Universidade do Minho

Departamento de Informática

Engenharia de Serviços em Rede
(1.º Semestre/2021-2022)

Trabalho Prático No.3 – Serviço Over the Top para entrega de multimédia

PG47347 Joel Salgueiro Martins

PG47087 Carlos Filipe Coelho Ferreira

PG47627 Rúben Daniel Almeida Adão


Joel Martins


Carlos Ferreira


Rúben Adão

21 de Dezembro de 2021
2021/2022

Tabela de Conteúdos

1 Introdução

Neste trabalho foi-nos requerido a construção de um serviço especifico desenhado sobre a rede aplicacional que consegui-se cumprir o objetivo de permitir a partilha de conteúdo do tipo vídeo a um conjunto de clientes seguindo a metodologia multicast dada nas aulas de Engenharia de Serviços de Redes.

Este serviço iria permitir a partilha de uma única copia de data entre routers seguindo um fluxo do tipo árvore em que cada cliente fosse uma folha e o servidor a nossa raiz, em vez de enviar múltiplas cópias do mesmo vídeo e sobrecarregar a rede.

Estes tipos de serviços derivam de uma necessidade de reduzir os recursos solicitados de forma cada vez mais voraz a redes como a internet,que ,de todas as formas não foi construída preparada para a grande entrega de conteúdo requerida a ela na atualidade por entidades que conhecemos como Twitch ou Youtube.

O desenvolvimento desta aplicação envolve uma quantidade significativa de trabalho e esforço, passando pela tomada de muitas importantes decisões sobre as técnicas a utilizar para garantir e equilibrar os diferentes atributos desejados no sistema como segurança, performance e adaptabilidade.

No Entanto, a equipa atribui a construção de tais aplicações uma grande importância pois assim como qualquer engenheiro de software esta ciente das dificuldades que provêm do crescimento imparável da Internet.

2 Arquitetura

O projeto desenvolvido consiste em tês aplicações desenvolvidas, que correm em ambientes distintos.

1. **Servidor** - Numa topologia CORE, esta aplicação corre apenas sobre um servidor. Aplicação cuja tarefas são: delegar aos nodos/clientes que são os seus vizinhos na rede *overlay* (desempenhando também um papel necessário para a adição de novos nodos em tempo real), iniciar o processo de *flodding* e execução do streaming de vídeo.

2. **Nodo** - Numa topologia CORE, esta aplicação corre sobre nodos não clientes e não servidores. Esta aplicação serve de ponte entre as varias ligações que representam a rede overlay e a sua tarefa principal é enviar os pacotes de streaming pelas rotas criadas durante a execução do programa.

3. **Cliente** - Numa topologia CORE, esta aplicação corre sobre nodos clientes, ou seja, nodos que pretendem visualizar a stream. A diferenciação desta aplicação Cliente para com a aplicação Nodo é o facto de que é esta aplicação que dá início ao processo de geração de rotas (através de *backtracking*) após um flooding - como iremos demonstrar à frente. Além disso, contrariamente à aplicação Nodo, esta aplicação está feita com o objetivo de visualizar o vídeo enviado pelo servidor por através de uma interface gráfica.

Em termos arquiteturais, podemos também dizer que o projeto apresenta 4 fases, que iremos descrever sucintamente.

2.1 Criação da Rede Overlay

Processo inicial em que o única aplicação estritamente necessária a correr em primeiro lugar é o Servidor. Conforme os restantes nodos se vão ligando à rede, o Servidor indica quais são os seus vizinhos na overlay. Uma conexão é estabelecida entre todos os vizinhos.

