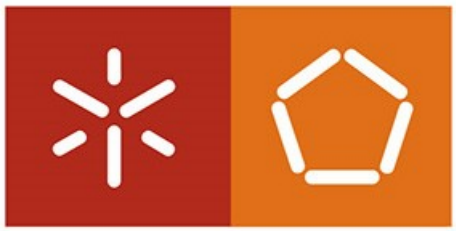

Universidade do Minho
Escola de Engenharia

Engenharia de Serviços em Rede-Serviço Over the Top para entrega de multimédia
Trabalho Prático 2-Grupo 72
2023/2024

Filipa Rebelo pg53624
Luís Fernandes pg54022
Luís Araújo pg54004

7 Dezembro 2023

1 Arquitetura da solução

Neste projeto optamos por utilizar a linguagem Java para a execução do mesmo, uma vez que é uma linguagem com a qual estamos familiarizados e já temos experiência em vários projetos anteriores.
Devido à natureza do projeto e à necessidade de incorporar um sistema de transmissão de dados em tempo real, decidimos que seria mais apropriado utilizar o protocolo de transporte UDP para as comunicações.
Optamos por utilizar este protocolo dada a sua rapidez no envio de pacotes em comparação com o protocolo TCP. Embora o UDP possa apresentar alguma perda de pacotes, essa perda não é considerável, e prevemos que não terá um impacto substancial na qualidade do streaming.

Na figura 1 encontra-se representado um diagrama ilustrativo da arquitetura da solução. É possível observar 4 threads em execução:
O primeiro thread envia os pacotes RTP recebidos dos nós vizinhos para o próximo nó no caminho para o destino (cliente). O segundo thread recebe os pacotes RTP vindos dos nós vizinhos. O terceiro thread envia as mensagens protocolares do protocolo MRP criado. O quarto thread recebe as mensagens protocolares do protocolo MRP criado. A tabela de encaminhamento contém informações sobre o próximo salto, tempo de atraso e os seus nós vizinhos.

Esta arquitetura permite que a rede funcione de maneira robusta e tolerante a falhas, já que cada nó mantém informações sobre rotas e tempos de atraso. Isso permite que a rede encontre caminhos alternativos para entregar pacotes de dados em caso de falha de algum nó ou caminho.

