Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Engenharia de Serviços em Rede

Ano Letivo de 2021/2022

Serviço Over the Top para entrega de multimédia

João Amorim, A74806
Tiago Matos, PG45585
Gonçalo Costeira, PG47219

30 de dezembro de 2021

ERS

Índice

1 Introdução

O consumo de vídeos em live, seja através das redes sociais, Facebook ou Instagram, ou através de plataformas como Youtube ou Twich, assim como o consumo de vídeo-on-demand em plataformas de streaming como Netflix ou HBO, tornou-se numa nova realidade no nosso dia-a-dia, levando a uma mudança de paradigma no consumo de vídeo dos utilizadores e consequentemente nos prestadores de serviços.

Este tipo de consumo de conteúdo ponto-a ponto, gera elevados volumes de tráfego, causando dificuldades ao próprio prestador de serviços (ISP) garantir a qualidade na entrega deste conteúdo ao cliente final. De forma a colmatar esta dificuldade, este tipo de tráfego passou a ser entregue através de CDNs (Content Delivery Network) e de serviços projetados em cima da rede IP pública, através de uma rede overlay, chamados de serviços Over-the-Top (OTT).

O objetivo é a implementação de uma rede overlay aplicacional em cima de uma rede IP, para partilha de conteúdos desde o servidor de conteúdo até aos vários clientes da rede. Esta solução deve ser escalável e para tal teremos vários nós overlay para transmissão dos dados, tendo em consideração a otimização de rotas entre os nós e os clientes. Como ponto final efetuar um streaming desde o servidor até ao cliente.

2 Arquitetura da Solução

Após uma primeira análise do problema e das várias soluções propostas, para conseguirmos uma solução que permitisse o cálculo de rotas, a transmissão de informação entre os diferentes agentes e um encaminhamento de pacotes eficiente, decidimos utilizar o protocolo UDP para toda a comunicação entre os agentes do sistema, visto que nos permite implementar o Real Time Protocol em cima de UDP (procedimento que será explicado na próxima secção). Para a construção da topologia overlay optamos seguir a estratégia manual, em que ao executar o programa são indicados os N vizinhos do nodo, construindo assim a rede overlay. De seguida para a construção das rotas a métrica utilizada como custo foi o número de saltos e o tempo de atraso (no caso do número de saltos ser o mesmo), que por iniciativa do servidor através da interface gráfica é possível recalcular as rotas quando um nodo é adicionado ou removido.

O nosso sistema é então composto por 3 agentes, o **Servidor**, o nodo **Overlay** e o **Cliente**.

O Servidor é responsável por 3 processos:

- **[1]** disponibilizar uma interface gráfica que permite iniciar os processos **[2]** e **[3]**;
- **[2]** responsável por enviar uma mensagem, para o próximo nodo, de forma a iniciar o cálculo das novas rotas;
- **[3]** responsável por iniciar a stream do vídeo.

O nodo Overlay é responsável por 3 processos:

- **[4]** disponibilizar uma interface gráfica que permite iniciar o **[6]** e parar a transmissão;
- **[5]** responsável por enviar uma mensagem para todos os vizinhos para recalcular as rotas e atualiza-las;
- **[6]** responsável por encaminhar o stream os clientes.

O Cliente é responsável por 2 processos:

- **[7]** disponibilizar uma interface gráfica que permite iniciar e terminar a receção da stream;
- **[8]** responsável por enviar uma mensagem para a rede overlay a indicar que é um cliente e adiciona-lo à rota.

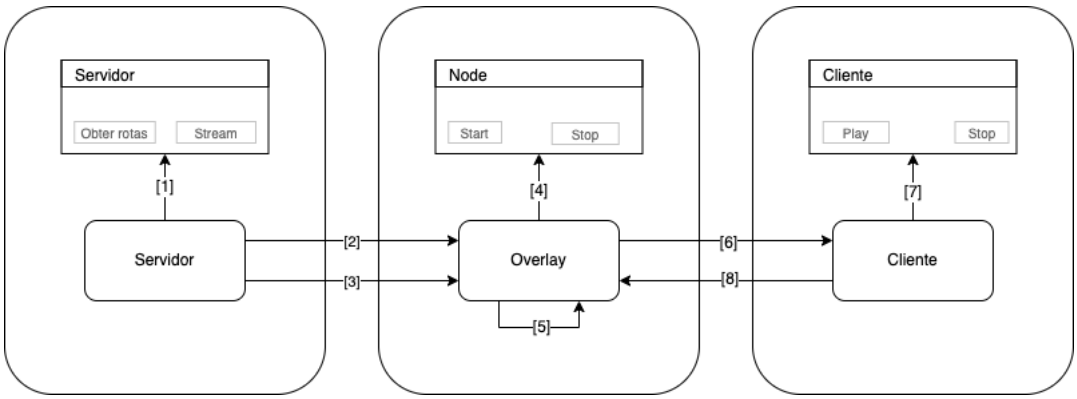

Figura 2.1: Arquitetura da solução

3 Especificação do(s) Protocolo(s)

A nível de protocolos, utilizamos o Real Time Protocol (RTP) em cima de UDP como protocolo de transporte. O RTP é utilizado para comunicações end-to-end em tempo real de dados de vídeo ou áudio, no nosso caso utilizamos o formato de vídeo Mjpeg (Payload Type 26). Foi definido o tamanho do header do RTP de 12 bytes, tendo os seguintes campos abaixo:

