Engenharia de Serviços em Rede

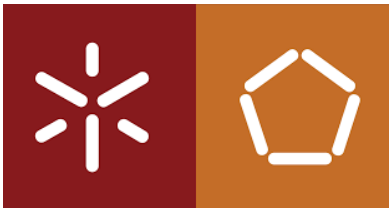
Relatório do Trabalho Prático Nº2 MEI - 2023/2024

Grupo PL 43

Hugo Martins
A95125

João Escudeiro
A96075

Ivo Ribeiro
PG53886



Universidade do Minho

Índice

Lista de Figuras

Lista de Tabelas

Resumo

O projeto proposto tem como objetivo explorar a transformação paradigmática ocorrida ao longo do último meio século na Internet, focando na mudança da comunicação de *end-system* to *end-system* para o consumo contínuo e voraz de uma variedade de conteúdos em tempo real. Esta mudança representa desafios significativos para a infraestrutura IP, originalmente não projetada para tal demanda, sendo superada por redes sofisticadas de entrega de conteúdo (CDNs) e serviços *Over the Top (OTT)*.

O foco principal do projeto é conceber um protótipo de um serviço multimédia OTT eficiente, otimizando recursos para proporcionar uma experiência ao utilizador de alta qualidade. Utilizando o CORE como plataforma de teste e uma ou mais topologias específicas, o objetivo é criar um protótipo de entrega de áudio/vídeo/texto em tempo real de um servidor central para um conjunto de clientes. O sistema proposto inclui a seleção de um *Rendezvous Point (RP)* responsável por receber o conteúdo e distribuí-lo eficientemente por meio de uma árvore de distribuição compartilhada, otimizando a criação e manutenção da mesma para minimizar a latência e a largura de banda necessária. A organização do *Overlay* aplicacional é destacada como crucial para a qualidade de serviço oferecida.

Neste documento é apresentada de forma concisa um resumo sobre todo o processo seguido durante a implementação do trabalho prático número dois.

Processo por etapas

1 Etapa 0 - Preparação das atividades

Inicialmente optamos por decidir qual seria a linguagem utilizada para a implementação. Decidimos por escolher **python**, por ser uma linguagem simples, flexível e versátil, pois pode ser considerada uma linguagem *Object-Oriented*, bem como *Imperative*.

Em termos de comunicação entre estruturas (RP, Cliente, Servidor e Nodo) decidimos utilizar o **protocolo TCP** para todas as comunicações que servissem para garantir a integridade das funcionalidades do sistema, assim como, o envio da lista de transmissões do *Servidor* ao *RP*, seleção da transmissão pelo *Cliente*, assim como a notificação de que não pretende continuar a assistir á mesma, e ainda todas as mensagens que servem para a construção e manutenção da *rede overlay* entre os nodos. Quanto ao envio dos pacotes com a informação sobre os frames de vídeo, todas as comunicações desde o *Servidor* ao *Cliente* são estabelecidas recorrendo ao **protocolo UDP** de maneira a tornar o nosso sistema o mais idêntico à realidade possível, pois num serviço de streaming não é garantida a entrega da totalidade dos pacotes.

Decidimos definir topologias que permitissem uma evolução fracionada, começando com uma topologia pequena apenas com um *Rendez-Vous Point*, um *Servidor* e dois *Clientes* (figura 1), passando para uma segunda topologia monde acrescentamos nodos entre os *Clientes* e o *RP* visando uma evolução ao nível de construção de rede pelo *RP* (figura 2), por fim expandiremos ainda mais a topologia para uma como a indicada no enunciado (figura 3) ficando com uma rede bem mais completa e diversificada.

Após conversas com os docentes e pesquisa sobre como funcionam estes tipos de redes num sistema real optamos por colocar grande parte do processamento no *RP*, assim como a construção e manutenção da *rede Overlay*, monitorização dos caminhos por onde a transmissão vai ser enviada ate aos *Clientes* finais e garantias da métrica de tempo na entrega dos pacotes, deixando o processamento ao nível dos *Nodos* muito mais simplificado sendo apenas necessário ver o conteúdo do pacote para verificar para onde tem o tem de enviar.

2 Etapa 1 - Comunicações Server-RP-Client

Numa primeira fase, decidimos estabelecer e garantir as conexões básicas entre os *Clientes* e os *Servidores* ao *RP*, tendo como base a topologia da *figura 1*, e iniciar uma troca de mensagens no formato de "string"dos *Servidores* aos *Clientes*. Todas as comunicações desta fase foram estabelecidas segundo o **protocolo TCP**.

