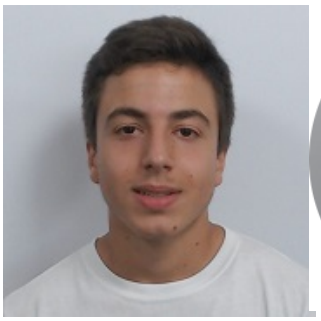
Universidade do Minho
Escola de Engenharia

Engenharia de Serviços em Rede

Serviço Over the Top para entrega de multimédia

PL3 - Grupo 5
Luis Sousa a89597
Maria Barros pg47488
Pedro Barbosa pg47577



a89597

pg47488

pg47577

29 de dezembro de 2021

Conteúdo

1 Introdução

Ao longo do último meio século de vida da Internet (a rede das redes), observou-se uma mudança irreversível de paradigma. A comunicação extremo-a-extremo, de sistema final para sistema final, dá lugar ao consumo voraz de conteúdos de qualquer tipo, a todo o instante, em contínuo e muitas vezes em tempo real. Este novo padrão de uso coloca grandes desafios à infraestrutura IP de base que a suporta. Apesar de não ter sido originalmente desenhada com esse requisito, tem sido possível resolver a entrega massiva de conteúdos com redes sofisticadas de entrega de conteúdos (CDNs) e com serviços específicos, desenhados sobre a camada aplicacional, e por isso ditos Over the Top (OTT).

Over-the-top é um termo genérico para um serviço utlizado sobre uma rede que não é oferecido pelo operador. É normalmente descrito como "over-the-top", pois estes serviços funcionam sobre um serviço já existente e não requerem qualquer afiliação com tecnologias ou modelos de negócio associadas ao operador de telecomunicações. Pode também ser definido como contéudo multimédia (televisão e conteúdo vídeo, por exemplo) distribuídos através de uma ligação Internet de alta velocidade e não de um provedor de serviços por cabo ou satélite.

A aplicação mais comum do modelo OTT é, sem dúvida, no dominio de vídeo digital, onde os fornecedores de conteúdo dependem de um operador de telecomunicações para disponibilizar conteúdo (muitas vezes interativo) para televisões, box por cabo e computadores pessoais. [1]

Um serviço de multimedia OTT, pode por exemplo usar uma rede overlay aplicacional, devidamente configurada e gerida para contornar os problemas de congestão e limitação de recursos da rede de suporte, entregando em tempo real e sem perda de qualidade os videos diretamente ao cliente final. Para tal, formam uma rede overlay própria, assente em cima dos protocolos de transporte (TCP ou UDP) e/ou aplicacionais (HTTP) da Internet. Neste trabalho pretende-se conceber e prototipar um desses serviços, que promova a eficiência e a otimização de recursos para melhor qualidade de experiência do utilizador.