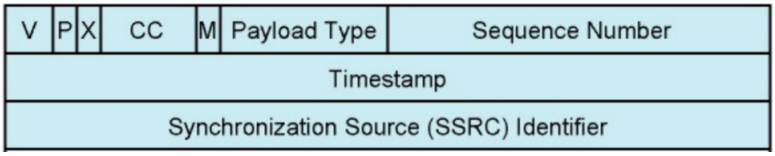

Figura 3.1: RTP Packet Header

O servidor encapsula o chunk num packet RTP, em seguida é encapsulado em UDP e enviado para a rede IP. O Cliente efetua o processo inverso, extrair o packet RTP do packet UDP e posteriormente extrai o vídeo e reproduz na aplicação de vídeo. Nos vários pontos da rede overlay conseguimos visualizar as mensagens RTP enviadas/recebidas conforme imagem abaixo:

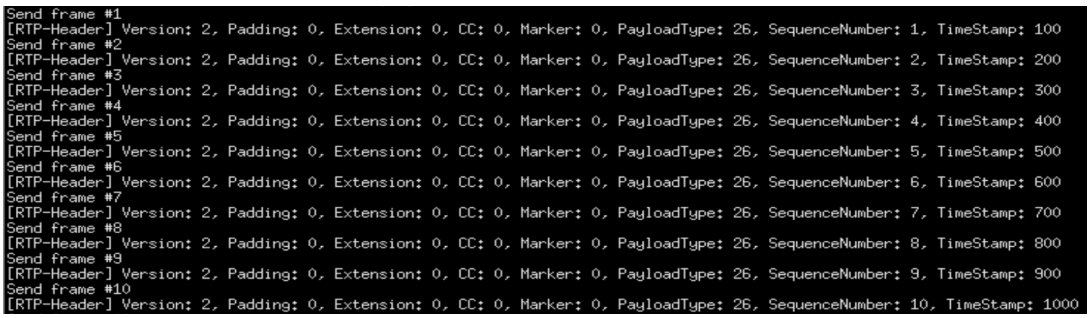

Figura 3.2: RTP Packet Servidor

Como verificado o RTP requer apenas a entrega de informação em tempo-real, tolerando alguma perda de pacotes. No TCP uma das características é a confiabilidade de entrega de todos os pacotes na sequencia correta o que pode causar atrasos avultados na entrega de trafego em real time, ao contrário do UDP, utilizado na maior parte de implementações do RTP.

Tipicamente as portas utilizadas para o UDP variam entre 1024 e 65535, como cenário de teste foi configurada a porta 3333 no Nó de Overlay (virado para o Servidor) e as portas 4444 para o Nó de Overlay e os Clientes.

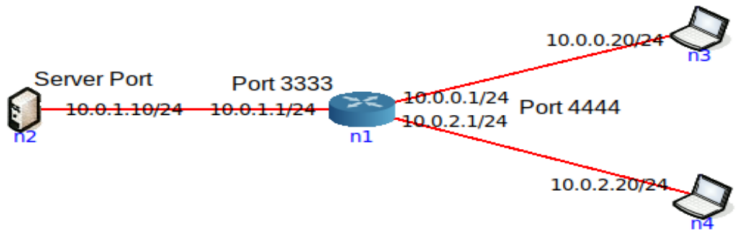

Figura 3.3: Cenário Teste 1

Podemos confirmar nas capturas do wireshark no Servidor e num dos Clientes a comunicação end-to-end utilizando UDP.

No.	Time	Source	Destination	Protocol	Length	Info
16	17.206906456	10.0.1.10	10.0.2.1	UDP	148	38160 → 3333 Len=6026
21	17.866237567	10.0.1.10	10.0.2.1	UDP	74	38160 → 3333 Len=5952
26	17.961896988	10.0.1.10	10.0.2.1	UDP	59	38160 → 3333 Len=5937
31	18.186493725	10.0.1.10	10.0.2.1	UDP	1474	38160 → 3333 Len=5872
35	18.285965900	10.0.1.10	10.0.2.1	UDP	1385	38160 → 3333 Len=5783
39	18.384169606	10.0.1.10	10.0.2.1	UDP	1303	38160 → 3333 Len=5701
43	18.489668284	10.0.1.10	10.0.2.1	UDP	1103	38160 → 3333 Len=5501
47	18.588295745	10.0.1.10	10.0.2.1	UDP	927	38160 → 3333 Len=5325
51	18.701826444	10.0.1.10	10.0.2.1	UDP	833	38160 → 3333 Len=5231
55	18.809892714	10.0.1.10	10.0.2.1	UDP	707	38160 → 3333 Len=5105
59	18.912148747	10.0.1.10	10.0.2.1	UDP	614	38160 → 3333 Len=5012
63	19.016309732	10.0.1.10	10.0.2.1	UDP	584	38160 → 3333 Len=4982
67	19.121156281	10.0.1.10	10.0.2.1	UDP	598	38160 → 3333 Len=4996
71	19.221784503	10.0.1.10	10.0.2.1	UDP	699	38160 → 3333 Len=5097
75	19.312840261	10.0.1.10	10.0.2.1	UDP	972	38160 → 3333 Len=5370
79	19.416284970	10.0.1.10	10.0.2.1	UDP	1299	38160 → 3333 Len=5697

```
▸ Frame 105: 1283 bytes on wire (10264 bits), 1283 bytes captured (10264 bits) on interface veth2.0.59, id 0
▸ Ethernet II, Src: 00:00:00_aa:00:03 (00:00:00:aa:00:03), Dst: 00:00:00_aa:00:02 (00:00:00:aa:00:02)
▸ Internet Protocol Version 4, Src: 10.0.1.10, Dst: 10.0.2.1
▸ User Datagram Protocol, Src Port: 38160, Dst Port: 3333
▸ Data (7161 bytes)
```