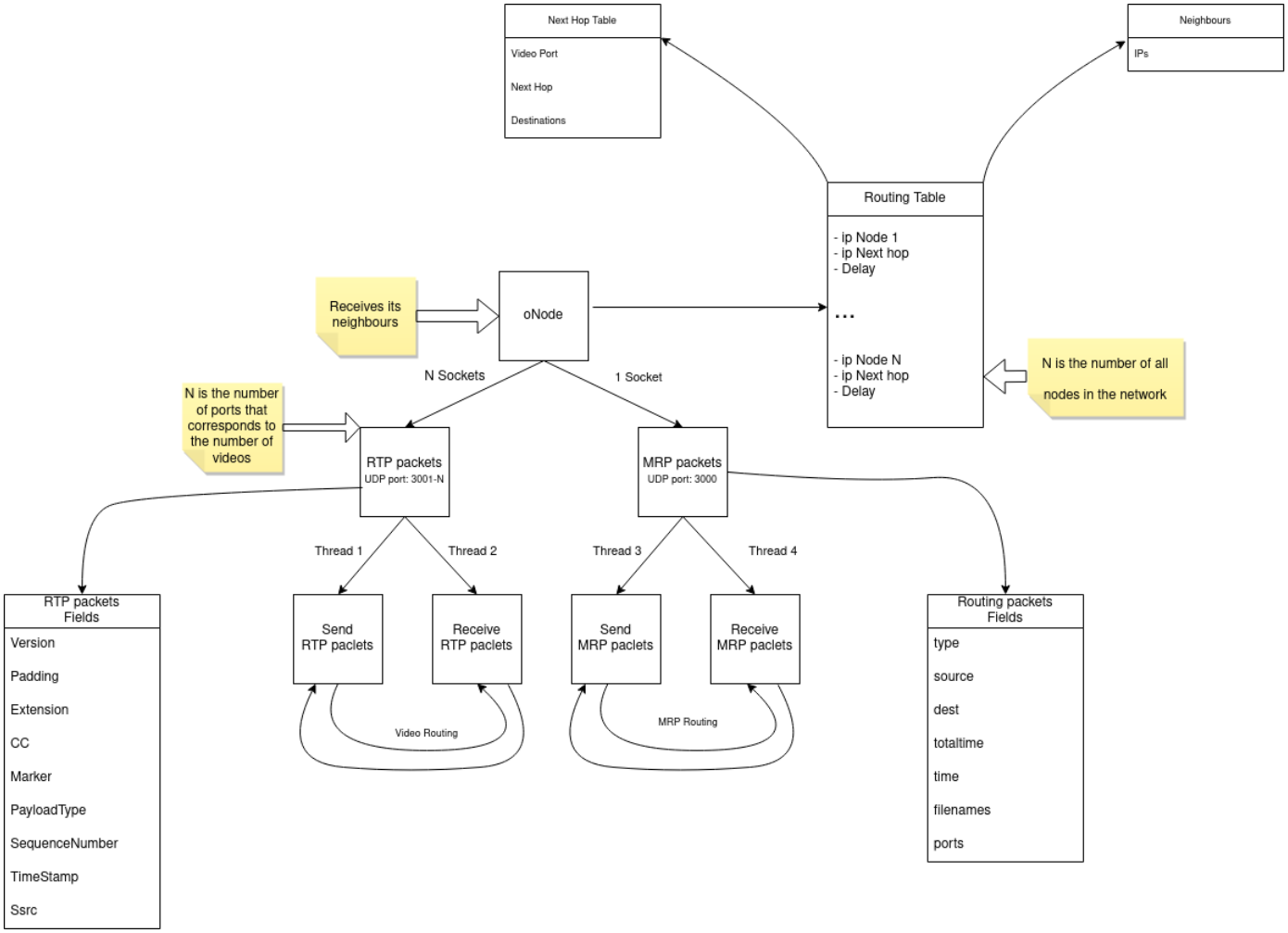

Figure 1: Arquitetura da solução

2 Especificação do(s) protocolo(s)

Tendo em conta os requisitos definidos no enunciado do projeto, foi necessário criar um protocolo aplicacional de controlo e um protocolo de streaming (fornecido, em parte, pela equipa docente).
Para tal surgem as classes MRPpacket e RTPpacket que implementam mensagens protocolares de controlo e streaming tal como iremos ver em seguida.

2.1 Formato das mensagens protocolares

MRPpacket

Esta classe foi desenvolvida para que fosse possível controlar o fluxo das mensagens de controlo e de dados que atravessam a topologia.
O formato das mensagens protocolares inclui os seguintes campos:

Método	Origem	Destino	Conteúdo
SUUP	RP	Server	type, time
YUUP	Server	RP	type, time, filenames
GETV	RP	Server	type, time, filenames, port
STRT	Client, ONode	vizinhos	type, source, totaltime, time
STOP	Client, ONode	RP	type, source, dest
STTV	Client	RP	type, source, dest, totaltime, time, filenames
STPV	Client	RP	type, source, dest, totaltime, time, filenames
ACKK	RP	Client	type, source, dest, totaltime, time, filenames, ports
GETI	ONode	Bootstrapper	type

- **type**: Identifica o tipo da mensagem, que pode ser um dos seguintes:
  - **SUUP**: Pacote do que vai do RP para o Server para saber se está ativo e se sim, que conteúdo tem.
  - **YUUP**: Pacote que vai do Server para o RP e que contém o conteúdo disponível do servidor.
  - **GETV**: Pacote que vai do RP para o Server com o vídeo pedido e a porta em que deve ser transmitido.
  - **STRT**: Enviado do Client para os vizinhos e é propagado por "flood" até ao RP. Serve para calcular os melhores caminhos e dar entrada ao cliente na rede.
  - **STOP**: Enviado do Client para o RP, de modo a retirar o nó da rede.
  - **STTV**: Pacote enviado do Client para o RP através da rota criada previamente(pelos ACKs), de modo a pedir um vídeo.
  - **STPV**: Enviado do Client para o RP. Serve para parar a transmissão de vídeo.
  - **ACKK**: O pacote é projetado para estabelecer um caminho reverso na rede, efetuando a rota inversa que foi inicialmente delineada pelo processo de "flood" executado pelo pacote STRT.
- **source**: Representa o endereço IP da origem da mensagem.
- **dest**: Representa o endereço IP do destino da mensagem.
- **totalTime**: Representa o tempo total de receção ou transmissão de dados.
- **time**: Representa a data e hora em que a mensagem foi enviada ou recebida.
- **filenames**: Corresponde a uma lista que contém os nomes dos ficheiros.

