Universidade do Minho
Departamento de Informática

Engenharia de Serviços em Rede

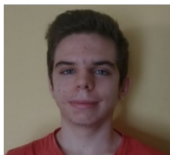
TP2 - Serviço Over the Top para entrega de multimédia

Grupo 7.4

Feito por:
Dinis Gonçalves Estrada (PG53770)
Emanuel Lopes Monteiro da Silva (PG53802)
Simão Pedro Cunha Matos (PG54239)


PG53770


PG53802


PG54239

December 6, 2023

Ano Letivo 2022/23

Conteúdo

1 Arquitetura da Solução **3**

2 Especificação dos protocolos **4**
2.1 Formato das mensagens protocolares 4
2.1.1 RTSPpacket 4
2.1.2 RTPpacket 4
2.2 Interações 5
2.2.1 Inicialização 5
2.2.2 Pacotes de Controlo 5

3 Implementação **9**
3.1 Construção da topologia overlay 9
3.2 Aplicação oNode 9
3.3 Bibliotecas utilizadas 9

4 Testes e Resultados **10**

Lista de Figuras

1 Inicialização de Clientes e Nós. 5
2 Inicialização de Servidores. 5
3 Mensagem SETUP de cliente 1. 6
4 Mensagem SETUP de cliente 2. 6
5 Mensagem PLAY de cliente 1. 7
6 Mensagem PLAY de cliente 2. 7
7 Mensagem PAUSE de cliente 1. 8
8 Mensagem TEARDOWN de cliente 2. 8
9 Rede overlay. 10
10 Teste com conversations do nodo n5. 11

1 Arquitetura da Solução

A solução final obtida pelo grupo tem por base as várias estratégias apresentadas pela equipa docente no decorrer das aulas tal como aquelas apresentadas no enunciado do presente trabalho prático.

Primeiramente, definimos que a linguagem de programação a utilizar para este projeto seria Python, isto porque é uma das linguagens em que o código fornecido pela equipa docente está disponível e também por ser a linguagem com que o grupo está mais familiarizado. Após isto, criamos 3 cenários de teste no emulador CORE. Definimos que o protocolo de tranporte a ser utilizado seria UDP, uma escolha fundamentada pelo facto de ser o mais adequado para streaming em tempo real. Isso deve-se à sua capacidade de enviar pacotes a uma taxa constante, independentemente de congestionamentos na rede ou da capacidade da aplicação de recebê-los, ao contrário do TCP. O UDP, sendo um protocolo simples que não garante a entrega das mensagens, elimina a necessidade de atrasos para retransmissão de pacotes, ao contrário da confiabilidade oferecida pelo TCP. Essas características tornam o UDP um protocolo apropriado para aplicações que requerem execução em tempo real.

Na segunda etapa foi necessário decidir qual a estratégia para a contrução desta rede. O grupo optou por uma abordagem baseada num controlador, isto é, indicar apenas um nó (bootstraper) como contacto para arranque da rede.

Na seguinte etapa, o grupo implementou um cliente e um servidor, usando como base no código fornecido, adaptando-o. Nesta fase, o cliente já é capaz de receber pacotes da rede overlay, com um número de sequência, e reproduzir o vídeo numa janela.

Depois, tratamos da etapa da monitorização dos servidores de conteúdos. Os servidores irão enviar mensagens de prova ao RP, de forma a obter um conhecimento atualizado das condições de entrega de cada um dos servidores. O pacote possui o endereço IP do servidor de origem e um carimbo de tempo indicando quando foi criado. Ao atingir o RP, este calcula o tempo atual e determina o intervalo decorrido desde a criação do pacote até a sua chegada.

A última etapa será de construção dos fluxos para a entrega de dados. Se um cliente quiser enviar uma mensagem, esta será enviada para o vizinho que se encontra na rota, previamente definida, até ao RP. Para cada nó que a mensagem atravessa, um fluxo é gerado. Isso implica que cada nó terá um fluxo dedicado para cada cliente que solicita serviços, e esse fluxo incluirá a próxima etapa até o cliente para o qual os pacotes RTP devem ser encaminhados.