Figura 3.4: UDP - Captura Wireshark Servidor

No.	Time	Source	Destination	Protocol	Length	Info
27	35.328730601	10.0.2.1	10.0.2.20	UDP	148	4444 → 4444 Len=6026
32	35.988234040	10.0.2.1	10.0.2.20	UDP	74	4444 → 4444 Len=5952
37	36.083729479	10.0.2.1	10.0.2.20	UDP	59	4444 → 4444 Len=5937
42	36.309622799	10.0.2.1	10.0.2.20	UDP	1474	4444 → 4444 Len=5872
46	36.419662956	10.0.2.1	10.0.2.20	UDP	1385	4444 → 4444 Len=5783
50	36.505895414	10.0.2.1	10.0.2.20	UDP	1303	4444 → 4444 Len=5701
54	36.612819431	10.0.2.1	10.0.2.20	UDP	1103	4444 → 4444 Len=5501
58	36.709999010	10.0.2.1	10.0.2.20	UDP	927	4444 → 4444 Len=5325
62	36.823640851	10.0.2.1	10.0.2.20	UDP	833	4444 → 4444 Len=5231
66	36.931800549	10.0.2.1	10.0.2.20	UDP	707	4444 → 4444 Len=5105
70	37.033909782	10.0.2.1	10.0.2.20	UDP	614	4444 → 4444 Len=5012
74	37.138021414	10.0.2.1	10.0.2.20	UDP	584	4444 → 4444 Len=4982

```
▸ Frame 110: 964 bytes on wire (7712 bits), 964 bytes captured (7712 bits) on interface veth4.0.59, id 0
▸ Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:05 (00:00:00:aa:00:05)
▸ Internet Protocol Version 4, Src: 10.0.2.1, Dst: 10.0.2.20
▸ User Datagram Protocol, Src Port: 4444, Dst Port: 4444
▸ Data (6842 bytes)
```

Figura 3.5: UDP - Captura Wireshark Cliente

4 Implementação

Esta secção irá ser divida em 3 componentes, nomeadamente Servidor, Overlay e Cliente, sendo que estas correspondem às aplicações que serão executadas nos três diferentes tipos de nodos.

4.1 Servidor

No que diz respeito ao Servidor, o grupo teve em atenção de que numa primeira fase este não executasse logo a transferência dos pacotes RTP para inicio da Stream nos Clientes, com os objetivos de permitir a implementação da funcionalidade de obtenção de rotas e para facilitar o teste de toda a aplicação.

Em relação à obtenção de rotas, o Servidor é o responsável pela obtenção das mesmas, sendo que, num ponto de vista de facilidade de perceção do Utilzador, neste caso nós e os Professores, esta obtenção é efetuada através do botão "Obter Rotas"apresentado na imagem em baixo:

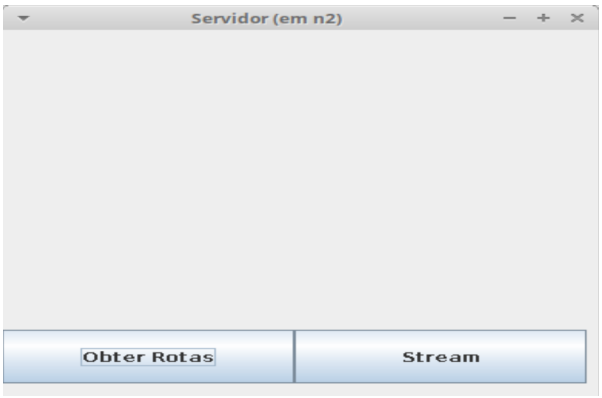

Figura 4.1: Aplicação Servidor

Caso isto fosse um sistema a implementar sem qualquer interação de Utilizador, uma solução viável seria executar a mesma ação, sem qualquer uso de botões mas através de um temporizador que enviaria os Pacotes para os nodos Overlay em certos intervalos de tempo, garantindo que as rotas se manteriam actualizadas.

É também neste Servidor que é dado inicio à Stream para os Clientes, novamente através de um botão, neste caso "Stream", que cria os vários pacotes RTP consoante o vídeo em questão. Caso não sejam passados argumentos, ele irá fazer Stream do vídeo fornecido nos exemplos.

No que diz respeito a todo o processo de criação de pacotes para obtenção de rotas, este começa com a definição do pacote a enviar. Inicialmente, era preciso arranjar uma solução que distinguisse um RTPpacket de um pacote para obtenção de rotas. No caso do RTPpacket, sabemos que o primeiro byte corresponde à versão do mesmo, sendo esta uma constante correspondente a 2. Posto isto, em cada pacote de obtenção de rotas, foi atribuído o valor 0 ao primeiro byte do array de bytes de cada pacote a enviar, sendo que irá ser feita posteriomente, em cada nodo do Overlay, uma comparação do valor deste mesmo primeiro byte. Se tiver o valor de 2, saberemos que se trata de um RTPpacket, enquanto que se tiver o valor de 0, saberemos que temos que o tratar como um pacote de obtenção de rotas. De seguida, este pacote teria que conter a informação a ser tratada pelos nodos do Overlay, sendo esta informação dividia em três campos:

- Mensagem - Indica qual o tipo de mensagem de obtenção de rotas. No caso do Servidor, este apenas enviará do tipo Discovery.
- ID - Indica qual o id do fluxo enviado pelo Servidor para obtenção das rotas que será utilizado para impedir a existência de envio de pacotes em ciclos infinitos.
- Hops - Métrica utilizada para descobrir qual a melhor rota.