2 Arquitetura da solução

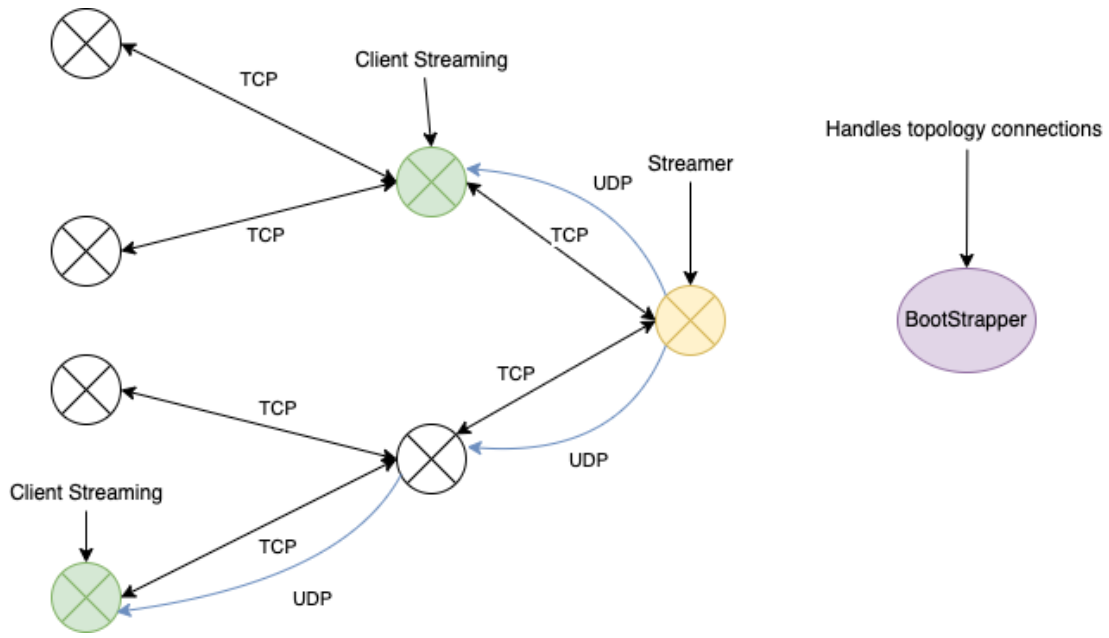


Figura 1: Diagrama : Arquitetura do Sistema

Inicialmente, começamos por definir qual a linguagem de programação que iremos usar para desenvolver o projeto. Assim, chegando a um consenso, escolhemos Java. Para além de já termos feito bastantes projetos com esta mesma linguagem, o que nos facilitará em algumas partes do trabalho, acreditamos que Java seja a linguagem mais apropriada para termos uma boa prestação neste trabalho prático.

Posteriormente, surgiu-nos um desafio de definir que protocolos iremos utilizar ao longo do projeto. Assim sendo, debatemos entre os elementos do grupo, e chegamos à conclusão que iremos utilizar o protocolo *TCP* para a construção da rede, isto é, construção das conexões entre nodos e servidor e envio de algumas mensagens entre eles. Para a parte restante do trabalho, ou seja, etapa em que iremos nos preocupar com streaming de video iremos usar protocolos *UDP*.

Seguidamente, o nosso grupo pensou na estratégia a utilizar na etapa 1 referente à construção da topologia overlay. Deste modo, decidimos escolher a segunda estratégia que consiste na conceção de um *bootstrapper* que irá controlar toda esta construção. Assim, este lê um ficheiro *.txt* com a configuração da topologia e, a partir do mesmo, trata das conexões para a construção da mesma. Desta forma, foi criado um *BootstrapperListener* que recebe a lista de nodos necessários para a criação da topologia e cria uma *thread* para cada um deles que vai indicar a cada nodo quantos vizinhos o mesmo possui e quais os seus ips. Em relação à conexão em si, cada nodo irá tentar estabelecer uma com os seus vizinhos. Caso consiga, a conexão estabelecida entre eles é do tipo *TCP*. Aquando da criação de um nodo é também criada uma *thread*, *OttNodeListener*, que estará sempre à espera dos vizinhos que ainda não se conectaram e que o queiram fazer.

O próximo passo consistiu na construção das rotas para os fluxos. Começamos por escolher o número de saltos como tipo de métrica para a escolha do melhor caminho. Seguidamente, pensamos sobre este passo e decidimos adotar a primeira estratégia. Sendo a topologia toda construída, o *bootstrapper* avisa o *streamer* que isso aconteceu. De seguida, este irá enviar um *packet* para os seus vizinhos com o número de saltos a 1. Quando cada nodo recebe este *packet* se este número de saltos for menor que o número de saltos do nodo, este irá mudar o seu número de saltos para esse novo número e irá retransmitir o *packet* para os seus vizinhos, e assim sucessivamente. Caso contrário, irá apenas descartar o *packet*.

Para a quarta e última etapa escolhemos a primeira estratégia. Desta forma, o *streamer* começa a enviar frame a frame o vídeo que pretende *streamar* encapsulado num *rtpPacket* que, por sua vez, é encapsulado num *DatagramPacket* e envia-o para o/os nodo/os vizinho/os consoante as melhores rotas escolhidas através da etapa anterior. Estes vizinhos voltam a retransmitir o packet para outros vizinhos da mesma forma. Caso queiram consumir a *stream* irão desencapsular os packets e transmitir frame a frame a mesma.