2 Especificação dos protocolos

2.1 Formato das mensagens protocolares

2.1.1 RTSPpacket

Será responsável por implementar as mensagens de controlo na rede overlay, que serão Strings de tamanho máximo 250 bytes. Os seus 3 campos, que serão separados pelo caractér ";", são: Type, Payload e Padding. Este último é um campo apenas constituído por zero com objetivo de preservar o tamanho de todas mensagens. Iremos ter os seguintes tipos de mensagem:

- **HELLO-** Utilizada na ligação ao bootstrapper.
- **HELLORESPONSE-** Resposta do bootstrapper a mensagem HELLO, indicando a o seu próprio endereço e dos seus vizinhos no Payload.
- **SETUP-** Utilizada para informar que foi criado um fluxo de streaming.
- **PLAY-** Utilizada para informar que foi aberto um fluxo de streaming.
- **PAUSE-** Utilizada para informar que foi fechado um fluxo de streaming.
- **TEARDOWN-** Utilizada para informar que foi removido um fluxo de streaming.
- **PROBE-** Irá permitir obter um conhecimento atualizado das condições de entrega de cada um dos Servidore

2.1.2 RTPpacket

Será utilizado para transportar informações, em bytes, de áudio e vídeo em tempo real. É constituído pelo RTP HEADER e pelo RTP PAYLOAD. Os diferentes campos do cabeçalho são:

- **Version (V):** 2 bits que indicam a versão do RTP em uso. O valor comum é 2.
- **Padding (P):** 1 bit que indica se há bytes de preenchimento adicionados ao final do cabeçalho.
- **Extension (X):** 1 bit que indica se existe uma extensão ao cabeçalho padrão.
- **CSRC Count (CC):** 4 bits que especificam o número de identificadores de fontes contribuintes (CSRC) presentes no cabeçalho.
- **Marker (M):** 1 bit que pode ser usado por aplicativos específicos para sinalizar eventos especiais.
- **Payload Type:** 7 bits que identificam o tipo de dados de mídia transportados pelo pacote RTP.
- **Sequence Number:** 16 bits incrementados para cada pacote enviado, permitindo a reconstrução ordenada dos pacotes.
- **Timestamp:** 32 bits que fornecem a marca de tempo para a amostra de mídia no pacote, usado para sincronização.
- **SSRC (Synchronization Source Identifier):** 32 bits que identificam exclusivamente a fonte da mídia.
- **CSRC (Contributing Source Identifier):** Lista de identificadores de fontes contribuintes, cada um com 32 bits de compriment

2.2 Interações

2.2.1 Inicialização

Os clientes e os nós conectam-se à nossa topologia enviando uma mensagem HELLO ao bootstrapper, que lhes irá responder com uma mensagem HELLORESPONSE. Já no caso do servidor, este para além das trocas de mensagens de inicialização, também irá enviar um pacote de prova ao RP.

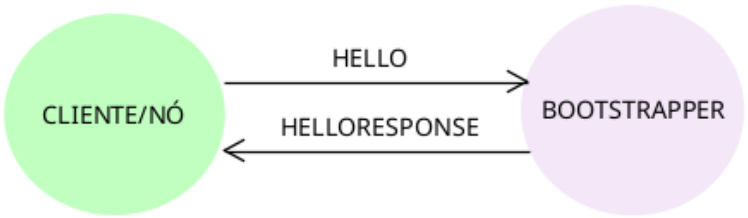


Figure 1: Inicialização de Clientes e Nós.

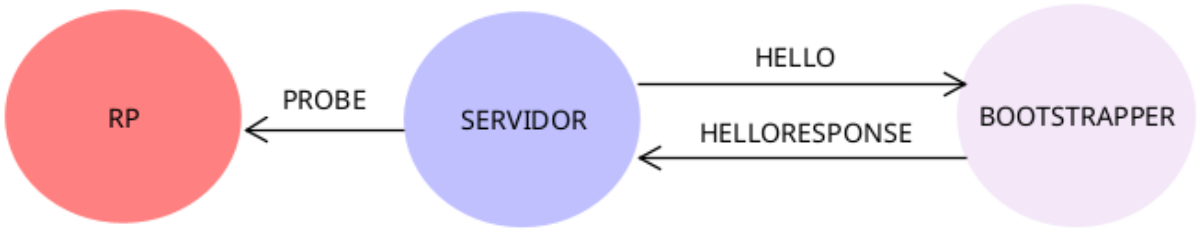


Figure 2: Inicialização de Servidores.