No overlay será explicado em mais detalhe os tipos de mensagens e o que é que cada nodo do Overlay faz com o ID. Quanto à implementação das restantes funcionalidades do Servidor, estas encontram-se de acordo com o exemplo fornecido pelos Professores.

4.2 Overlay

Aqui encontra-se o CORE das funcionalidades do sistema. Esta aplicação é responsável pela comunicação entre o Servidor e o Cliente, para que o Servidor consiga executar a Stream de vídeo para todos os Clientes que o requisitam, da maneira mais eficiente possível, formando entre si uma rede de overlay aplicacional, reenviando todo os dados necessários para que a entrega seja efetuada.

Em primeiro lugar é necessário entender que estes nodos Overlay se comportam como Servidores e Clientes, no sentido em que recebem pacotes, tratam esses pacotes e reenviam esses pacotes consoante a necessidade da rede. Tal como vimos no Servidor, estes pacotes serão de dois tipos, nomeadamente os pacotes de obtenção de rotas e os "RTPpackets".

Para a implemntação da funcionalidade de obtenção de Rotas e tal como vimos, é enviado um pacote com 3 parâmetros diferentes, Tipo de Mensagem, ID e Hops. No caso do primeiro este tipo terá três opções diferentes:

- Discovery - Mensagem inicial enviada pelo Servidor e replicada pelos nodos Overlay, para descoberta das rotas consoante o número de Hops.

- RemoveCliente - Mensagem enviada para os nodos anteriores que continham o nodo Overlay actual na sua tabela de rotas, para que esses removam este nodo que envia a mensagem. Isto acontece quando um nodo da melhor rota actual, encontra um nova rota mais eficiente, ou seja, com um menor número de saltos.

- AddCliente - Mensagem enviada para todos os nodos anteriores que farão parte da rota por onde irá ser enviado o fluxo para a execução da Stream.

Com estes três tipos de mensagens, é possível, numa fase inicial com a mensagem de Discovery, ir percorrendo todo o Overlay para obter os nodos que constituem as melhores rotas. Caso um nodo receba uma mensagem de um outro fluxo que contenha uma métrica menor, é enviada então uma mensagem do tipo RemoveCliente para todos os nodos que fariam parte dessa rota antiga, para que estes se removam recursivamente até chegar ao nodo inicial. Se já existisse nesse nodo do Overlay um destino final, era enviada uma mensagem AddCliente para todos os nodos que iriam fazer parte dessa rota mais eficiente.

Em relação ao ID, este serve para que seja identificado o fluxo que vem no pacote do Servidor. Caso um nodo esteja a receber um pacote com um ID que já tenha recebido, este só o irá tratar caso o número de Hops seja inferior à melhor rota que já obtém, garantindo que o ciclo é impedido mas que não são descartadas possíveis rotas que poderiam ser as melhores para a transferência de pacotes para os Clientes.

No que diz respeito à Interface do Overlay, foi seguida a mesma estrutura do Servidor, sendo a mesma apresentada na imagem em baixo:

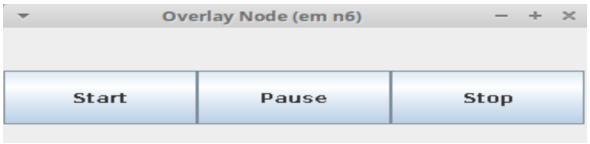

Figura 4.2: Aplicação Overlay

4.3 Cliente

No caso do Cliente, a lógica deste é exatamente a mesma que se encontra no Overlay, com a diferença de que este apenas terá que reenviar pacotes de obtenção de rotas para os nodos do Overlay que lhe enviaram esses mesmos pacotes, confirmando que estes serão de facto os nodos que farão parte da rota para o fluxo da "Stream"ou então para os remover dessa mesma rota se eles fizessem parte dela, assumindo que uma nova rota com melhor métrica foi encontrada.

Em relação à Stream de vídeo, esta está implementada de acordo com o exemplo fornecido, sendo que a Interface para o Cliente se encontra na imagem em baixo:

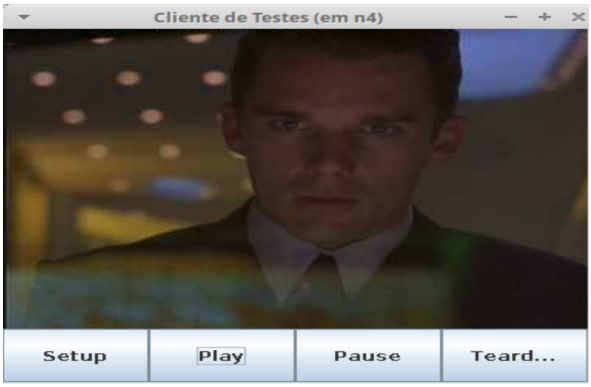

Figura 4.3: Aplicação Cliente

5 Testes e Resultados

Para efeito de testes, utilizamos o CORE para emular a rede física onde está assente a rede overlay, composta pelo Servidor de conteúdos, Routers e respetivos Clientes conforme cenário abaixo.