3 Especificação dos protocolos

3.1 Formato das mensagens protocolares

O tipo de mensagens protocolares do nosso projeto difere com o tipo de *packet* utilizado. Existem quatro tipos diferentes:

- ***Packet.START_FLOODING***

  Este *packet* é enviado pelo *bootstrapper* ao *streamer* em específico, no final da construção da topologia, de modo a este saber que pode iniciar o *flooding*.

- ***Packet.FLOODING***

Este *packet* é enviado de nodo em nodo aquando do *flooding*. Um dos campos deste *packet* é uma mensagem que contém o número de saltos.

- ***Packet.CANCEL_STREAM_FLOW***

  Este *packet* é enviado aquando do *flooding* para um nodo avisando-o do redirecionamento da *stream* e que esta já não passar por lá.

- ***Packet.CONFIRM_STREAM_FLOW***

  Este *packet* é enviado aquando do *flooding* para um nodo avisando-o do redirecionamento da *stream* e que esta começa a passar por lá.

3.2 Interações

Existem as seguintes interações no serviço OTT para entrega de multimédia:

- ***Bootstrapper-Streamer***

  No final da construção da topologia avisa o *streamer* que este pode iniciar o *flooding*.

- ***Bootstrapper-Nodo OTT***

  Interage com o Nodo OTT para a inicialização de conexão entre eles e outros Nodos OTT.

- ***Nodo OTT - Nodo OTT***

  Interagem entre si tanto para a transmissão de *packets* de *flooding* como para a retransmissão de *packets* de *streaming*.

4 Implementação

4.1 *Package Bootstrapper*

Este *Package* contém as classes referentes ao *Bootstrapper*.

- **Classe *Bootstrapper***

  Esta classe tem como atributos o *IP* do *Bootstrapper*, a porta pela qual este vai comunicar e uma *string* que é o ficheiro de configuração da topologia. Esta classe espera pela conexão de todos os nodos necessários para arrancar a topologia. Quando isto acontece, é chamada a função *start_flooding*, que envia um *packet* ao *streamer* para este iniciar o *flooding*.

- **Classe *Topology***

  Esta classe tem como atributos um *Map* que associa um nome de um nodo a uma lista de IPs dos seus vizinhos e uma lista com os nomes dos nodos necessários para a criação da topologia. Nesta classe é feito o *parsing* do ficheiro de configuração da topologia.

4.2 *Package OttNode*

Este *Package* contém as classes referentes aos nodos *OttNode*.

- **Classe *Neighbor***

  Esta classe tem como atributos um *Socket*, um *boolean active* que indica se um vizinho se encontra ou não ativo, uma *thread*, *receiver_thread*, que irá tratar daquilo que o nodo recebe e uma, *sender_thread*, que irá tratae daquilo que envia e uma *Blocking_Queue* de *packets* à espera de serem enviados. Nesta classe existe uma função, *register_established_connection* que regista uma conexão criada.

- **Classe *OttNode***

  Esta classe tem como atributos uma *string* com o *IP* do *streamer*, três *ints* que representam a porta de *stream udp*, a porta de comunicação com o nodo vizinho e a porta de comunicação com o *bootstrapper* respetivamente, um *ServerSocket*, um *NodeType* que representa o tipo do nodo, um *boolean consuming* que indica se o nodo pretende consumir a *stream* ou não, um *set* de *InetAddress* com os *IPs* dos vizinhos para onde serão retransmitidos os *packets* de streaming, um *InetAddress* com o *IP* do nodo de onde a *stream* vem, um *int* que representa o número de saltos, um *ConcurrentMap* com o *IP* dos vizinhos, uma *BlockingQueue* de *packets* recebidos e uma *StreamWindow*. Como funções mais importantes de destacar tem, get_neighbors_from_bootstrapper, através da qual o nodo sabe quais são os seus vizinhos, handle_flooding_packet, onde se lida com os *packets* provenientes do *flooding*, inicia classes para todas as *threads* utilizadas e ainda código onde é feito o encapsulamento dos *packets* de streaming e destes mesmos em *packets* UDP.