2.2.2 Pacotes de Controlo

Vamos agora demonstrar as interações que irão ocorrer quando os clientes enviam mensagens de SETUP, PLAY, PAUSE e TEARDOWN, após a topologia overlay estar construída. Ambos os clientes querem reproduzir o mesmo vídeo.

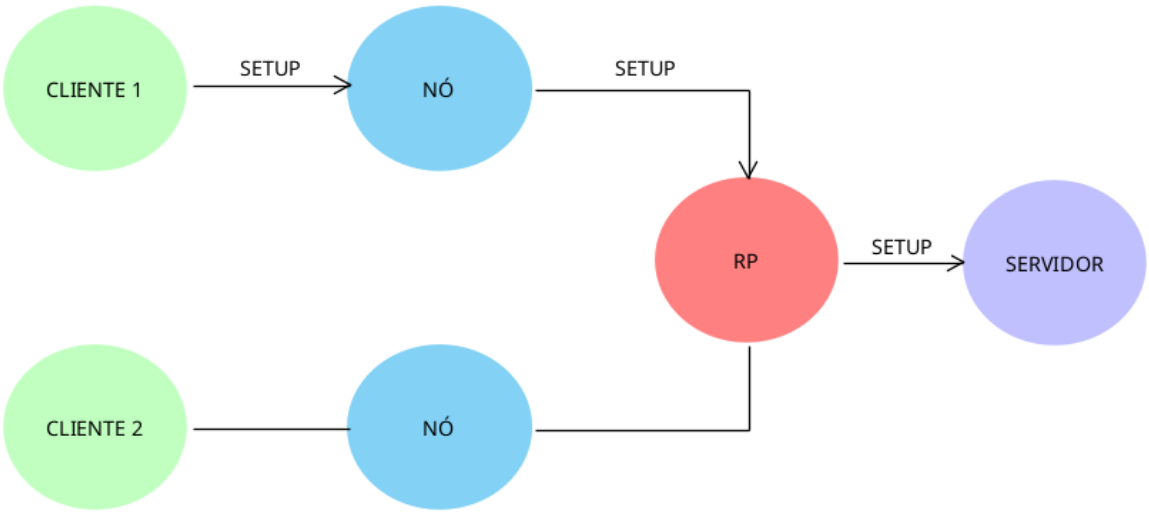


Figure 3: Mensagem SETUP de cliente 1.

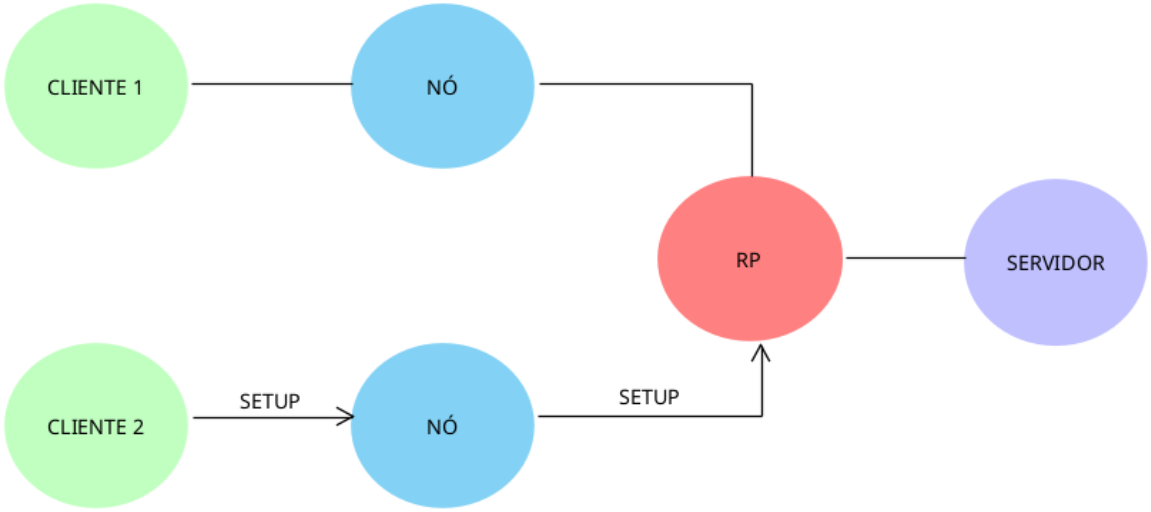


Figure 4: Mensagem SETUP de cliente 2.