RTPpacket

Esta classe foi fornecida, pela equipa docente, para a transmissão de conteúdos de streaming pela topologia. Esta funcionará como um Real-time Transport Protocol, algo bastante usado atualmente em aplicações de tempo real como, por exemplo, entrega de dados áudio ponto-a-ponto, como Voz sobre IP.
O formato das mensagens protocolares geradas por esta classe segue o formato definido pelo cabeçalho do protocolo RTP, composto por:

- Version: Identifica a versão do protoloco RTP utilizado.
- Padding: Indica se os dados adicionais foram adicionados ao pacote.
- Extension: Indica se o pacote contém uma extensão específica
- CC: Traduz o número de indicadores CSCR presentes no cabeçalho.

- Marker: Campo reservado para aplicação específica.
- Payload Type: Indica o tipo de dados transportados no pacote como por exemplo áudio, vídeo etc.
- Sequence Number: Identifica a ordem dos pacotes enviados, permitindo que a reconstrução no destino seja ordenada.
- TimeStamp: Utilizado para sincronização temporal entre os pacotes RTP.
- SSRC: Identificador de origem de sincronização.

2.2 Interações

A interação entre o Cliente e o Servidor é facilitada por trocas de pacotes via UDP, para transmissão de vídeos em tempo real.

2.2.1 Inicialização

Os servidores são ligados com o seu conteúdo e ficam à espera de comunicação vinda do RP. De seguida, o RP é ligado e troca mensagens com o servidor, envia o SUUP e recebe o YUUP com o conteúdo dos servidores que estiverem ligados à rede. Por sua vez, são ligados os ONodes e estes ficam à espera que o cliente se ligue. Por fim, ligam-se os clientes, através de um envio de pacote STRT.

2.2.2 Pedido de vídeo

Um dado cliente, depois de se ter ligado à rede, recebe os vídeos disponíveis e envia(até ao RP) pacotes STTV, através da rota criada pelos ACKKs. Na propagação do pacote até ao RP, se algum dos nodos estiver a transmitir o vídeo, começa a transmitir também para o cliente. Chegando ao RP, se este não estiver a transmitir o vídeo, pede-o ao servidor com um GETV.

2.2.3 Recebimento e Exibição

O servidor, depois de lhe ser requerido um vídeo, faz a transmissão de pacotes RTP, de modo "unicast", para o RP. Por sua vez, o RP envia em "multicast" os pacotes do vídeo até ao cliente que o requisitou. Como esta transmissão é "multicast" é possível evitar duplicação dos pacotes em partes iguais dos caminhos para os clientes. O cliente por sua vez exibe no "GUI" o "payload" dos pacotes RTP recebidos.

2.2.4 Paragem de transmissão

Caso um cliente queira parar a transmissão de um vídeo, clica no botão "STOP" e é mandado o pacote STPV. O mesmo é propagado até ao RP e caso o este deixe de precisar de transmitir o vídeo (nenhum cliente está a transmitir), envia um pacote STPV ao servidor.

3 Implementação

3.1 Etapa 2: Construção da Topologia Overlay com Árvore Partilhada

A rede overlay foi construída com uma abordagem modular usando várias classes interconectadas para formar e manter essa rede.

Começamos com a classe ONode, esta contém a RoutingTable do nó, possui também um socket para os MRP packets e uma lista de sockets para os RTP packets, sendo um socket para cada vídeo. Esta estrutura do ONode é partilhada pelo Client, RP e Server.

O RP representa o nó central responsável pela coordenação da rede. Ele envia mensagens de inicialização (SUUP) para os servidores de vídeo e outros nós, estabelecendo conexões iniciais e servindo como ponto de referência para a rede. O RP guarda as informações sobre os servidores na classe Source. Esta é composta por um ip(endereço do servidor), um conjunto de métricas, no nosso caso apenas a latência, uma lista de conteúdos (nomes dos vídeos) e uma flag para indicar o estado do servidor.

Para representar os servidores da rede foi criada a classe do Server, cujo propósito é trocar pacotes com o RP, tanto MRP como RTP. Esta guarda ainda a lista de vídeos contidos em si.

Para gerir informações de roteamento, criamos a classe RoutingTable, que armazena detalhes sobre os próximos saltos para cada nó, para cada vídeo e para cada salto na rede, em que tem a lista de clientes a quais está a transmitir vídeo. Esta possui também uma lista dos nós vizinhos. Cada entrada individual nesta tabela é representada pela classe RoutingTableLine, que contém o próximo salto para o destino, bem como a latência do mesmo.