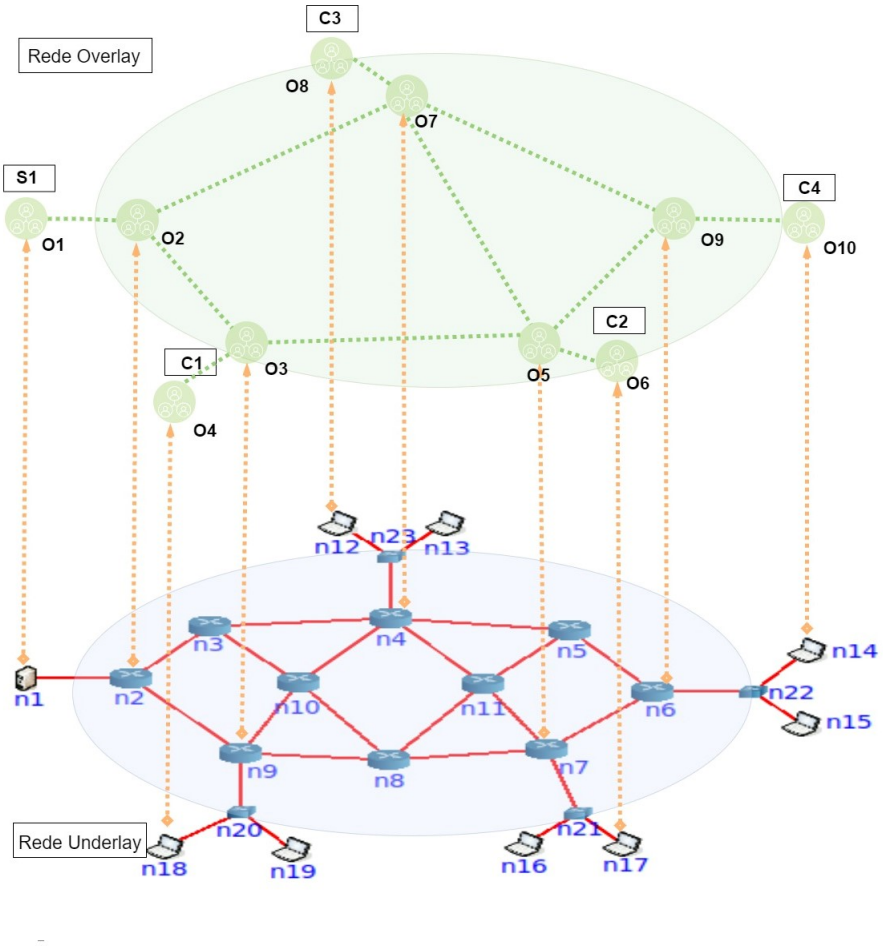

Figura 5.1: Topologia da Rede

Após a execução da simulação da rede no CORE, é necessário abrir os terminais dos vários pontos da rede para execução das aplicações implementadas. Em cada nó Overlay é necessário passar como parâmetros os nós vizinhos, conforme topologia abaixo e posteriormente

iniciar/efetuar o start do overlay conforme imagem.

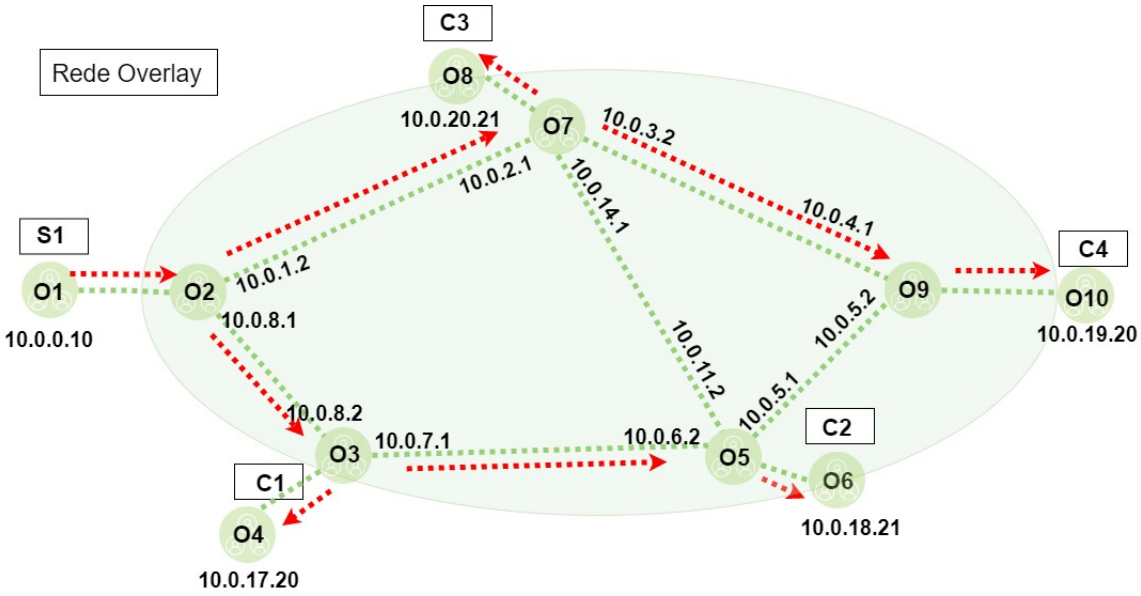

Figura 5.2: Rotas Rede Overlay

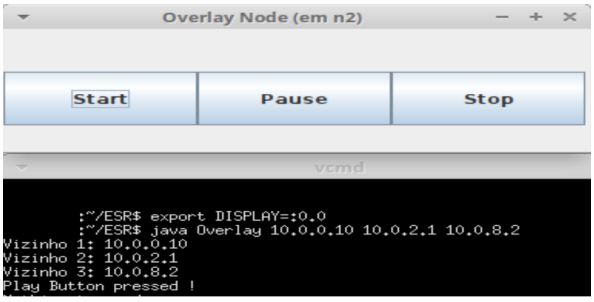

Figura 5.3: Overlay Node

Iniciamos também a aplicação dos nós dos clientes executando apenas o comando "java Cliente"e efetuando o start da mesma. Por fim iniciamos o Servidor, "java Servidor"e na aplicação selecionamos "Obter Rotas"para que o Servidor envie um pacote para efetuar uma inundação controlada da rede, evitando que os nodos que recebem, reenviem para o nodo de onde receberam, com o objetivo de atualizar a tabela de rotas contida em cada nodo do Overlay. Nos restantes nós da rede Overlay conseguimos verificar as rotas enviadas para o servidor e o respectivo custo, conforme podemos verificar no nó overlay O7, tendo como melhor rota até ao servidor o 10.0.1.2 com um custo de 2 hops.

