Universidade do Minho

Escola de Engenharia

Engenharia de Serviços em Rede

Mestrado em Engenharia Informática

Serviço *Over the Top* para entrega de multimédia

Trabalho Prático 2 - PL51

Francisco Novo - PG50374
Francisco Izquierdo - PG50384
Tiago Ribeiro - PG50779

Dezembro, 2022

Conteúdo

1. Introdução

No âmbito da Unidade Curricular de Engenharia de Serviços em Rede, foi desenvolvido o segundo trabalho prático proposto pelos docentes. Neste projeto pretendeu-se conceber um protótipo de entrega de áudio/vídeo/texto com requisitos de tempo real, a partir de um servidor de conteúdos para um conjunto de N clientes, no qual os mesmo estão assentes sobre uma camada *underlay* que tem a capacidade para detalhar várias camadas *overlay* diferentes. Além disso, neste trabalho pretendeu-se conceber e prototipar um dos serviços disponibilizados pela camada de transporte, nomeadamente através de um ou mais protocolos de transporte, que promovesse a eficiência e a otimização de recursos para melhor qualidade de experiência do utilizador.

Para tal, a ferramenta *CORE* mostrou ser fulcral no suporte ao desenvolvimento do projeto. Ao longo deste relatório, serão abordadas as várias etapas que constituíram a solução final sendo explicadas em detalhe, no qual cada uma mostrou ser importante na sua medida.

2. Arquitetura da solução

A solução desenvolvida teve por base uma arquitetura, tendo esta sido pensada e detalhada com vista a alcançar o objetivo proposto. A arquitetura desenvolvida visa em servir de suporte para poderem ser atendidos os requisitos:

- Serviço de streaming;
- Construção de rotas para a entrega de dados;
- Monitorização da rede overlay;
- Ativação e teste do servidor alternativo;

Além disto, a arquitetura desenvolvida deve estar apta a qualquer camada *overlay*, uma vez que para uma mesma camada *underlay*, podemos ter diferentes camadas *overlay*. Com isto, a solução assenta sobre os protocolos de transporte TCP e UDP simultaneamente, segundo os seguintes pretextos:

- **Protocolo TCP**: Protocolo usado na trocar de mensagens entre os vários nodos constituintes da camada *overlay*;
- **Protocolo UDP**: Protocolo usado na entrega do serviço de stream;

Assim, qualquer troca de informação entre os nodos da camada *overlay* é feito através do protocolo TCP/IP, no qual as mesmas mensagens incluem informação relevante aos aspetos mencionados acima.

Um ponto importante a referir posto isto, é a construção da topologia *overlay*, no qual é feita passando para cada nodo, a lista dos endereços ips dos nodos vizinhos no arranque do programa. Deste modo, cada nodo guarda a lista de ips dos nodos vizinhos a quem, ao longo do decorrimento do programa, irá contactar e receber informações. Além disto, existe uma separação nas funções de cada nodo existente na camada, isto é, dependendo do tipo de nodo, este tem uma determinada função. Com isto, temos os seguintes tipos de nodos:

- **Cliente**: Nodo responsável por pedir a stream pretendida e disponibilizá-la ao utilizador do serviço;
- **Router**: Nodo responsável por encaminhar a stream a um determinado cliente, bem como, construir as rotas para entrega de dados, monitorizar a rede e permitir o cliente contactar o servidor;
- **Servidor**: Nodo responsável por disponibilizar a stream a potenciais clientes, bem como ajudar na construção de rotas;

Com isto, consoante o tipo de nodo, a aplicação terá um determinado comportamento. De modo geral, a arquitetura está estabelecida para funcionar do seguinte modo:

1. Cada router é inicializado sem conhecer a topologia, esperando por mensagens;
2. Posteriormente são ligados os os servidores, que ao arrancarem o serviço, difundem uma mensagem para a rede a informar da sua ativação, com informação que permite os routers terem um certo conhecimento da camada *overlay* e construirem as suas rotas.
3. Uma vez definidas as rotas, os clientes podem aceder ao serviço de stream, no qual realizam um pedido à rede;

A arquitetura implementada foi projetada com vista a mitigar um dos maiores problemas implícitos neste tipo de serviço, a sobrecarga do lado do servidor para um número crescente de clientes. Para tal, a solução passou por cada router ao receber um pedido por parte de um novo cliente pelo serviço stream, verificar se o mesmo já não o está a enviar a um outro cliente. Em caso afirmativo, o router não reencaminhará o pedido até ao servidor, sendo este encarregue de servir o novo cliente em UNICAST, reencaminhando também para este o serviço de stream já prestado. Desta forma, além de não haver uma sobrecarga no lado do servido, é feito um controlo e gestão de fluxos e dados a circular na rede.