Para facilitar a troca de mensagens entre os nós, desenvolvemos as classes MRPsend e MRPreceive, envolvidas na implementação do protocolo MRP(multicast routing protocol). Esse protocolo é responsável por transmitir informações de controlo entre os nós, como dados de roteamento e atualizações da rede.

Além disso, implementamos classes específicas para lidar com o envio e recebimento de pacotes de vídeo usando o protocolo RTP. As classes VideoForwarder, VideoStream, RTPsend e RTPreceive trabalham em conjunto para assegurar a transmissão adequada dos pacotes de vídeo pela rede overlay.

Por fim, a existência da classe Client serve como meio de exibir o conteúdo requisitado por um cliente a um servidor. Esta contém métodos para criação de uma interface GUI, usando a biblioteca "swing".

Essas classes foram desenvolvidas de forma modular e interdependente, permitindo que cada uma desempenhe seu papel específico na criação, manutenção e operação eficiente da rede de overlay.

3.2 Etapa 3: Serviço de Streaming

3.2.1 Preparação do Vídeo

Para a leitura do vídeo criamos a classe VideoStream que é responsável por abrir e ler um arquivo de vídeo. Esta extrai os "frames" do vídeo, determina os seus tamanhos e disponibiliza os para serem enviados pela rede. Cada "frame" do vídeo é transformado num array de bytes para ser transmitido.

3.2.2 Envio do Vídeo

Na classe VideoForwarder, a cada período de tempo definido, um novo "frame" do vídeo é lido utilizando a VideoStream. Esse "frame" é encapsulado num pacote RTP, que contém informações como o tipo de conteúdo (no caso, vídeo), o número do "frame", o tempo do "frame" e os dados reais do "frame" (array de bytes). O pacote RTP gerado é então enviado pela rede. Isso é feito na classe RTPsend. Aqui, um DatagramPacket é construído com os dados do pacote RTP e enviado para a rota com destino ao Client.

3.2.3 Recebimento do Vídeo

Do lado do recetor, a classe RTPreceive está constantemente à espera de pacotes RTP na rede. Quando um pacote chega, é recebido pelo DatagramSocket. Esse pacote contém os dados do "frame" do vídeo em formato RTP. O pacote é redirecionado para o próximo salto na rede caso tenha sido recebido por um nó intermédio ou desempacotado para extrair os dados do "frame" de vídeo, caso tenha sido recebido por um cliente. No caso do cliente, o "frame" de vídeo é exibido na interface da aplicação.

3.3 Etapa 4: Monitorização dos Servidores

Para efetuar o cálculo da métrica implementamos o método duration presente na classe MRPReceive que consiste em calcular a diferença de tempo em milissegundos entre dois nós na rede de forma a obter a latência. É uma métrica bastante simplista e da qual falamos nas limitações da solução, frisando formas de a mesma ser melhorada.

3.4 Etapa 5: Construção dos Fluxos para Entrega de Dados

Quando um cliente solicita um determinado conteúdo na rede, o processo inicia-se com a submissão dessa solicitação ao nó mais próximo disponível. Esse nó mais próximo atua como um ponto inicial na procura do conteúdo desejado. Se este nó em questão já possuir o fluxo desse conteúdo, ele prontamente entrega-o ao cliente, permitindo o acesso imediato. O pedido, é na mesma propagado até ao RP, de maneira a todos os nós na rota saibam desse pedido.

No entanto, quando esse nó inicial não detém o conteúdo requisitado, entra em ação um processo de propagação dessa solicitação. A solicitação é encaminhada para outros nós conectados ao longo do caminho na direção do RP, até um nó que já se encontre a transmitir o conteúdo desejado ou até que o RP seja alcançado.

Essa abordagem garante eficiência na procura do conteúdo, permitindo que a solicitação percorra os nós de forma a encontrar o nó mais próximo que possua o conteúdo requisitado ou, em última instância, alcance o RP. O RP é o ponto de origem ou distribuição central do conteúdo na rede.

Dessa forma, o processo de procura e distribuição do conteúdo orienta-se por uma rota eficaz e direta. O objetivo é garantir que o cliente obtenha o conteúdo solicitado, traçando um caminho otimizado ao longo da estrutura da rede para sua transmissão.

3.5 Etapa 6: Definição de um método de recuperação de falhas e entradas de nós adicionais

Esta secção não foi abordada, uma vez que, na implementação que fizemos, novos clientes podem ser adicionados, o mesmo não acontece para os ONodes, pois as rotas que criamos, até ao RP, são estáticas.