Quando o Cliente 1 envia uma mensagem de SETUP, esta será reencaminhada até ao RP, que por sua vez a irá encaminhar até ao servidor. Como já foi criado um fluxo de stream entre o RP e o servidor, quando o Cliente 2 envia a mensagem de SETUP, esta será enviada apenas até ao RP.

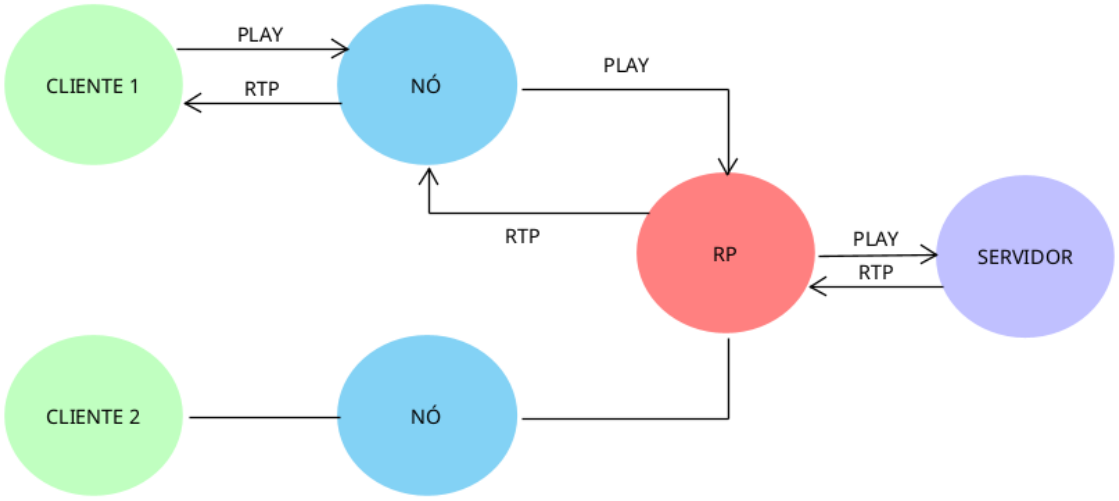


Figure 5: Mensagem PLAY de cliente 1.

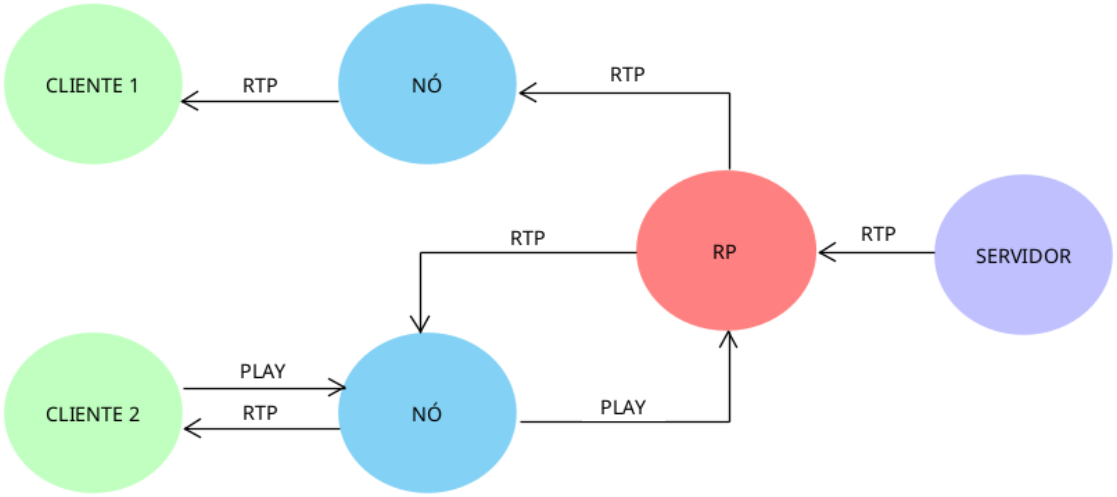


Figure 6: Mensagem PLAY de cliente 2.