- **Classe *OttNodeListener***

  Esta classe tem como atributo um *ServerSocket*. Esta classe fica à espera de conexões de um nodo com os seus vizinhos e regista-as.

- **Classe *OttNodeUdpListener***

  Esta classe tem como atributos um *DatagramSocket*, um *Timer*, um *Lock* e uma *Condition*. Um *OttNodeListener* fica à espera de receber *packets* UDP, nomeadamente *packets* de streaming. Também nesta classe é feito a retransmissão dos mesmos para nodos vizinhos. Caso o nodo queira consumir a stream, também é nesta classe que é feito o desencapsulamento do *packet* de *streaming*.

- **Classe *StreamWindow***

  Esta classe tem como atributos um *JFrame*, um *JPanel*, uma *JLabel* e um *ImageIcon*. Nesta classe é criada uma StreamWindow.

4.3 *Package utils*

Este *Package* contém as classes auxiliares que são usadas por ambos os *Packages* acima referidos.

- **Classe *Packet***

  Esta classe tem como atributos um *PacketType*, um *InetAddress* e uma *string*. Nesta classe é criado um *packet*.

- **Classe *RtpPacket***

  Esta classe tem como atributos principais um *array* de *bytes* com o *header* e outro com o *payload* e um *int* que representa o tamanho do mesmo. Nesta classe é criado um *packet*, *RtpPacket*, onde se encapsula os dados do ficheiro de vídeo.

- **Classe *VideoStream***

  Esta classe tem como atributos *MAX_FRAME_SIZE*, *FRAME_PERIOD*, *VIDEO_LENGTH*, *VIDEO_FILENAME*, *VIDEO_HEIGHT* E *VIDEO_WIDTH*. Esta classe cria um *VideoStream*.

5 Testes e resultados

De forma a testar as capacidades da nossa aplicação, decidimos usar 3 cenários diferentes.

5.1 Cenário 1

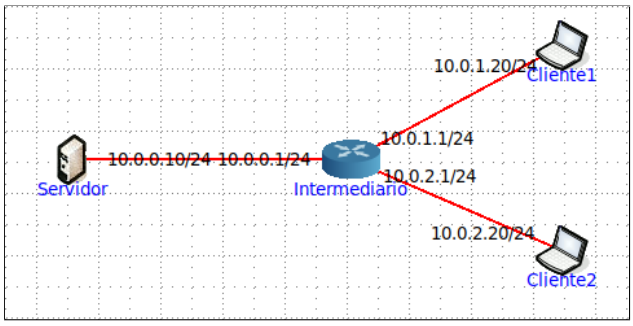


Figura 2: Cenário 1

Na figura seguinte podemos observar o momento após todos os nodos da topologia se terem conectado ao *Bootstrapper*.