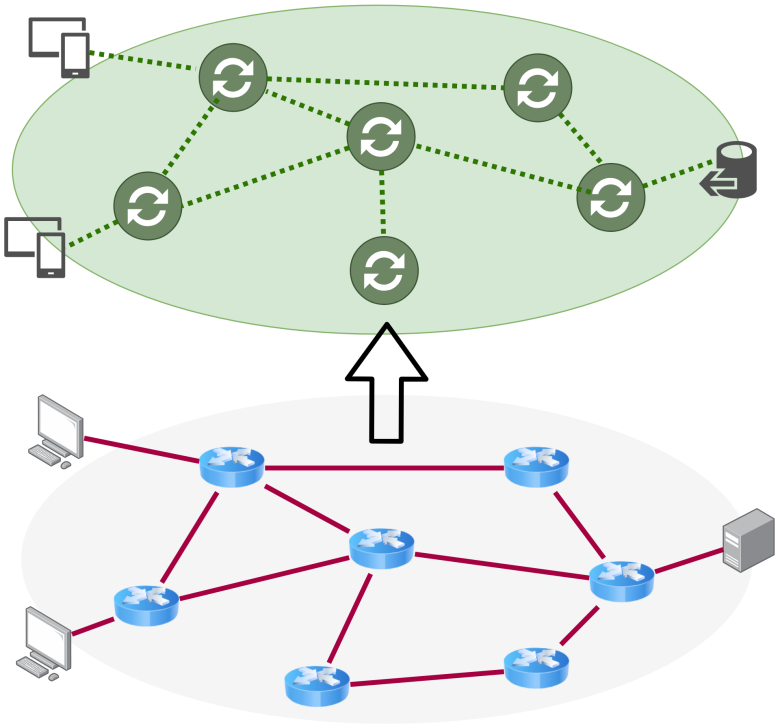

Figura 1: Exemplo da criação de uma rede overlay.

2.2 Flooding e Backtracking

Após uma rede overlay estabelecida com pelo menos um caminho efetivo entre todos os clientes e o Servidor, este último dá início ao processo de flodding. De forma breve, pacotes de teste são enviados pela rede através dos nodos quem enviam para os seus vizinhos, de forma a

todos os nodos se aperceberam de qual a melhor rota por onde consegue obter pacotes vindos do Servidor. Após este mecanismo, e assegurando que o Cliente está ciente de que vizinho deverá obter o conteúdo da stream, este inicia o processo de backtracking - por ordem, o cliente e os nodos implicados nas rotas dizem ao seu vizinho "preferido"que querem que os dados sejam encaminhados pelo próprio. Eventualmente o mesmo é feito até ao Servidor.

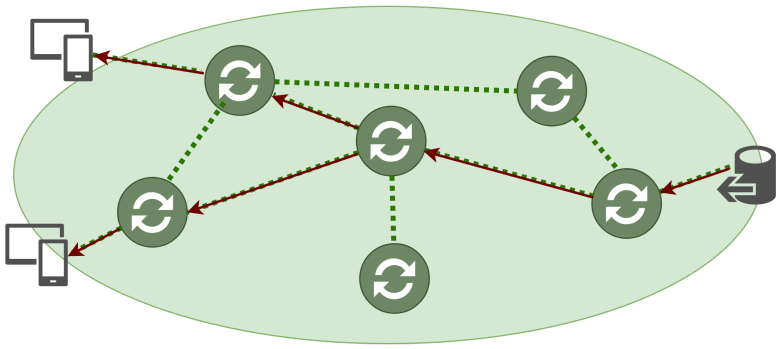

Figura 2: Criação de uma rota fruto de um processo de flood.

2.3 Streaming de Multimédia

Este processo inicia-se após o flooding e o backtracking estarem finalizados. Consiste apenas na viagem dos pacotes de multimédia pelas rotas definidas através dos mecanismos descritos anteriormente.

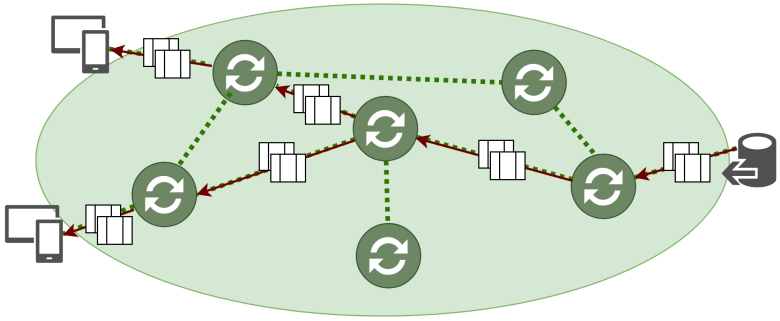

Figura 3: Representação de envio de pacotes por streaming.