Após isso, os clientes vão enviar mensagens de PLAY, indicando que desejam receber a stream. Novamente, no caso do Cliente 1 a mensagem será enviada até ao servidor, que vai começar a enviar pacotes RTP e no caso do Cliente 2 a mensagem será enviada apenas até ao RP, pois este já esta a receber os pacotes RTP.

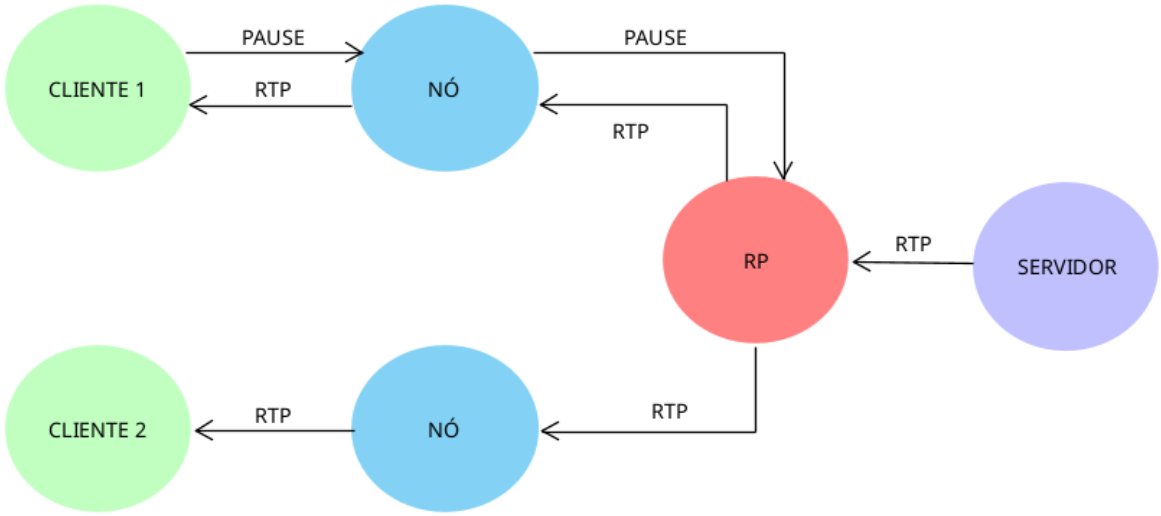


Figure 7: Mensagem PAUSE de cliente 1.

O Cliente 1 quer colocar a stream em pausa, logo irá ser enviada uma menagem de PAUSE, fechando o seu fluxo de stream. Como ainda existe um fluxo de stream aberto entre o Cliente 2 e o servidor, o RP apenas deixa de enviar os pacotes RTP para o Cliente 1.

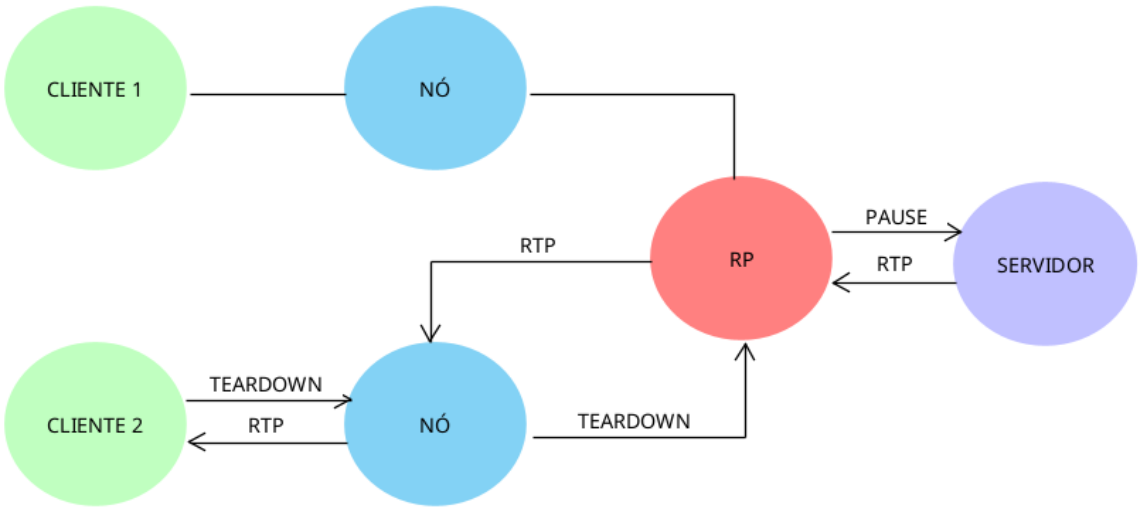


Figure 8: Mensagem TEARDOWN de cliente 2.