Figura 5.4: Overlay Node O7

Teste 1 - Streaming em 2 clientes em simultâneo

Este cenário simula a entrega do conteúdo de vídeo em 2 clientes em simultâneo (C2 e C4), ligados a diferentes nós overlay (O5 e O9). Verificamos que o streaming ocorre sem qualquer problema nos dois clientes e apenas é verificado o tráfego nos nós overlay com melhor rota até ao servidor.

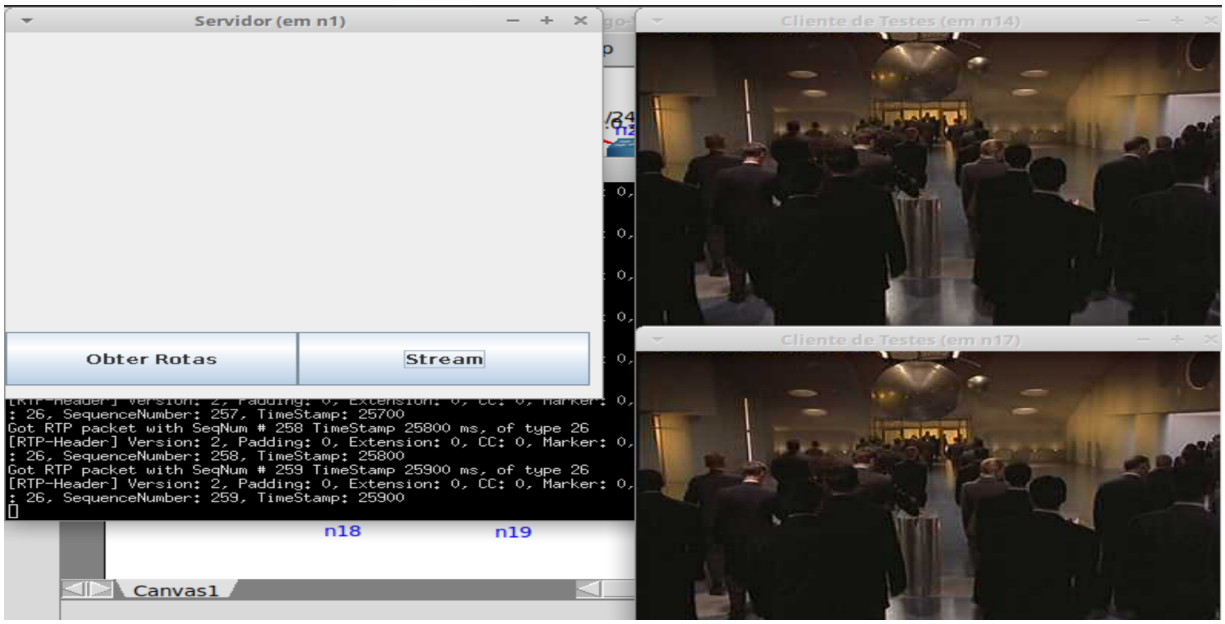

Figura 5.5: Teste 1: Streaming em 2 clientes em simultâneo

Teste 2 - Streaming em 2 clientes em simultâneo com falha nó Overlay

Este cenário simula a falha de um nó Overlay na rota de um dos clientes onde está a ser reproduzido o video. Verificamos que quando existe a falha no overlay (O3) o streaming falha no cliente específico (C2), mantendo o serviço do outro cliente (C3) em funcionamento conforme esperado. Na imagem podemos ver o C3 em funcionamento até ao final da stream, em seguida o C2 parada após falha do O3, último terminal.

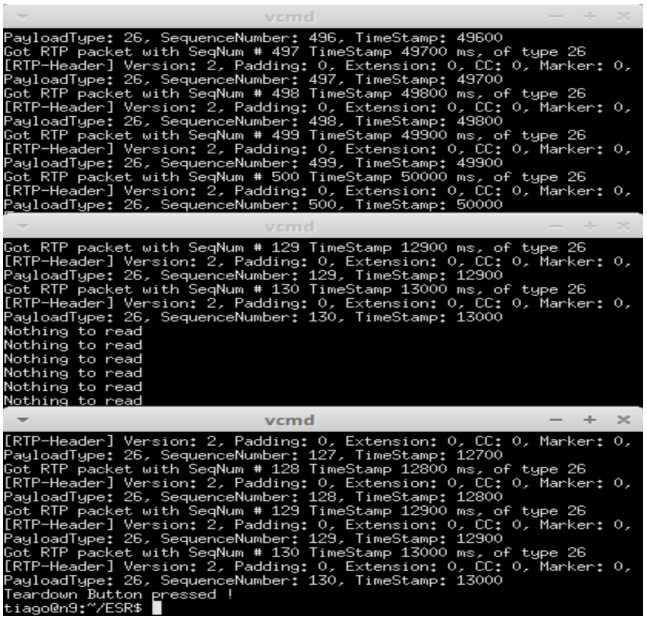

Figura 5.6: Teste 2: Streaming em 2 clientes em simultâneo com falha nó Overlay

Teste 3 - Start-Stop-Start Streaming num cliente

Este cenário simula a reprodução do video num cliente, entretanto este faz um stop e retoma em seguida a visualização do mesmo. Verificamos que quando o cliente faz o stop, o streaming é fechado no cliente, se entretanto o cliente retomar o video, este continua a visualizar o vídeo. Isto acontece devido ao chunk solicitado pelo cliente continuar a ser enviado pelo servidor pelos nós overlays.