2.4 Manutenção

A fase de manutenção diz respeito à continuação de streaming de forma efetiva do Servidor para o Cliente após o seu início ter sido bem sucedido, tendo em conta que os nodos na rede podem sair ou entrar na rede overlay, implicando mudanças das rotas de encaminhamento.

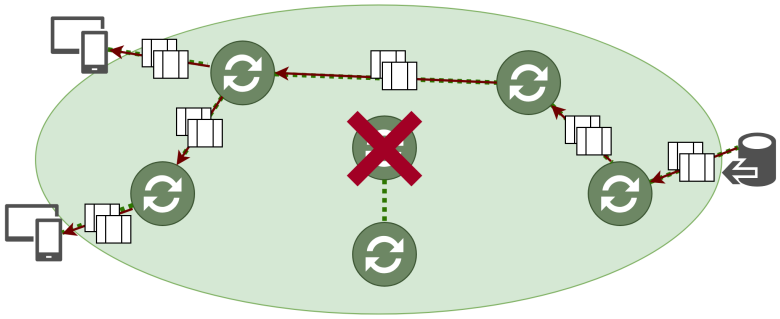

Figura 4: Resultado da remoção de um nodo na overlay.

3 Especificação dos protocolos

3.1 TCP vs UDP

Inicialmente, uma das decisões tomadas pelo grupo foi acerca do protocolo de transporte que melhor se adapta ao problema proposto pelo enunciado.

Por um lado faz sentido a utilização de UDP para um serviço de streaming uma vez que perdas de pacotes não são críticas ao desempenho do sistema (é preferível perder pacotes do que provocar atrasos excessivos). Por outro lado o protocolo TCP assegura-nos a receção de mensagens que no nosso sistema são vistas como mensagens críticas (por exemplo, a mensagem que o Servidor envia a cada um dos nodos que se conecta à rede, para indicar quais são os seus vizinhos).

Tendo estas razões em conta, decidimos utilizar uma abordagem mista e utilizar ambos os protocolos, simplesmente para tarefas distintas:

TCP

- Conexão Servidor/Nodo
- Mensagens de atualização de vizinhos
- Conexão entre vizinhos
- Flooding e BackTracking

Desta forma não só conseguimos efetuar a fiabilidade de mensagens de elevada importância, além de que obtemos outros benefícios, como por exemplo, os nodos conseguem saber imediatamente que um nodo vizinho praou de funcionar e eventualmente estabelecer uma redirecionamento das rotas de streaming.

UDP

- Envio de pacotes de multimédia

Uma vez que estes pacotes não devem ser sensíveis a perdas nem a erros, o protocolo UDP é o mais indicado

3.2 Protocolos de Aplicação

Iremos agora fazer uma breve descrição dos protocolos aplicacionais criados por cima dos protocolos discutidos acima (não apresentam nenhum nome identificativo, pelo que serão denominados pelo protocolo de transporte que utilizam).

3.2.1 Protocolo sobre TCP

Este protocolo não possui tamanho pre-definido uma vez que a própria camada de transporte e a utilização de Java oferecem ferramentas que tornam tal especificação redundante. Para a utilização deste protocolo é apenas necessário enviar um cabeçalho com flags (este com tamanho definido), seguido pelo conteúdo apropriado (dependendo das flags ativas). As flags utilizadas foram:

- **FLAG_DATA** - indica que este pacote representa dados (é uma flag de desenvolvimento não utilizada em produção)
- **FLAG_KILL** - flag que solicita que um nodo termine a sua execução
- **FLAG_FLOOD** - flag que indica que o pacote é de flood
- **FLAG_ACTIVATE** - flag que solicita que um nodo esteja ativo na rede overlay
- **FLAG_DEACTIVATE** - flag que solicita que um nodo esteja desativo na rede overlay

- **FLAG_NEW_NEIGHBOURS** - indica que o pacote irá enumerar novos vizinhos que o nodo deve atualizar
- **FLAG_ID** - flag que indica o id de um nodo (é uma flag de desenvolvimento não utilizada em produção)

Nota: As flags são enviadas como bits, neste caso como existem apenas 7 flags, basta 1 byte de cabeçalho.