4 Limitações da Solução

- Colocação de locks (sincronização);
- Escolha do servidor pela latência média e packet loss (métricas de monitorização mais robustas);
- Implementação do funcionamento do tipo STOP, de modo a tornar a remoção de nós dinâmica;
- Implementação de uma melhor biblioteca de streaming e adição de outros vídeos para uma melhor visualização dos conteúdos transmitidos;
- Implementação de um temporizador no socket que transmite pacotes RTP para fecho do mesmo caso não esteja a receber pacotes RTP;
- Reconstrução da árvore em intervalos de tempo específicos, de forma a tornar dinâmica a criação da árvore e pemitir a adição de nodos;

Por fim, a principal limitação que nos afetou de forma evidente foi quando um dos ONodes deixou de enviar pacotes corretamente para todos os clientes e não conseguimos detetar se o motivo advém da falta de leituras e escritas atómicas (locks), ou se temos algum erro no algoritmo de propagação de pacotes, nomeadamente STPVs.

5 Testes e resultados

Nesta secção iremos apresentar alguns testes realizados na topologia do core de modo a testar todas as funcionalidades implementadas, analisando se o resultado obtido foi de acordo com o resultado pretendido.

Quanto à topologia esta contém 3 servidores, S1, S2, S3(separado dos outros 2), 4 nós intermédios, RP(nó central), N1, N2, N3 e por fim 3 clientes, 2 deles C1 e C3 conectados ao mesmo nó N1 e o C2 conectado ao N2. Foi também introduzido um delay de 100 ns entre o RP e o N3.