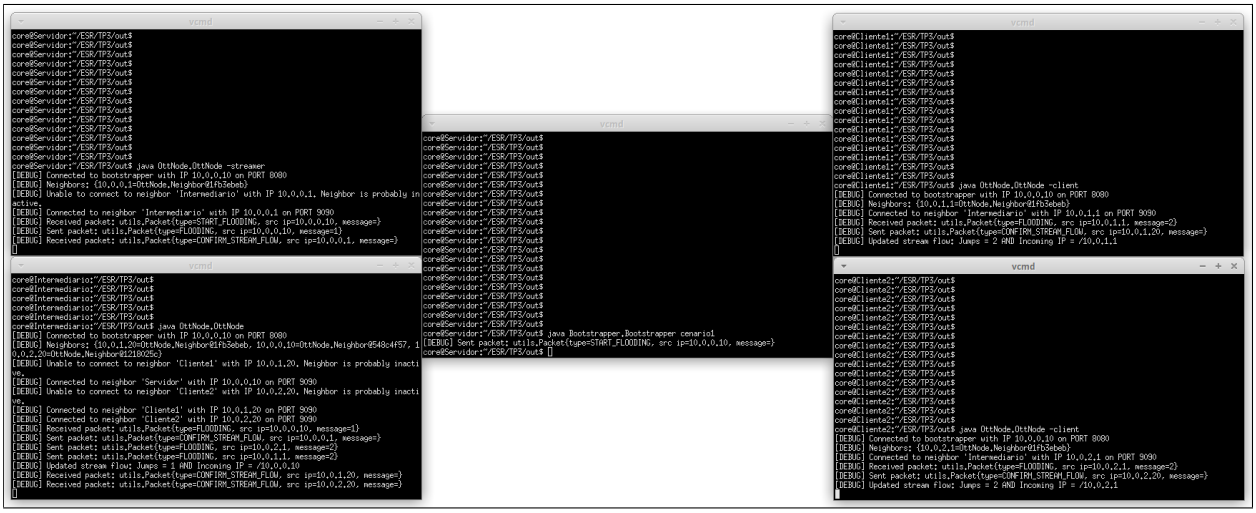


Figura 3: Nodos conectados no cenário 1

Na figura seguinte podemos observar um cliente a consumir a *stream*.

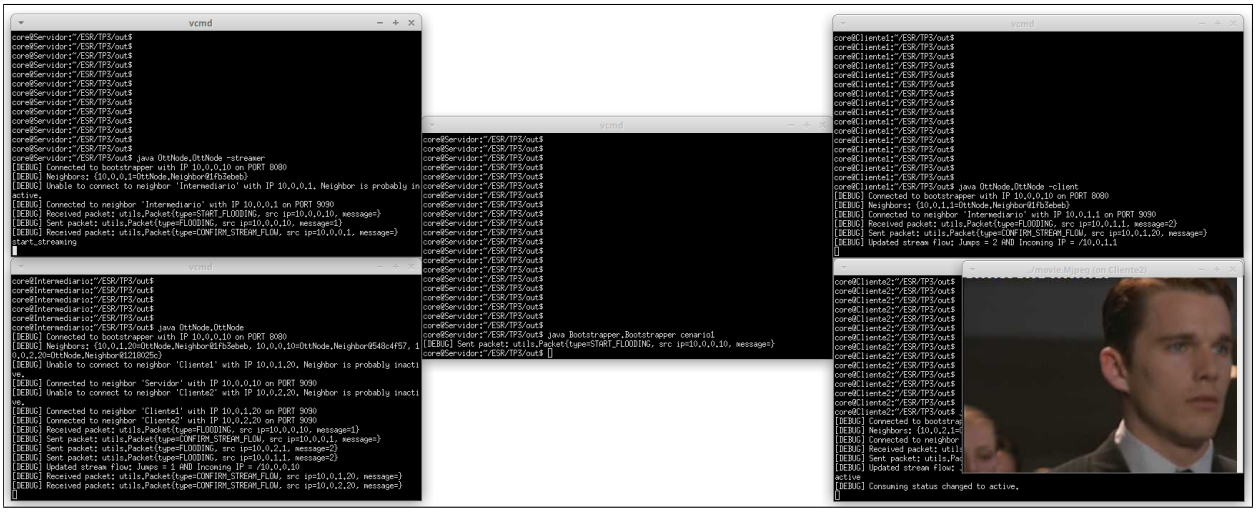


Figura 4: Cliente a consumir a *stream* no cenário 1

5.2 Cenário 2

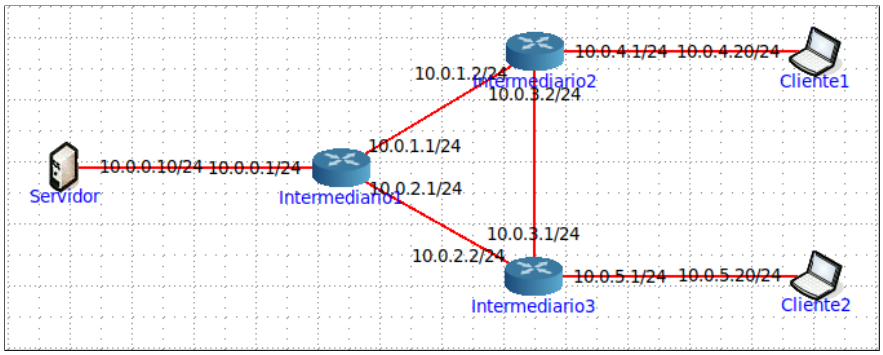


Figura 5: Cenário 2

Na figura seguinte podemos observar dois clientes a consumir a *stream*.