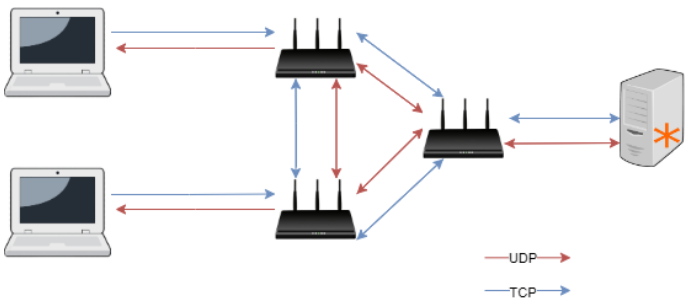

Figura 2.1: Arquitetura

3. Especificação dos Protocolos

Tal como abordado previamente no capítulo anterior, são usados os dois protocolos de transporte na solução do problema. Os protocolos UDP e TCP/IP são usados na solução tendo cada um, um propósito associado.

3.1 Protocolo UDP

O protocolo UDP é apenas usado para servir um dado cliente quando este requisita o serviço de stream pretendido, sendo que está implementado nos 3 tipos de nodos. Nomeadamente no cliente, para entregar o conteúdo de stream, no router de forma a redirecionar o conteúdo stream aos clientes e no servidor para enviar o conteúdo stream aos clientes. Deste modo, o protocolo UDP é utilizado para disponibilizar o serviço stream sendo motivado pelo facto de que uma vez que é um serviço em tempo real, a perda de pacotes respetivos à stream a entregar ao cliente, não é crítica dado que o conteúdo perdido não invalida o teor do conteúdo seguinte, sendo mais vantajoso nestas situações, perder pacotes ao invés de esperar pelos mesmo.

3.2 Protocolo TCP/IP

No que concerne ao uso do protocolo TCP, este é usado na troca de mensagens que contêm informação necessária para gerir toda a atividade presente na camada *overlay*, sendo estas trocadas entre os nodos da mesma. O uso deste protocolo justifica-se na vertente em que é necessário garantir a consistência de toda a informação presente na camada *overlay*, informação esta relevante aos tópicos já mencionados. Acima de tudo, uma vez que informações relevantes serão trocadas entre os nodos, queremos garantir a entrega fiável das mensagens que contêm estas informações. Deste modo, teremos a certeza que cada nodo ao enviar mensagens com uma determinada informação, esta é garantidamente entregue ao nodo destino, assegurando a integridade da informação presente na camada.

Com isto, a informação presente nas mensagens sobre este protocolo é diversificado, tendo por isso um formato apropriado, por forma a organizar a informação contida. Neste âmbito, iremos dar ênfase ao formato destas mensagens protocolares.

3.2.1 Formato das Mensagens protocolares

Figura 3.1: Formato da mensagem protocolar

Campo	Descrição
Tipo	Tipo do pacote.
Fonte	Ip do nodo que originou o pacote.
PortaTCPFonte	Número da porta TCP/IP do nodo que originou o pacote.
Host	Ip do nodo que reencaminhou o pacote.
PortaTCPHost	Número da porta TCP/IP do nodo que reencaminhou o pacote.
Destino	Ip do nodo de destino a quem se pretende entregar o pacote.
Saltos	Número de saltos efetuados pelo pacote.
TimeStamp	Instante de tempo em que foi criado o pacote.
Id	Id único do pacote.
Tokens	Lista de *tokens*.

Tabela 3.1: Campos da mensagem protocolar

Posto isto, convém dar ênfase a alguns dos campos presentes no formato dos pacotes e explicar a sua função. Assim, temos:

- **PortaTCPFonte** e **PortaTCPHost**: Campos com relevância apenas para uso local.
- **Saltos**: Permite calcular o número de saltos que separam o nodo que recebeu o pacote até ao nodo que originou o pacote. Através disto, um router sabe o número de saltos até ao servidor que originou a mensagem de tipo 1, tendo a métrica saltos da tabela de encaminhamento que diz respeito a uma determinada rota, para esse mesmo servidor.
- **TimeStamp**: Permite calcular o atraso em milissegundos entre o nodo que recebeu o pacote até ao nodo que originou o pacote. Através disto, um router sabe o atraso até ao servidor que originou a mensagem de tipo 1, tendo a métrica atraso da tabela de encaminhamento que diz respeito a uma determinada rota, para esse mesmo servidor.
- **Id**: Permite caracterizar de forma única cada pacote.
- **Tokens**: Permite a implementação do *floading* controlado, estando presente nos pacotes de tipo 1, uma vez que cada router antes de reenviar ou servidor antes de enviar o pacote insere o seu *token*. Assim, um router ao receber o pacote, ao verificar que este já contém o *token* do seu router vizinho, não lho reencaminha, dado que já passou por este.