Nesta fase garantimos que os *Servidores* quando se conectavam ao *RP*, enviavam uma mensagem com a lista de transmissões que eram capazes de transmitir. Essa informação estaria contida no seu ficheiro de configuração próprio, e seria devidamente guardada no *RP* para quando um *Cliente* se conectar ao mesmo seja devidamente informado acerca de todas as transmissões possíveis na rede. Garantimos ainda que o *Cliente* consiga selecionar uma das transmissões, apos ser informado das possíveis, para assistir e que dessa ação é desencadeado um pedido ao *RP* de encaminhamento dessa transmissão.

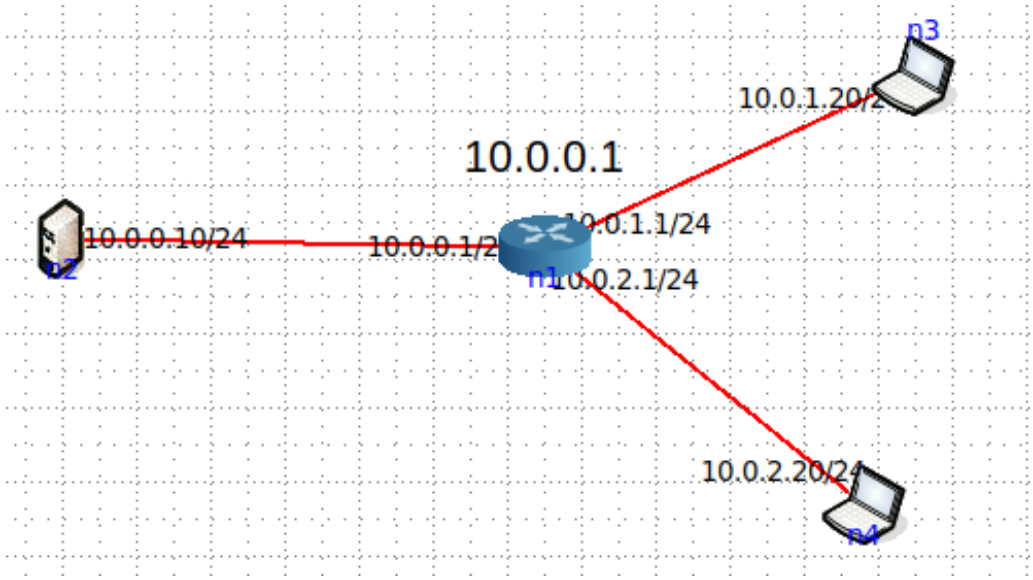


Figura 1: Topologia Inicial.

3 Etapa 2 - Árvore Partilhada Simples

Numa segunda fase optamos por adicionar à nossa topologia *Nodos* entre os *Clientes* e o *RP* (figura 2), para que conseguíssemos perceber como poderíamos adaptar o funcionamento do nosso sistema, que ate aqui apenas comunicava em **unicast** com cada *Cliente*, a um sistema que enviaria as mensagens do *RP* aos diversos *Clientes* no mesmo *Nodo* em serviço **multicast**.

Nesta fase, como já ficaram definidas todas as estruturas do sistema optamos por alterar a nossa estratégia de pré-configuração passando de um ficheiro individual para cada *Nodo*, para um ficheiro global que contém toda a informação do sistema reduzindo a vulnerabilidade de erros e até mesmo a escalabilidade da solução.

Para tal, e como previamente tínhamos definido que grande parte do processamento seria feito no *RP* incluindo a monitorização dos pacotes, decidimos, nesta fase, iniciar o processo de controlo do envio das mensagens provenientes dos *Servidores*. Este controlo ocorria de maneira simples onde o *RP* sabia à priori os caminhos até aos *Clientes* (informação presente no ficheiro de configuração global) e então quando ocorria um pedido de acesso a uma transmissão por parte de um *Cliente*, seria adicionado o caminho do *RP* ate ao mesmo a um campo no pacote que é criado com a informação sobre os frames presente no pacote recebido do *Servidor*, pacote esse que é enviado de seguida para o seu nodo vizinho.