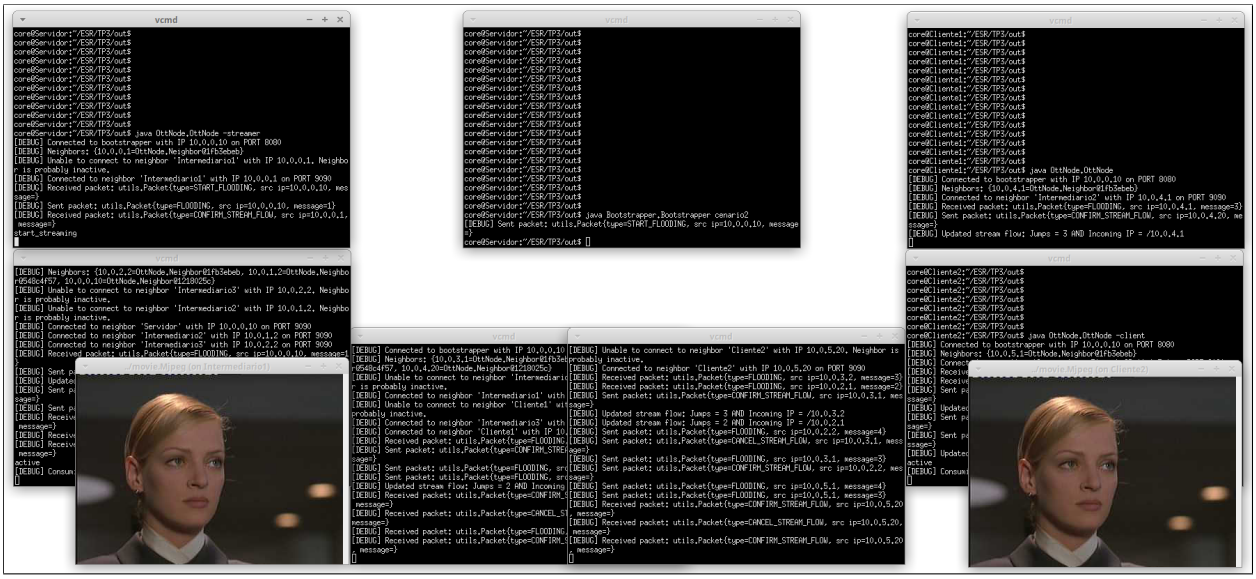


Figura 6: Dois clientes a consumir a *stream* no cenário 2

5.3 Cenário 3

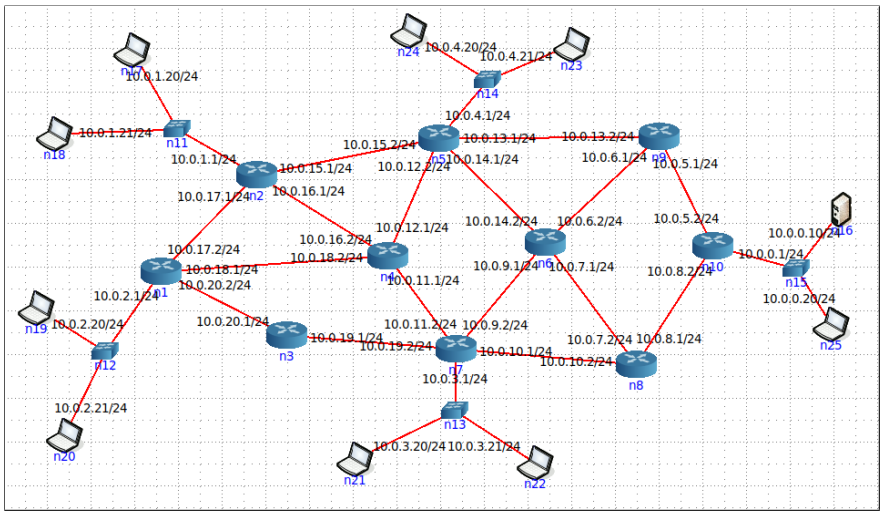


Figura 7: Cenário 3

Na figura seguinte podemos observar alguns clientes a consumir a *stream*.