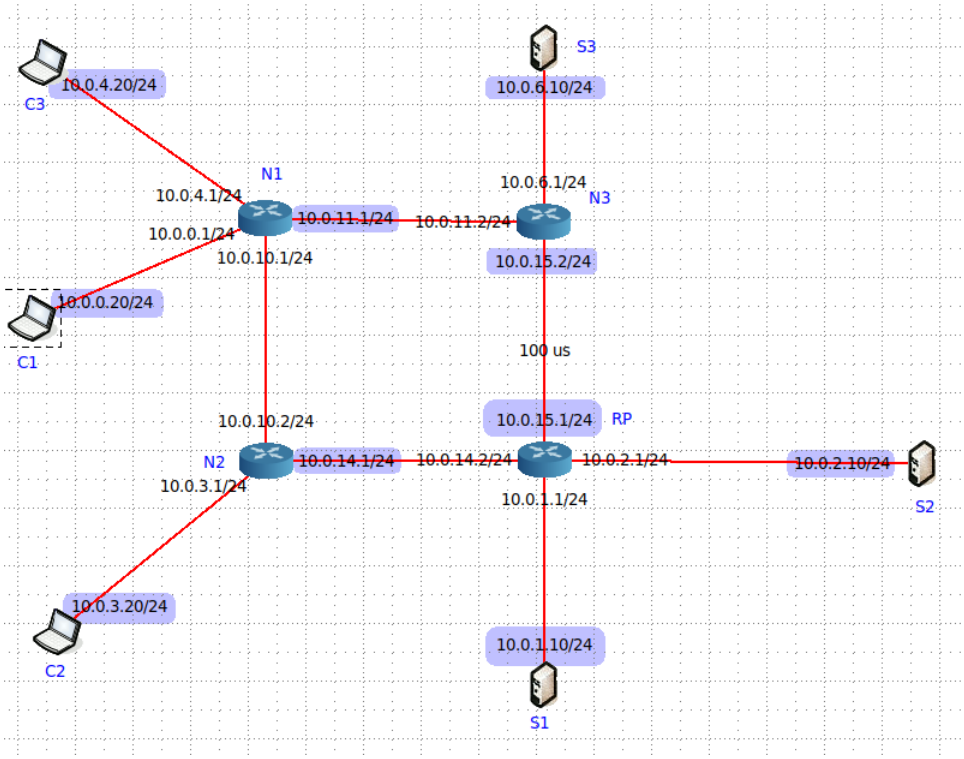

Figure 2: Topologia detalhada

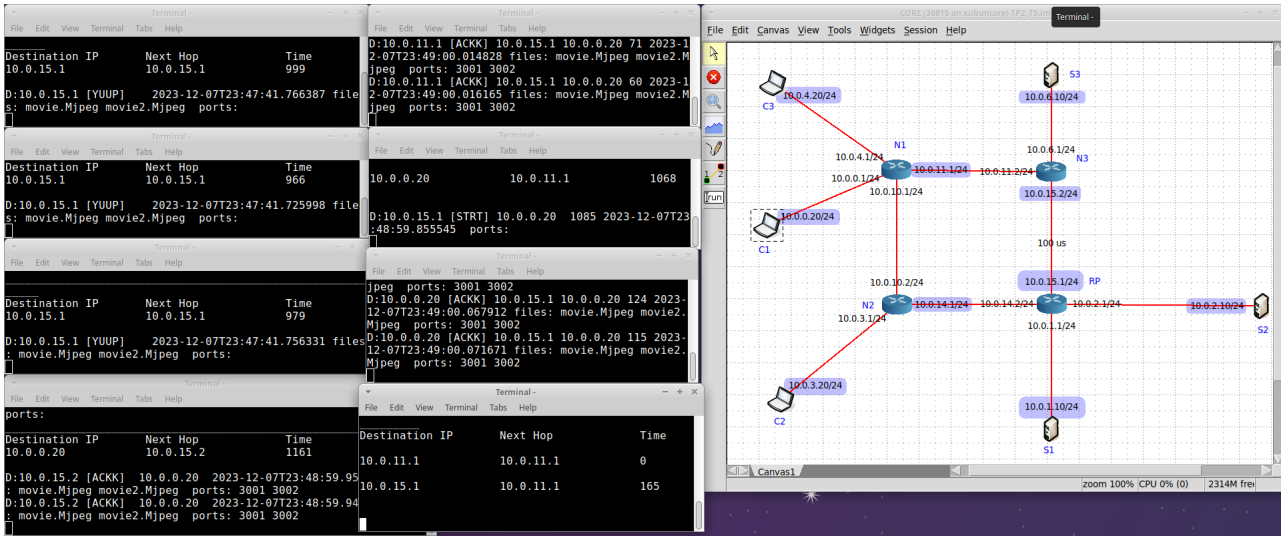

Figure 3: Nós ligados na rede e topologia usada

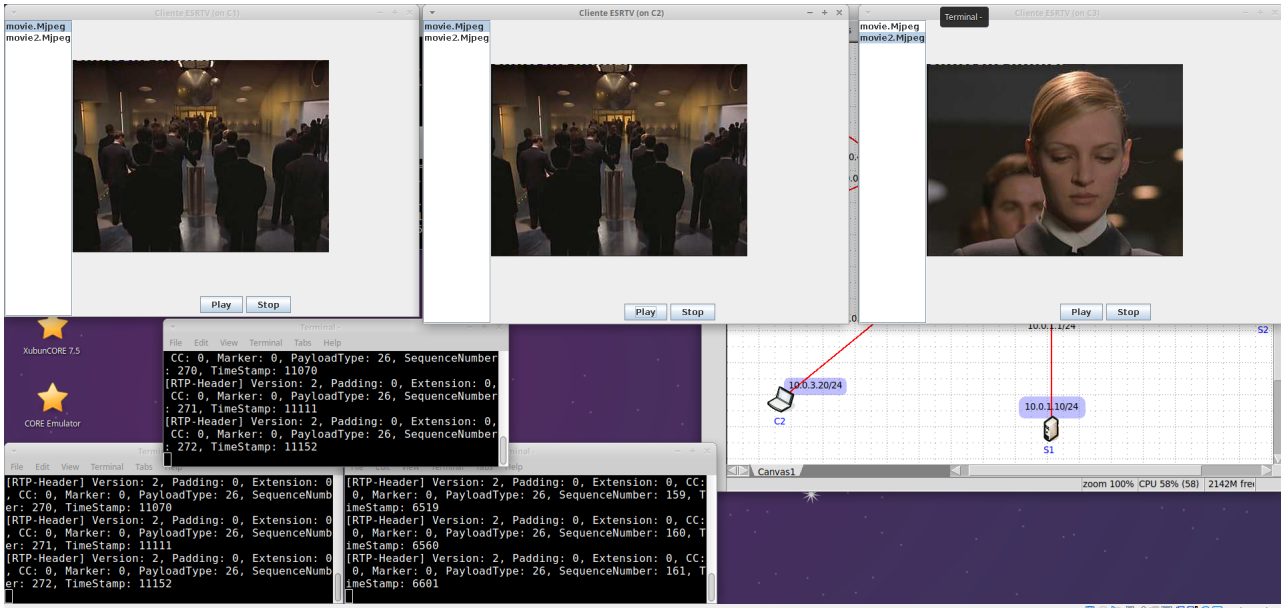

Figure 4: Exibição do conteúdo dos servidores nos clientes