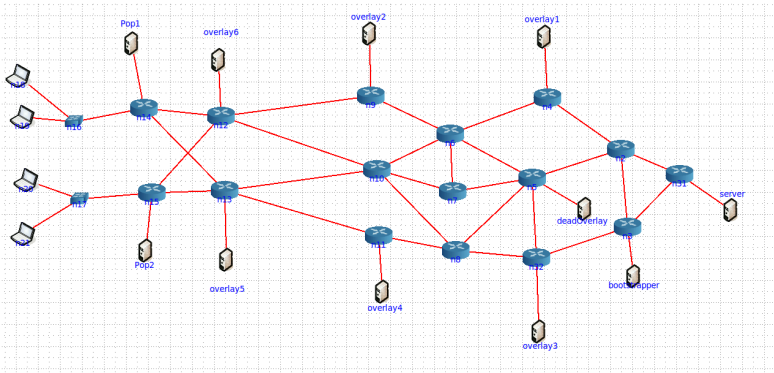

Fig. 3. Topologia underlay

A topologia desenhada pelo grupo foi desenhada com o intuito de conter caminhos alternativos entre todos os nós para que possa ser testada a capacidade da rede resistir a perdas de nós/ligações. Existe também um nó de overlay chamado "dead-Overlay" que será utilizado para testar o cenário de adição de um nó enquanto a rede está em funcionamento.

5.2 Testes

Após a implementação da nossa aplicação, foi possível testar e verificar com a topologia ilustrada nas figuras acima a possibilidade de clientes assistirem a múltiplas streams de videos diferentes simultaneamente, é importante notar que, durante todo o tempo que os clientes não estão a visualizar uma stream os videos continuam a avançar no servidor.

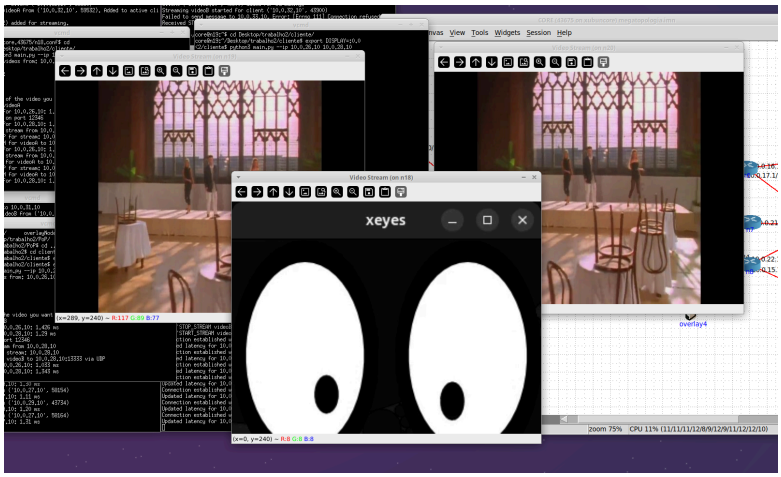

Fig. 4. 3 clientes a assistirem a streams de dois vídeos distintos

Foram também testados os seguintes cenários que serão depois demonstrados na defesa deste trabalho por uma questão de simplicidade:

- Cliente alternar entre PoPs quando existem aleternativas melhores (Monitorização da Rede Overlay).
- PoPs e nós de overlay a alternarem as suas rotas de acordo com variações na rede (Monitorização da Rede Overlay).
- Perda de nós de overlay/PoPs com recuperação total da rede (Definição do método de recuperação de falhas).
- Adição de nós na rede fazendo com que a árvore de distribuição os inclua.

6 Conclusão

Este trabalho prático alcançou o objetivo principal de conceber e implementar um sistema Over-the-Top (OTT) eficiente para a transmissão de conteúdos multimédia, utilizando uma rede overlay. A abordagem desenvolvida demonstrou ser capaz de otimizar a entrega de vídeo em tempo real, minimizando a latência, ajustando-se dinamicamente às condições da rede e garantindo uma experiência consistente para os utilizadores.

A implementação utilizou uma estrutura modular, composta por diferentes tipos de nós, incluindo servidores de conteúdos, nós de overlay, pontos de presença (PoPs) e clientes. Cada componente desempenhou um papel crucial na formação de uma rede robusta e escalável. A utilização de protocolos TCP e UDP foi estrategicamente ajustada para equilibrar confiabilidade e eficiência, enquanto mecanismos como mensagens de heartbeat e timestamps asseguraram a continuidade dos fluxos e a recuperação de falhas.

Os testes realizados confirmaram a capacidade do sistema de gerenciar múltiplas streams simultâneas, ajustar rotas de transmissão e integrar novos nós de maneira dinâmica, destacando a flexibilidade e resiliência da solução. Cenários como perdas de nós ou alterações na topologia foram tratados com sucesso, reforçando a viabilidade do sistema em ambientes reais.

Como trabalho futuro, sugere-se a ampliação do protótipo para suportar a integração de algoritmos avançados de routing e load balancing para optimização adaptativa.

Em resumo, este projeto não só demonstrou a aplicabilidade de redes overlay para soluções de streaming em tempo real, como também contribuiu para um desenvolvimento de conhecimento mais profundo nas dinâmicas envolvidas na entrega eficiente de conteúdos multimédia.