Se o Cliente 2 envia uma mensagem de TEARDOWN, indicando que quer eliminar o seu fluxo de stream, esta mensagem será reencaminhada até ao RP que, como ainda possui o fluxo fechado do Cliente 1 apenas envia uma mensagem de PAUSE ao servidor.

3 Implementação

3.1 Construção da topologia overlay

Como o grupo optou por uma abordagem baseada num controlador para construir a rede, é necessário criar manualmente um ficheiro de configuração que indique qual dos nós será RP e que contenha informações sobre os vizinhos de todos os nós na árvore:

```
RP-10.0.2.1

#10.0.3.10
10.0.3.1

#10.0.3.1
10.0.3.10
10.0.6.2
10.0.2.1

...
```

Como se pode verificar, utilizamos "RP-" para indicar o Rendezvous Point e # para indicar um nó, sendo as seguintes linhas o seus vizinhos. Também é importante notar que a ordem em que os vizinhos aparecem no ficheiro é relevante, pois o primeiro vizinho de cada nó será aquele que se encontra "no caminho" até ao RP.

3.2 Aplicação oNode

Será a partir desta módulo que os nós na árvore serão inicializados. Deverão ser utilizados os seguintes argumentos:

1. **Bootstrapper:** *-bs <config file>*
2. **Nó:** *-n <bootstrapper ip> <número de vídeos>*
3. **RP:** *-rp <bootstrapper ip> <número de vídeos>*
4. **Servidor:** *-s <bootstrapper ip> <vídeos>*
5. **Cliente:** *-c <bootstrapper ip> <rtp port> <rtsp port>*

Por padrão, para inicializar o cliente, as portas rtp e rtsp a utilizar são 6666 e 5555, respetivamente. Estas portas deverão ser incrementadas uma unidade dependendo de qual vídeo a ser transmitido e a sua posição como argumento na inicialização do servidor. Por exemplo, caso o servidor seja inicializado da seguinte forma, *-s <bootstrapper ip> videoA videoB*, o cliente deverá utilizar as portas 6667 e 5556, respetivamente, para reproduzir o videoB.

3.3 Bibliotecas utilizadas

- **socket:** Utilizada em programação de redes para comunicação entre processos em diferentes dispositivos, fornecendo funcionalidades para criação e manipulação de sockets.
- **threading:** Oferece suporte à programação concorrente, permitindo a execução paralela de tarefas.
- **datetime:** Facilita a manipulação de datas e horários em Python.
- **time:** Fornece funcionalidades relacionadas ao tempo, como medição e temporização.
- **sys:** Permite a interação com o interpretador Python e o sistema operacional.

- **PIL:** Biblioteca para manipulação de imagens em Python.
- **PyQt5:** Ferramentas para desenvolvimento de interfaces gráficas em Python.
- **os:** Facilita a interação com o sistema operativo.
- **random:** Geração de números aleatórios e operações relacionadas à aleatoriedade.
- **pickle:** Usado para serialização e desserialização de objetos Python.
- **traceback:** Fornece informações detalhadas sobre exceções.

4 Testes e Resultados

Para testar o nosso serviço de streaming, utilizando o emulador CORE, criamos a seguinte topologia:

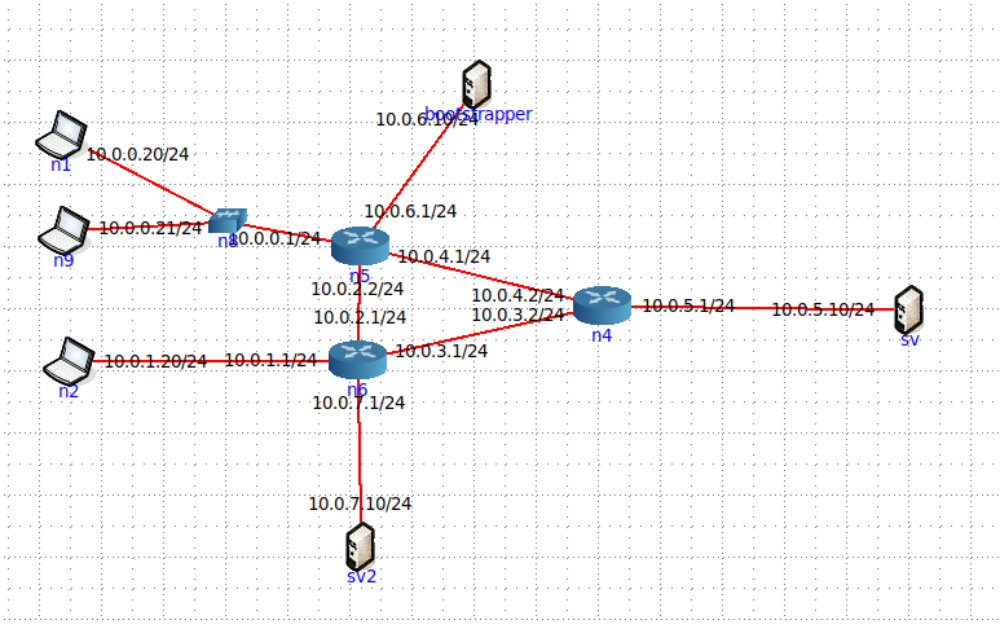


Figure 9: Rede overlay.

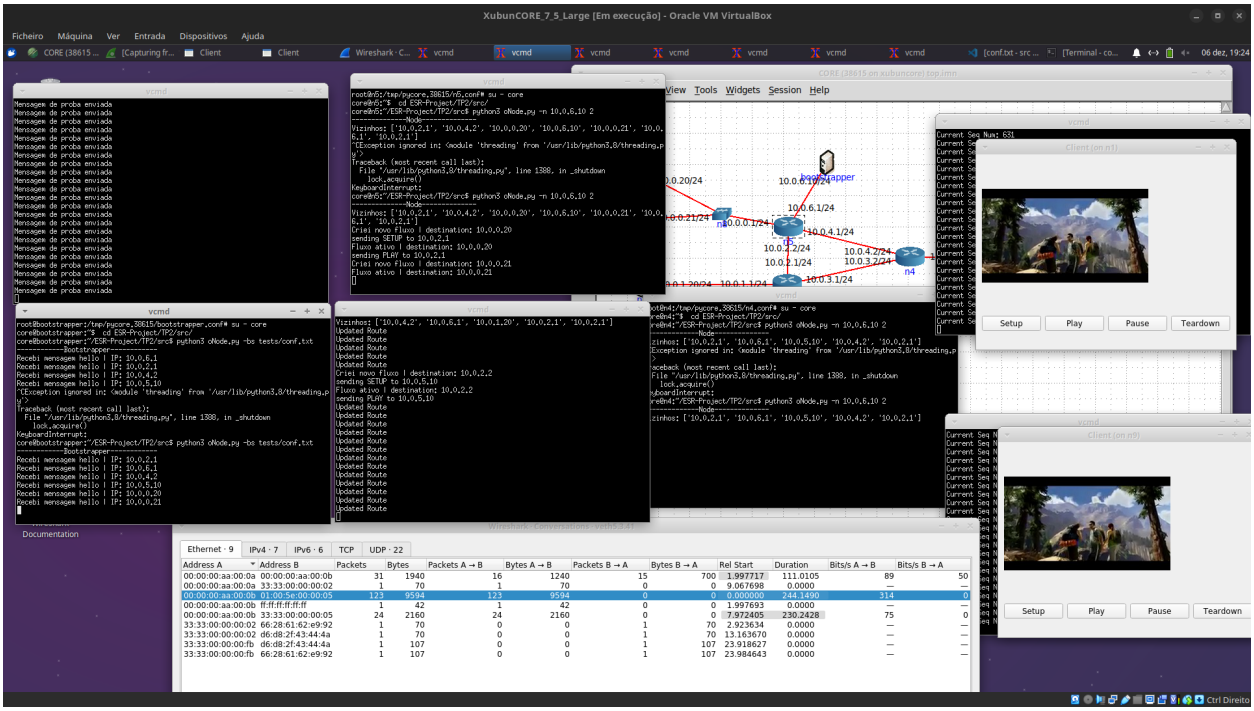


Figure 10: Teste com conversations do nodo n5.