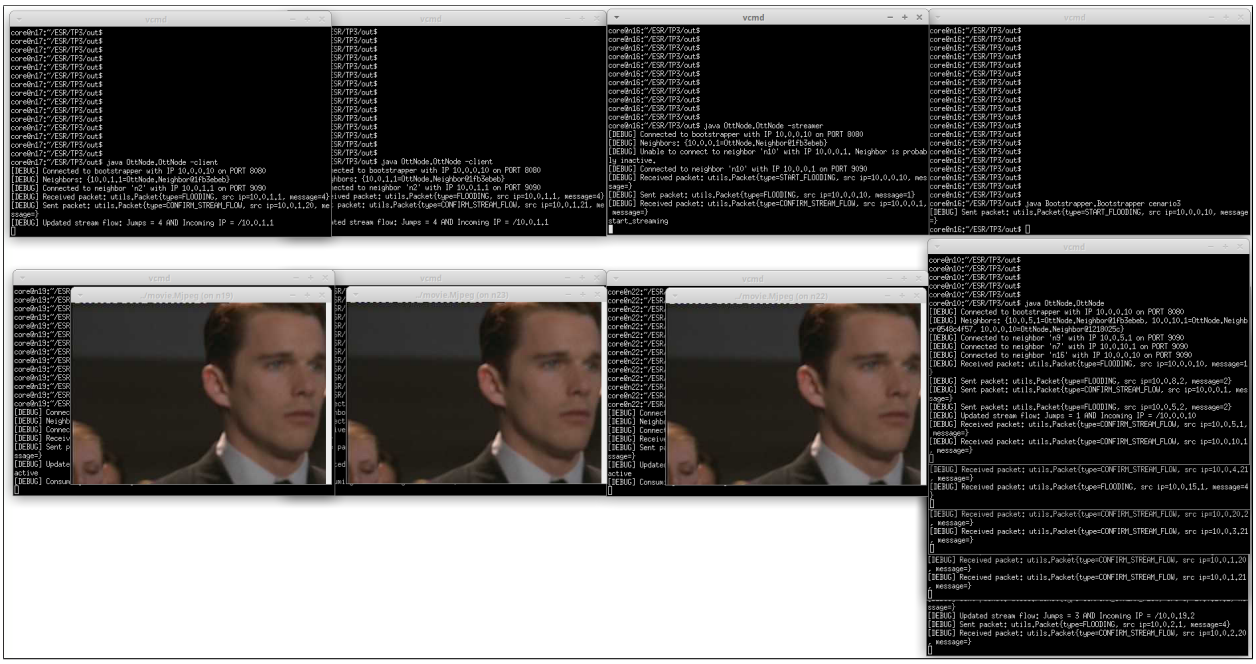


Figura 8: Alguns clientes a consumir a *stream* no cenário 3

6 Conclusão

Em jeito de conclusão, o grupo sente-se satisfeito com o resultado final obtido e também em ter conseguido ultrapassar vários obstáculos com que se deparou ao longo do projeto. Ademais, pensamos que os objetivos principais do trabalho prático propostos pelos docentes da unidade curricular de Engenharia de Serviços em Rede foram atingidos com sucesso, uma vez que conseguimos pôr em prática os conhecimentos obtidos nas aulas e ainda aqueles que aprendemos de forma autodidata. Deste modo, fomos capazes de desenvolver um Serviço *Over The Top* para efeitos de *streaming* de vídeo em tempo real através de conexões TCP e UDP.

Referências

[1] https://repositorio-aberto.up.pt/bitstream/10216/106138/2/203502.pdf