3.2.2 Interações

Uma vez abordado o formato e conteúdo dos vários campos presente num pacote, resta agora explicar como os diferentes pacotes são trocados entre os nodos da camada *overlay*. Para isto, o campo que diz respeito ao **tipo** de pacote permite distinguir o tipo de informação contida em cada pacote. Assim, para cada pacote, haverá uma interação a decorrer, sendo que temos os seguintes tipos de pacotes:

1. Este pacote contém informação que ajuda o router a criar a sua tabela de encaminhamento, com rotas dirigidas a cada destino (servidor) possível, tendo várias métricas, como interface de saída, número de saltos, atraso e estado da rota, relativas aos mesmos. Este pacote é originado apenas pelos servidores quando são iniciados e em intervalos de tempo constante (de forma a haver uma manutenção atualizada das rotas) e ao chegar a um router, é realizado o *floading* controlado do mesmo, por forma a propagar a informação.

2. Este pacote contém informação sobre o pedido de stream. Este geralmente é originado por um cliente, sendo posteriormente reencaminhado pelos routers consoante a melhor rota ao melhor servidor. Contudo pode também ser originado por um router que está a servir os pacotes de stream a um dado cliente quando as condições de entrega se degradam para um dado servidor, havendo por isso um melhor servidor a quem pedir a stream.

3. Este pacote contém informação sobre o término de stream para um cliente. Este pacote é originado por parte do cliente quando pretende abandonar a sessão de stream.

4. Este pacote contém informação sobre o término de stream para um router. Geralmente é originado quando um router não tem nenhum nodo a quem redirecionar os pacotes de stream, enviando o pacote ao router que lhe servia os pacotes da stream. Contudo, pode ser originado quando as condições de entrega se degradam, no qual o router pretende mudar de rota e pedir os pacotes de stream a um outro router.

Posto isto, as interações que decorrem podem ser vistas da seguinte forma, cada router é inicializado aguardando que os servidores sejam ativados e lhes enviem pacotes de tipo 1 por forma a construir as suas tabelas de encaminhamento. Seguidamente, ao haver o estabelecimento de rotas, ao haver um novo cliente, este envia um pacote de tipo 2 para que lhe seja servido a stream. Assim que este deseja abandonar a sessão envia um pacote de tipo 3 que é reencaminhado pelos routers enviando pacotes de tipo 4 caso estes não esteja a fornecer a stream a um outro cliente. Contudo, quando as condições de entrega se degradam, caso um router esteja a enviar pacotes da stream a um dado cliente, gera um pacote do tipo 4 enviando-o ao nodo que lhe estava a reencaminhar os pacotes de stream e gera um pacote do tipo 2 enviando-o ao nodo vizinho pertencente à melhor rota do novo servidor.

4. Implementação

No que concerne à implementação da solução final, há aspetos a considerar relativamente aos diferentes tipos de nodos. Antes disto convém mencionar a linguagem de programação escolhida no desenvolvimento da aplicação, sendo esta Java uma vez que o grupo de trabalho

tinha mais experiência na mesma o que permitiu uma abordagem mais confiante perante o problema proposto. Com isto, iremos dar ênfase às classes implementadas e que suportaram a solução.

- **MenuUI**: Esta classe está presente em qualquer nodo, sendo que disponibiliza um menu ao serem inicializados, no qual podem escolher operar segundo um dado modo, relativo ao tipo de nodo. Após a escolha do modo de operar, o menu dá a indicação ao utilizador para que insira a lista de nodos vizinhos do nodo e este dá indicação à aplicação para decorrer segundo o modo escolhido.

- **Cliente**: Esta classe é utilizada por nodos cujo modo de funcionamento é de cliente. Assim que um cliente inicia, esta classe gera um pacote do tipo 2 por forma a pedir a stream. Além disto, permite gerir a entrega da stream com o auxílio da classe **ClienteStream**, que faz gestão dos pacotes UDP respetivos à stream.