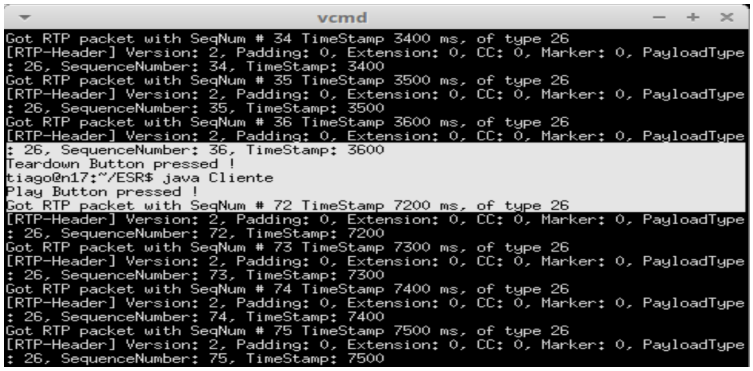

Figura 5.7: Teste 3: Start-Stop-Start Streaming num cliente

Teste 4 - Cenário completo em funcionamento Neste último teste colocamos o cenário completo em funcionamento, baseado na topologia completa que foi apresentada, com os 4 clientes a visualizar o streaming em simultâneo. Após ativação, são apresentados os vizinhos que cada nodo Overlay possuí, ficando o Servidor, nodos do Overlay e Cliente em espera. De seguida, são apresentados os dados das rotas obtidas parte do Servidor e finalmente é iniciada a Stream por parte do Servidor para os vários Clientes