3.2.2 Protocolo sobre UDP

Foi utilizado o protocolo providênciado pelos docentes da unidade curricular

4 Implementação

Serão agora descritos os vários processos enumerados na secção de arquitetura, com maior detalhe:

4.1 Criação de uma Rede Overlay

Para a execução do nosso serviço, antes de qualquer outro nodo iniciar, é necessário que o Servidor esteja a correr. A partir deste ponto, qualquer nodo que se conecte à overlay inicia contacto com o servidor (todos os nodos conhecem o ip e a porta de serviço do servidor) e anunciam que pretendem-se juntar à rede. O servidor verifica se existe algum nodo ativo que seja vizinho do nodo que se está a ligar, e para todos os que estiverem ativos, avisa-os que possuem um novo vizinho. Da mesma forma anuncia ao nodo a ligar os vizinhos que estão ativos.

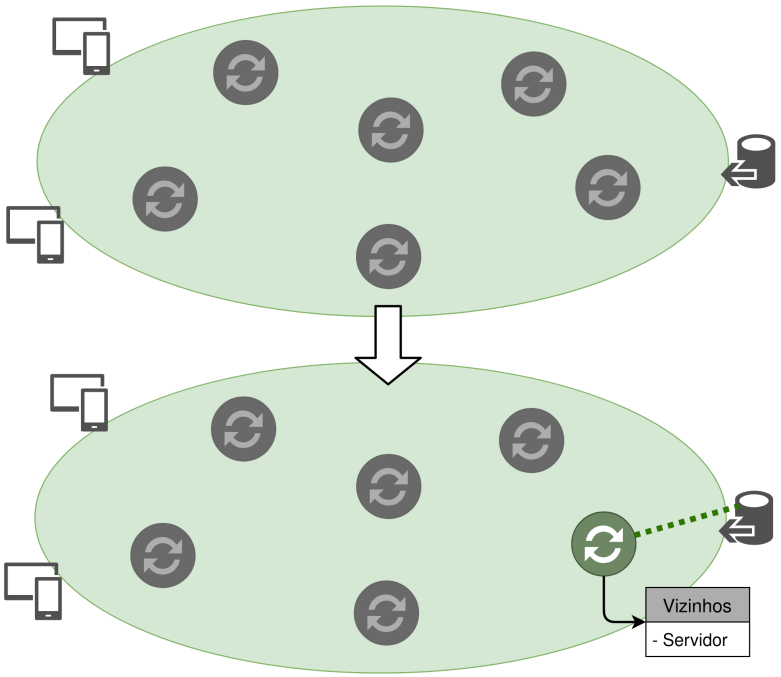

Figura 5: Conexão do primeiro nodo ao servidor.

Na figura acima é dado uma representação da transição inicial da rede overlay, apenas com o servidor, para um estado seguinte após a ativação do primeiro nodo. Como vizinhos este apenas possui o servidor.