- **Servidor**: Esta classe é utilizada por nodos cujo modo de funcionamento é de servidor. Assim que o servidor inicia, cria para cada nodo vizinho uma *thread* da classe **VizinhoServerWorker** sendo esta classe responsável por gerir a informação presente no nodo como encaminhar quer a stream, quando necessário, quer pacotes de manutenção de rotas. Em intervalos de tempo constantes, vai enviando aos nodos vizinhos pacotes de monitorização de rotas que vão sendo proliferados através de um inundamento controlado na rede pelos nodos routers.

- **Router**: Esta classe é utilizada por nodos cujo modo de funcionamento é de router. Uma vez que os nodos routers são os primeiros a serem inicializados, aguardam em modo servidor pela chegada de pacotes. Assim que estes chegam, cria uma *thread* da classe **RouterWorker** respetiva ao nodo vizinho que entregou o pacote, sendo esta classe responsável por gerir a informação presente no nodo, bem como lidar com o tipo de pacote TCP/IP recebido.

- **RouterUDPWorker**: Esta classe está presenta na implementação da classe **Router**, dado que é responsável por entregar os pacotes UDP da stream aos clientes/ nodos vizinhos em UNICAST.

- **UDPCenter**: Esta classe encontra-se presente tanto na implementação do servidor como do router, sendo a classe que contém a informação para gerir os pedidos de stream e as conexões entre nodos vizinhos. Contém informação sobre o número e endereços ips dos nodos que realizaram o pedido por entrega da stream, por forma a reencaminhar os pacotes UDP em UNICAST no caso de haver mais que um cliente. Além disso, tem informação sobre a tabela de encaminhamento do nodo, no caso de este se tratar de um router. Esta classe é comum às várias *threads* presentes no nodo, sendo acedida concorrentemente, de modo a manter a integridade e coerência dos dados do nodo.

Posto isto, convém realçar o uso de *threads* tal como acima mencionado, sendo que estas têm um papel fundamental no que concerne à ligação com um dado nodo vizinho. Assim, cada *thread*, tem a conexão TCP/IP com um e um só determinado nodo vizinho, tratando de enviar e receber os dados de e para esse nodo vizinho.

4.1 Tabela de Encaminhamento

Além do mencionado, podemos dar ênfase nos routers, nomeadamente na já mencionada tabela de encaminhamento, uma vez que estas, após o estabelecimento das rotas, são fundamentais na troca de informação entre nodos, mas acima de tudo no pedido e entrega da stream quando é realizado um pedido pela mesma. Com isto e a título de exemplo, o formato da tabela de encaminhamento para um router que possui como nodos vizinhos cujos ips são 10.0.0.20, 10.0.3.2 e 10.0.2.1 e este se insere na topologia que tem na camada *overlay* como servidores cujos ips são 10.0.11.10 e 10.0.1.10 é a seguinte:

```
+----------------------+----------------+----------+--------------+------------+
|  Interface de Saída  |  Atraso (ms)   |  Saltos  |   Servidor   |   Estado   |
+----------------------+----------------+----------+--------------+------------+
|      *10.0.3.2       |      191       |    2     |  *10.0.1.10  |  Inativa   |
|       10.0.2.1       |      230       |    3     |  *10.0.1.10  |  Inativa   |
|       10.0.2.1       |      804       |    2     |  10.0.11.10  |  Inativa   |
|       10.0.3.2       |      826       |    3     |  10.0.11.10  |  Inativa   |
+----------------------+----------------+----------+--------------+------------+

+------------------------------+
| * -> Melhor rota/servidor    |
| Token = 511552166            |
+------------------------------+
```

Figura 4.1: Tabela de Encaminhamento

Através da tabela de encaminhamento, tal como indicado na imagem, o router sabe diversas rotas para os diversos servidores presentes na camada *overlay*, no qual através da indicação de qual é o melhor servidor, sabe qual a melhor rota associada a este tal como indicado. De notar que, os estados e métricas de cada rota não são constantes sendo atualizadas através de mensagens com informação relevante que são difundidas na topologia, bem como qual o melhor servidor.

5. Testes e resultados

Uma vez detalhada toda a arquitetura, especificação dos protocolos e implementação, resta agora abordar os cenários de teste e fazer uma introspeção dos mesmos. Para tal, iremos mostrar o seguimento dos vários processos que acontecem no decorrer da execução da solução, para a camada *overlay* assente na camada *underlay* seguintes.