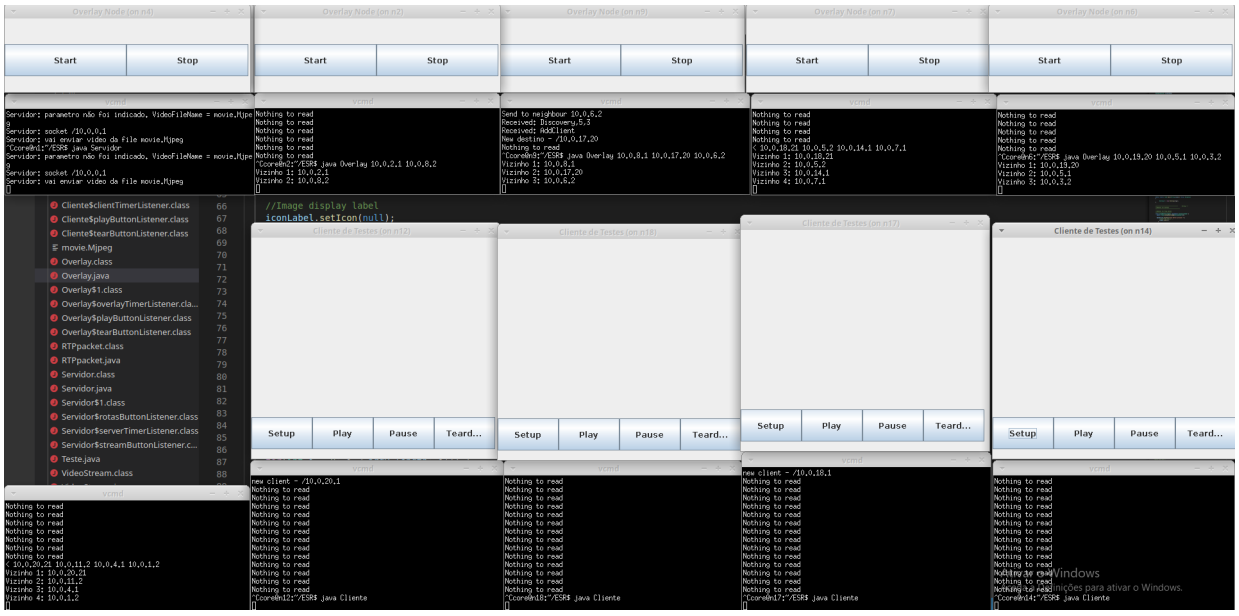

Figura 5.8: Teste 4: Inicialização de todos os Nodos

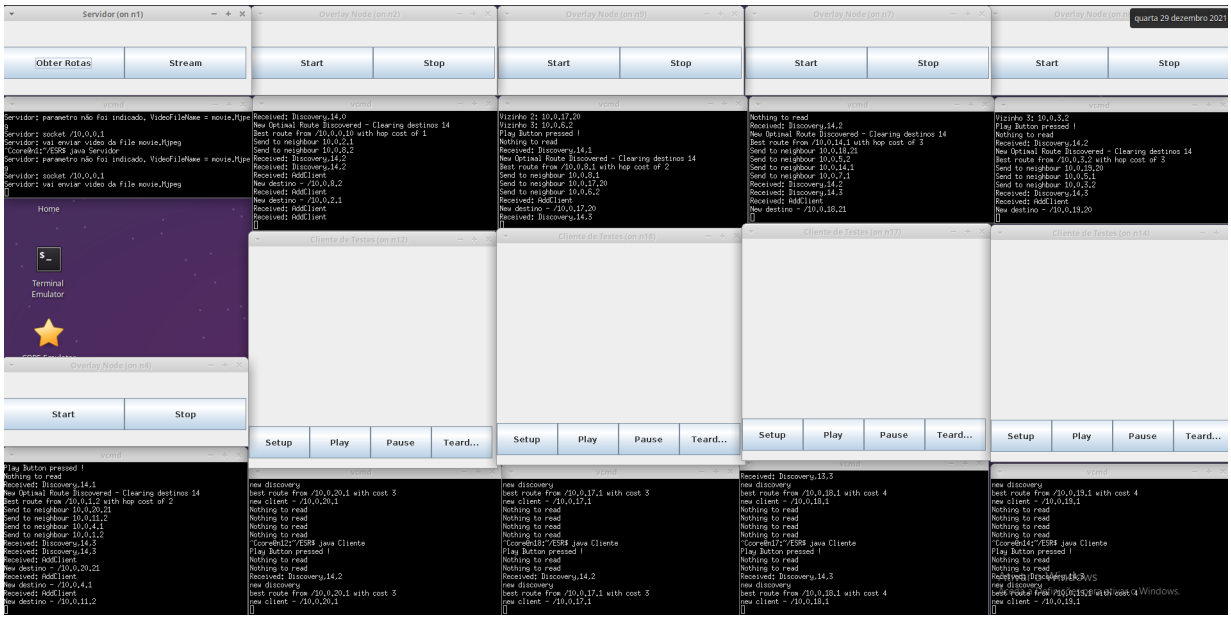

Figura 5.9: Teste 4: Apresentação dos dados/rotas dos nodos do Overlay e dos Clientes

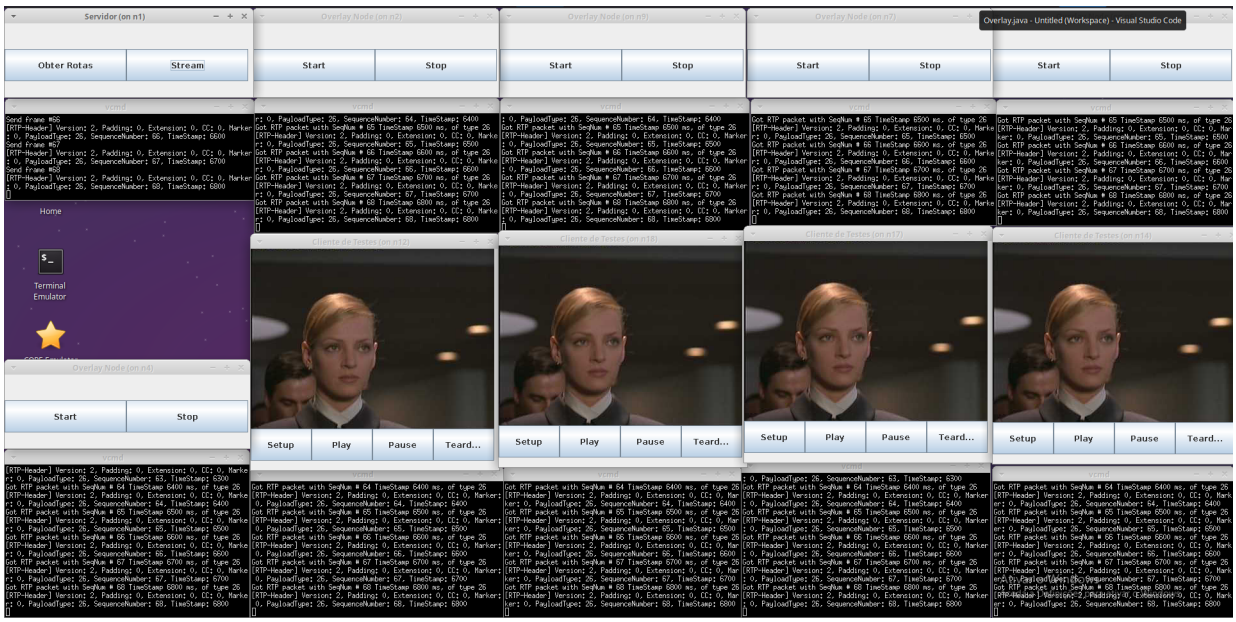

Figura 5.10: Teste 4: Clientes a receber a Stream por parte do Servidor, passando pelos nodos do Overlay

6 Conclusões

Com a realização deste trabalho foi construída uma topologia overlay para partilha de conteúdo em real-time em cima de uma rede física (underlay). Para a construção da rede overlay optamos por uma abordagem manual, na linha de comandos indicamos quais os vizinhos de um determinado nó overlay.

Relativamente à construção de rotas para os fluxos, definindo qual a melhor rota tivemos em consideração o número de saltos e caso o número de saltos seja o mesmo, o tempo de atraso da mensagem enviada pelo servidor. Manualmente o servidor envia uma mensagem de controlo para todos os nós e ao receberem essa mensagem, cada nó atualiza a tabela de rotas com as respetivas informações, ficando estas rotas "suspensas" até à reprodução da stream pelo servidor, estando o cliente já ativo à espera do stream.

Para o envio/reprodução do conteúdo, foi implementado um servidor de streaming capaz de ler o vídeo e enviar para a rede overlay e implementado um cliente capaz de reproduzir o mesmo vídeo numa janela.

Referências

[Kurose and Ross, 2021] Kurose, J. F. and Ross, K. W. (2021). Computer networking, a top-down approach 7th.

http://hdl.handle.net/1822/74179

https://github.com/mshahriarinia/OSPF

http://www.cricte2004.eletrica.ufpr.br/edu/anterior/cd00/trab/rtp/