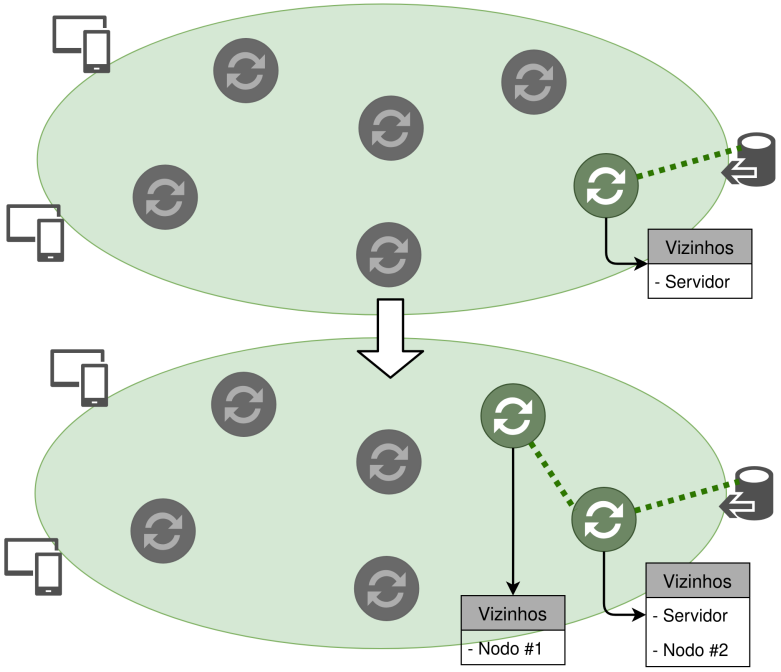

Figura 6: Conexão do segundo nodo.

Quando o segundo nodo se conecta o Servidor envia-lhe uma mensagem a alertar de que é vizinho do nodo #1, assim como uma mensagem ao nodo #1 a dizer que um novo vizinho se ligou à rede overlay.

Este processo repete-se até que uma configuração inicial da rede de overlay esteja conectada (como requisito mínimo, para avançar para a próxima fase é necessário a existência de pelo menos um cliente)

4.2 Flooding e Backtracking

4.2.1 Flooding

O flooding é o processo a partir do qual os nodos se apercebem de qual será o melhor vizinho a utilizar para que os dados cheguem a si mesmo (seguinda uma determinada métrica). Neste caso a métrica utilizada foi o número de saltos na reder overlay.

A estratégia de flooding é simples: o servidor envia um pacote com o sinal de número de salto "1". Cada nodo envia aos seus vizinhos o mesmo pacote incrementando-lhe um valor de salto, e de cada vez que recebe um pacote, guarda o vizinho pelo qual o recebeu assim como o valor de salto numa tabela de redirecionamento. Além disso ninguém envia pacotes para vizinhos a partir dos quais já receberam.

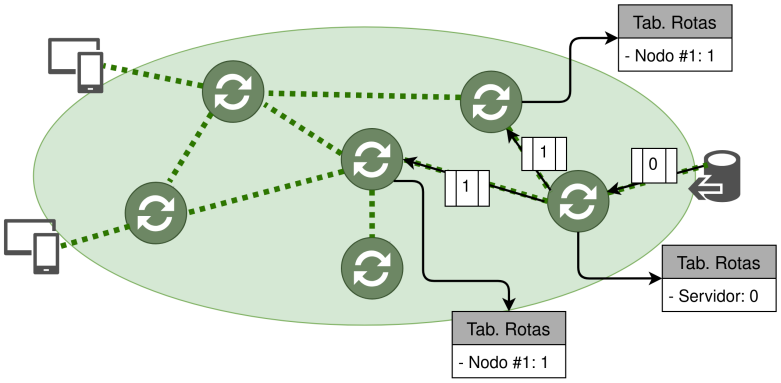

Figura 7: Representação do iníciode um flooding.

Eventualmente, todos os nodos da rede apresentaram uma tabela com os vizinhos por onde podem receber pacotes e o associado custo de salto.

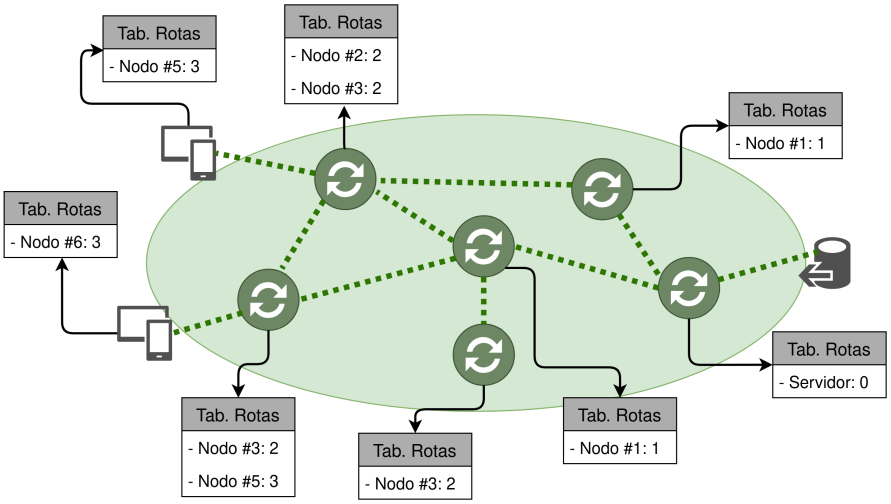

Figura 8: Fim do processo de flooding.