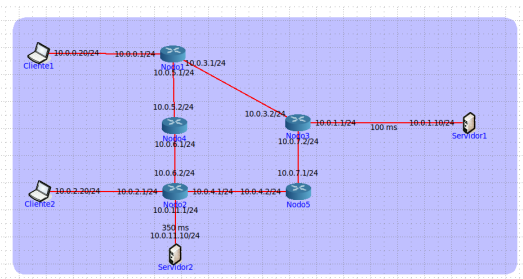

Figura 5.1: Camada *underlay*

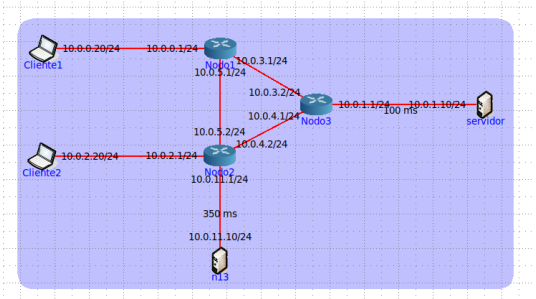

Figura 5.2: Camada *overlay*

Com isto, tal como descrito teremos da seguinte forma ordenada e através da numeração das imagens os seguintes eventos:

1. Routers aguardam pela chegada de pacotes provenientes do início de sessão por parte dos servidores, por forma a construirem as suas tabelas de encaminhamento;

2. Servidores difundem (periodicamente) pacotes de tipo 1 para a rede sendo recebidos pelos routers que realizam o inundamento controlado dos pacotes pela rede, construindo as suas tabelas de encaminhamento;

3. Cliente conecta-se à rede, pedindo a stream enviando um pacote de tipo 2 que é reencaminhado até ao melhor servidor da topologia;

4. Outro cliente conecta-se à rede, pedindo a stream enviando um pacote de tipo 2 que é reencaminhado pela rede, que caso chegue a um router que já entregue pacotes da stream, o serve em UNICAST;