4.2.2 Backtracking

Depois do flooding, os clientes iniciam o processo de backtracking. Para tal, verificam na tabela de rota qual o vizinho mais próximo (segundo a métrica de saltos) e avisam-lhe que pretendem receber pacotes a partir dele. Por consequência, o vizinho após receber este tipo de mensagem executa uma ação similar até eventualmente a informação chegar ao servidor. Desta forma o Servidor sabe para quem mandar pacotes e os nodos sabem para quem os redirecionar.

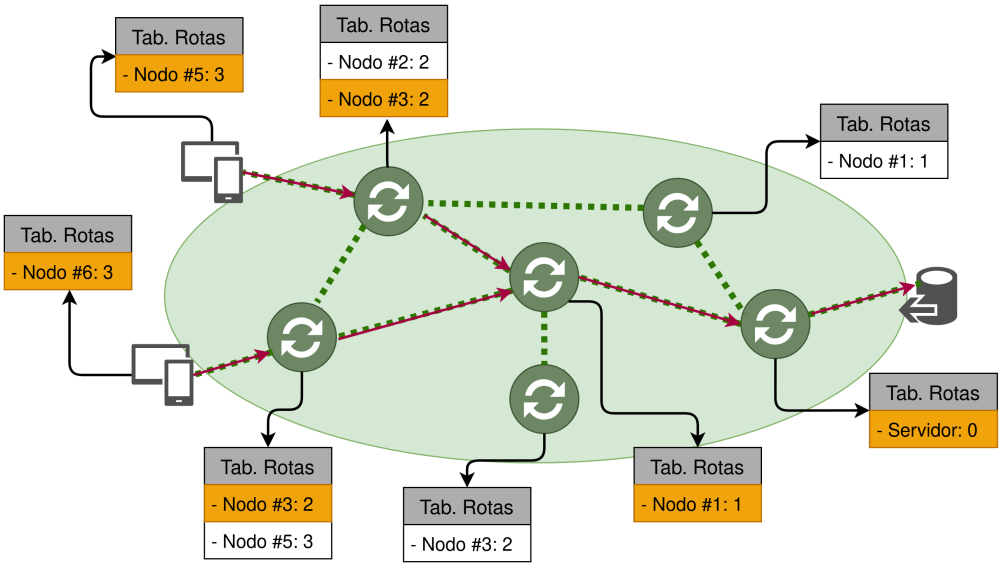

Figura 9: Representação do processo de backtracking.

4.3 Streaming de Multimédia

Para a implementação do streaming de multimedia foi utilizado o código fornecido pelos docentes.

4.4 Manutenção

Quando um vizinho detecta que um dos seus vizinhos se desligou, envia uma mensagem ao Servidor e este reinícia o processo de flood. Quando isto acontece nota-se uma paragem e um retorno do streaming

5 Testes e Resultados

Após inserir a topologia no CORE (além de estabelecer certos detalhes, como por exemplo passar por argumento a todos os nodos o ip e a porta de serviço do servidor), basta primeiramente conectar um número arbitrário de nodos e depois escrever no terminal do Servidor a palavra "flood". Este comando desencadeia o primeiro flood, que gerará as rotas. Utilizando o código de streaming fornecido, por cima da nossa aplicação, podemos ver o vídeo a ser transmitido pelo servidor até dois computadores.