5. Condições de entrega degradam-se, mudando o melhor servidor;

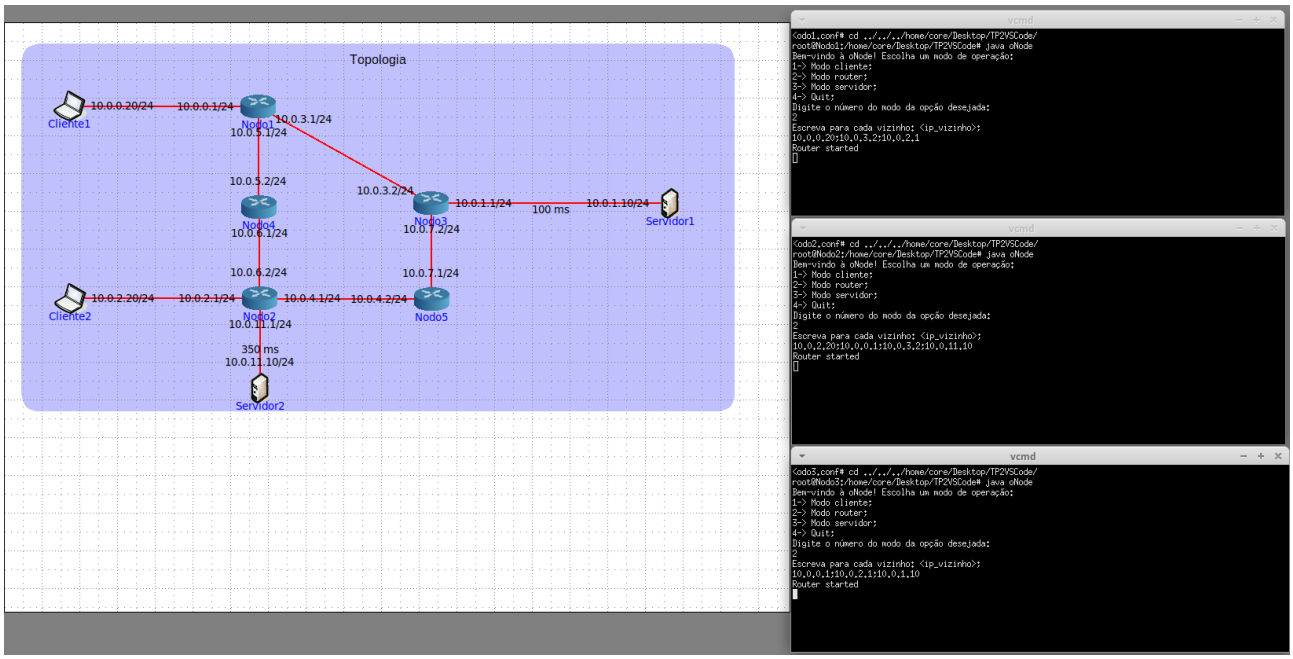

Figura 5.3: Imagem 1

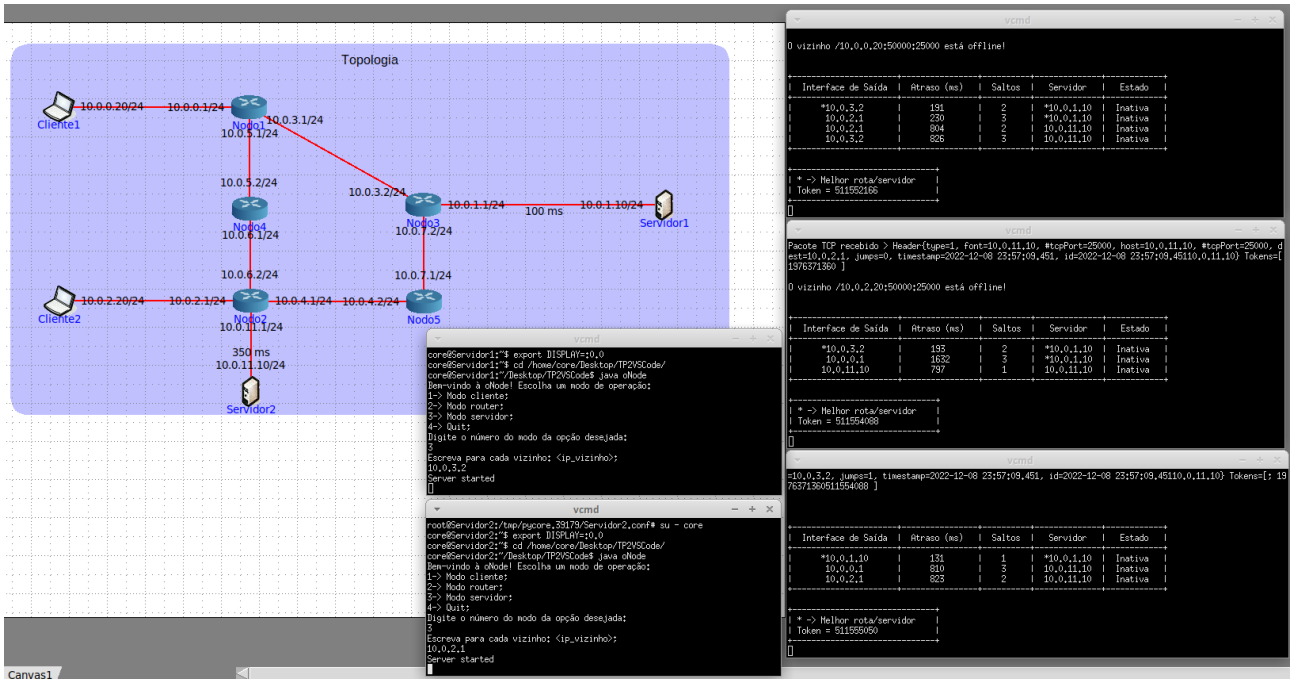

Figura 5.4: Imagem 2

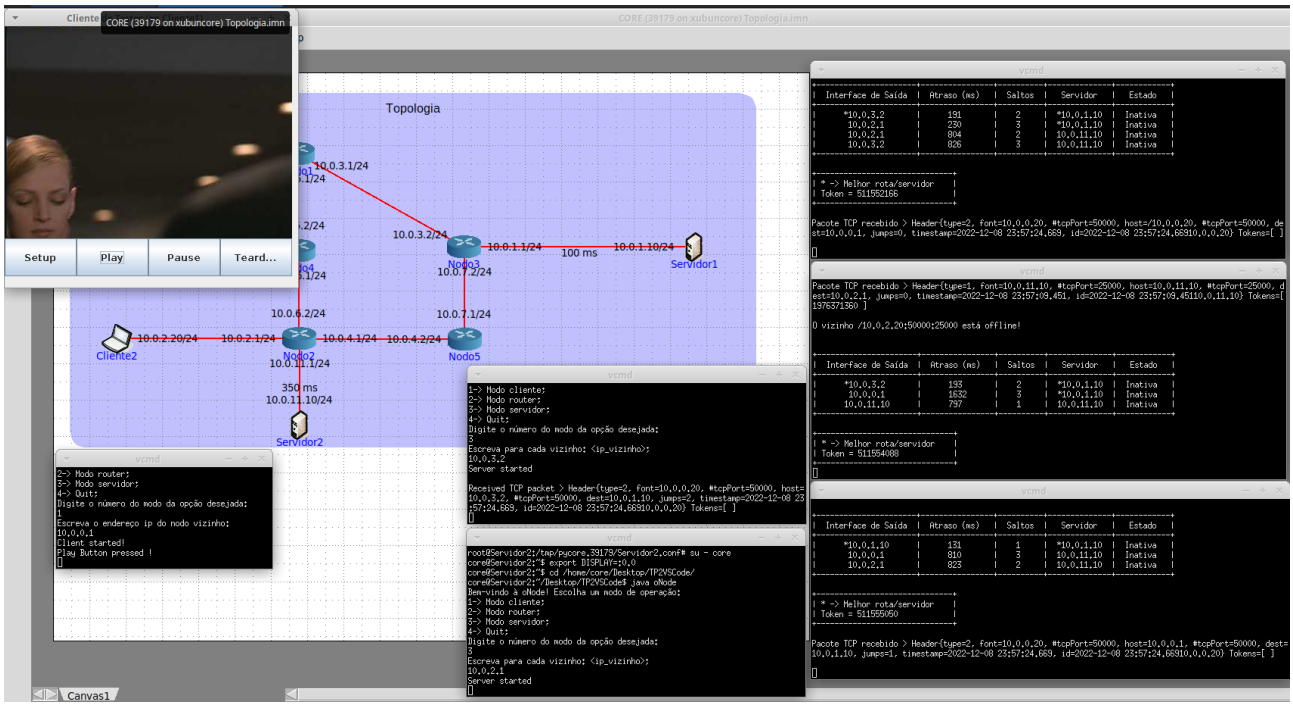

Figura 5.5: Imagem 3

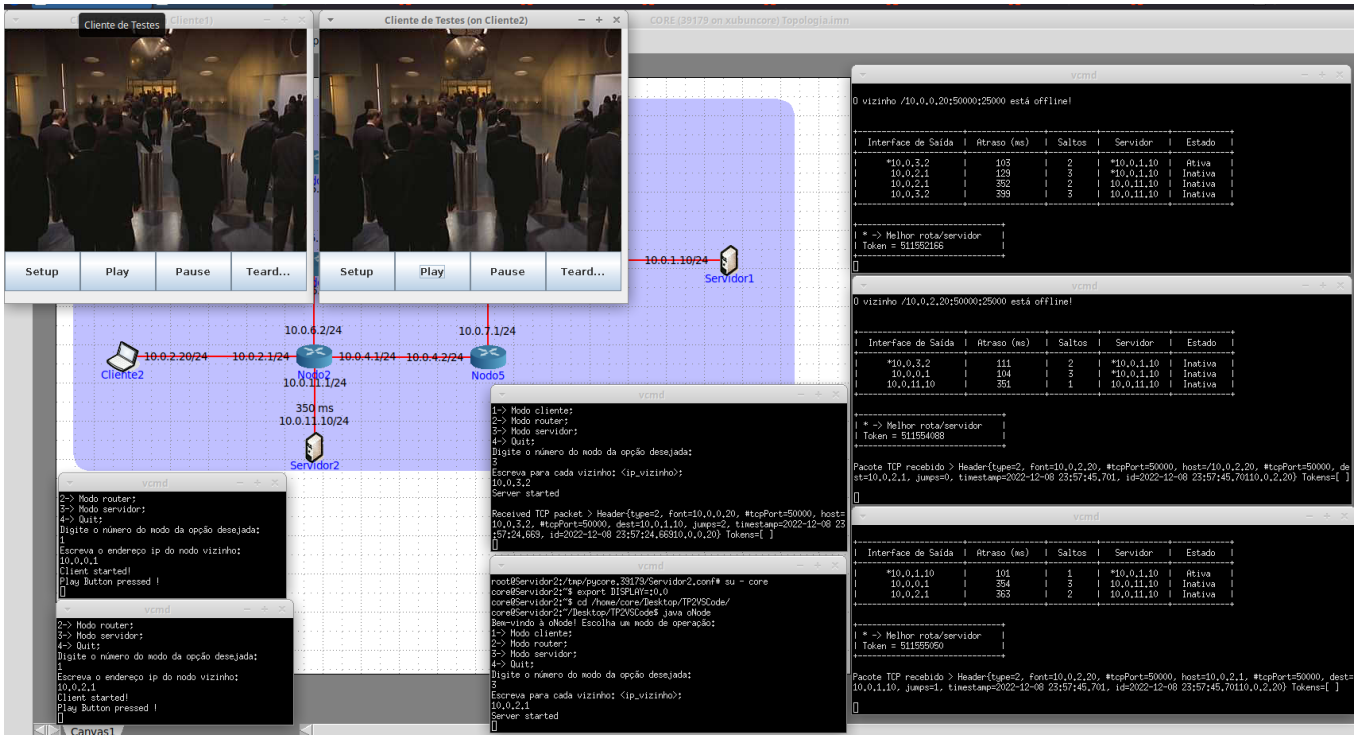

Figura 5.6: Imagem 4

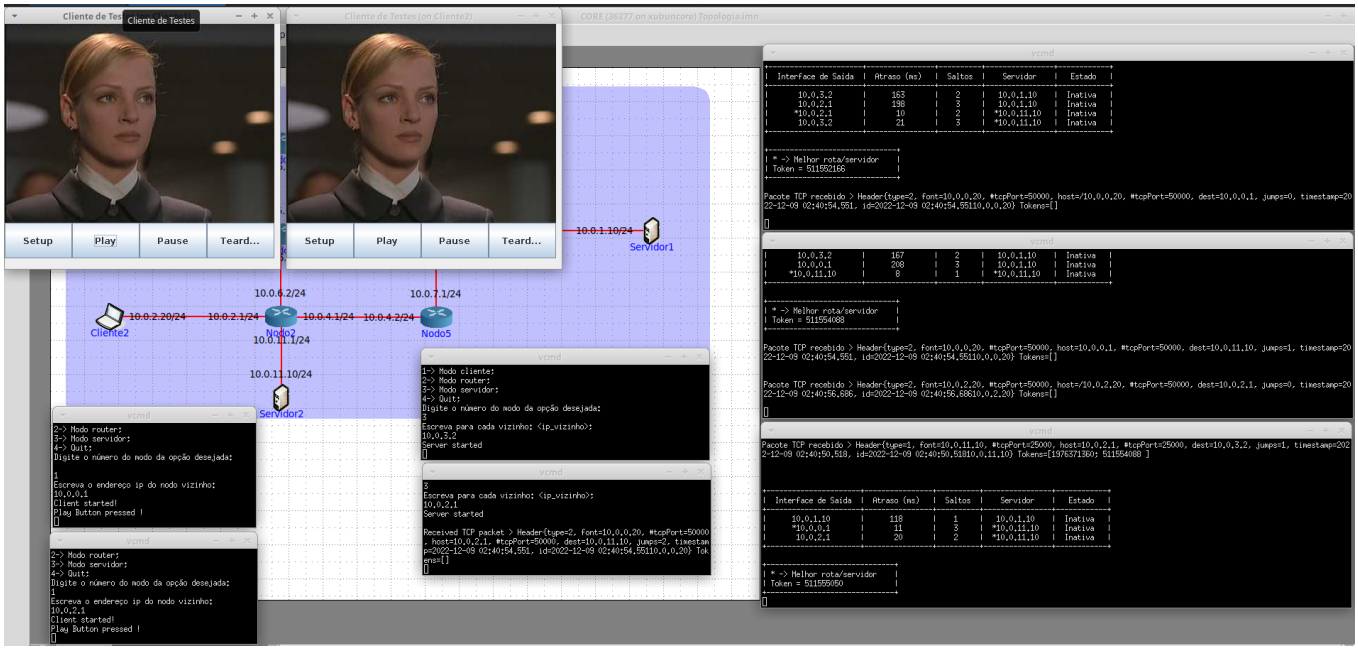

Figura 5.7: Imagem 5

6. Conclusões e trabalho futuro

A realização deste projeto permitiu consolidar noções acerca de entrega de áudio/vídeo/texto com requisitos de tempo real, a partir de um servidor de conteúdos para um conjunto de N clientes. Acima de tudo, permitiu dar um visão sobre a distinção das camadas *underlay* e *overlay* e a aplicação desta última no que concerne a entrega de stream. Permitiu também abordar conceitos de UNICAST no mesmo âmbito, mas também perceber como tirar proveito dos protocolos de transporte por forma a atingir o objetivo proposto, de forma inteligente, eficiente e fazendo uma ótima gestão dos recursos disponíveis.

Do ponto de vista de escalabilidade, a solução final desenvolvida, tem em vista e está preparada para funcionar a larga escala, isto é, independentemente da topologia em que assente e para uma grande camada *overlay*, isto é, com um elevado número de nodos clientes e nodos routers, realizar o seu propósito.

Por fim, uma vez que a solução final foi implementada com sucesso e na integra, concluímos que podemos retirar um balanço positivo deste projeto já que foi essencial para o nosso conhecimento pois adquirimos boas bases para trabalhos futuros.