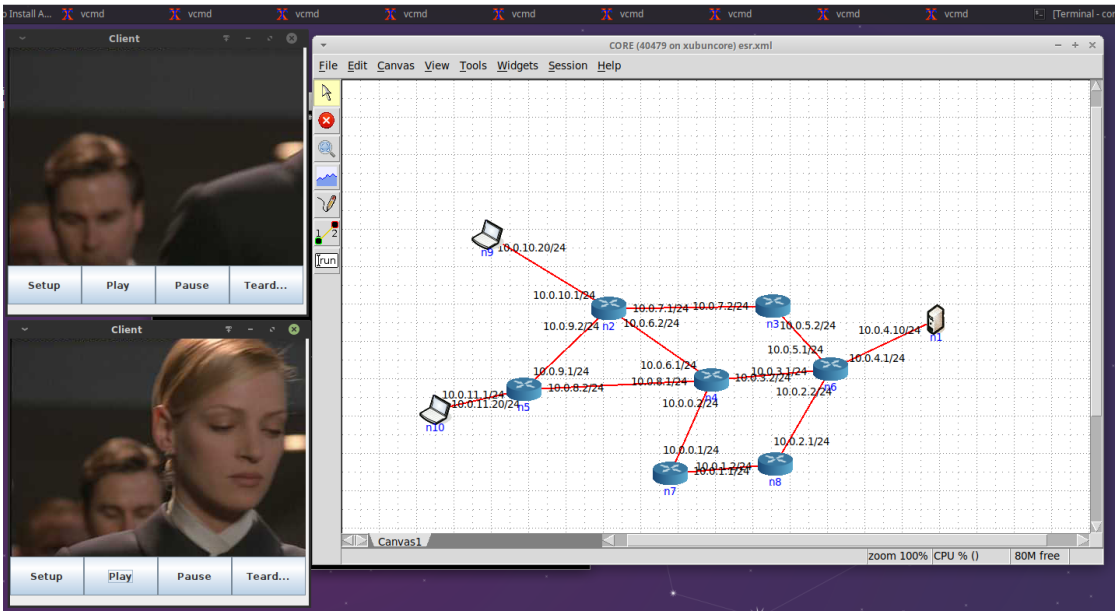

Figura 10: Print de um instante durante a transmisão da stream. As duas janelas de vídeo correspondem ao laptop 1 e 2.

Um detalhe interessante que podemos verificar é de que os vídeos não estão perfeitamente síncronos. Isto deve-se ao facto de termos adicionado delay ao link que liga um dos laptops ao seu router vizinho.

Podemos verificar também que os nodos que pertencem ao overlay mas não fazem parte da rota não apresentam mensagens de emissão de pacotes UDP:

```
received: 2,
handling.................
este socket esta conectado a 7004
metrica atualizada para nodo 7004
checking to send flood to 7002
checking to send flood to 7004
updated metricas: {7002=2, 7004=3}
received: 2,
handling.................
este socket esta conectado a 7004
metrica atualizada para nodo 7004
checking to send flood to 7002
checking to send flood to 7004
updated metricas: {7002=2, 7004=3}

Process finished with exit code 130 (interrupted
```

```
received UDP packet
sent UDP packet
received UDP packet
sent UDP packet
received UDP packet
sent UDP packet
received UDP packet
sent UDP packet
received UDP packet
sent UDP packet
received UDP packet
sent UDP packet
received UDP packet
sent UDP packet
received UDP packet
sent UDP packet

Process finished with exit code 130 (interrupte
```

Figura 11: À esquerda o output de um nodo do overlay que não pertence à rota. À direita, um nodo que pertence.

6 Conclusão

O desenvolvimento deste software demonstrou mais uma vez á equipa a grande complexidade existente nas redes que permitem a entrega suave e eficiente de conteúdo aos nossos computadores, que ao utilizador comum da Internet se encontra transparente e desconhecida.

A Internet é um campo de software que não apresenta nenhum sinal de diminuição, pelo contrário, demonstra mesmo após estes anos ainda um crescimento mais acentuado.

Como engenheiros de software esta é uma área que estará sempre presente na nossa carreira e conhecimento sobre o domínio será continuadamente valorizado.

A realização deste projeto aumentou as nossas capacidades de analise e desenvolvimento de aplicações sobre a camada aplicacional de uma rede e consequentemente um conjunto de boas práticas e ideias para futuros projetos que sejam exigidos durante a nossa carreira.