Os *Nodos* do tipo *Node* acrescentados á topologia nesta fase terão um funcionamento bastante simples, visando à escalabilidade do nosso sistema uma vez que, simplificando o seu processamento poderemos implementar o nosso sistema em larga escala e sem necessidade de elevados recursos. Estas estruturas apenas irão ficar à espera de pacotes provenientes dos seus nodos vizinhos e apenas necessitam de a cada pacote recebido, verificar o caminho colocado no *RP*, presente no mesmo, e, mediante o caminho contido no pacote e a informação que cada nodo contém, reenviar o mesmo pacote para o nodo vizinho descritos nesse caminho.

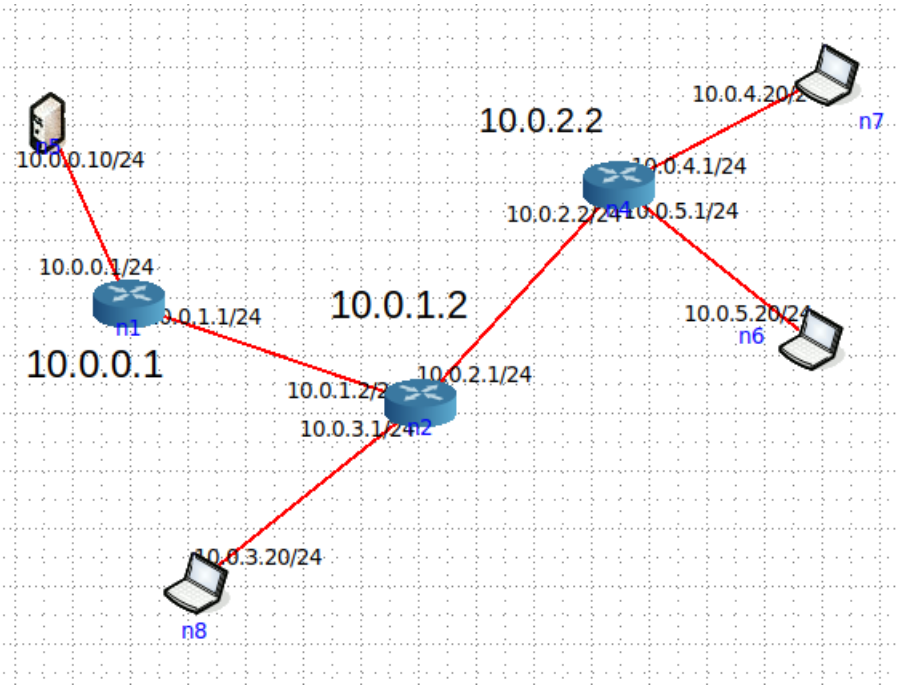


Figura 2: Topologia com Cliente Nodos, Servidor e RP.

4 Etapa 3: Serviço de Streaming

Numa terceira fase, iniciamos a implementação do serviço de troca de frames para que seja possível obter informação sobre um vídeo no *Cliente* final. Para tal implementamos um serviço no *Servidor* que funciona da seguinte forma: quando há um pedido de acesso à transmissão por parte de um *Cliente*, o *Servidor* abre então o vídeo em formato "mp4" e a cada frame constrói dois pacotes para enviar a informação desse frame em bytes. Esta estratégia tem como base a construção de um pacote sempre com o mesmo tamanho para facilitar a receção e o envio entre as estruturas. Visto que o pacote irá conter informação sobre o percurso do mesmo desde o *RP* até ao *Cliente* final, decidimos fracionar a informação sobre cada frame em bytes por dois pacotes. Todo o processo de construção dos pacotes, bem como o conteúdo dos mesmos e a sua alteração (inserção dos caminhos) é explicado no capítulo dos Pacotes ao longo da rede.

Neste serviço de envio de frames por pacotes (nesta fase ainda seguindo o protocolo TCP) decidimos definir um controlo de 25 frames por segundo de débito por parte dos *Servidores* para que a receção ao nível do *Cliente* fosse a mais idêntica à realidade possível e as transições de frames no cliente o mais suaves possível.

Assim como na fase anterior, o *RP* é o responsável por reconstruir o pacote assim que o recebe de um *Servidor* combinando a informação sobre o percurso do pacote até ao *Cliente* final e a informação sobre os frames da transmissão.

5 Etapa 4: Monitorização dos Servidores de conteúdos

Após conseguirmos receber o conteúdo em vídeo fracionado nos *Clientes*, decidimos implementar um controlo de conteúdo.

Numa primeira fase garantimos que se um *Cliente* já tivesse solicitado acesso a uma transmissão e um *Servidor* já estivesse a transmitir esse conteúdo, caso um segundo *Cliente* solicitasse acesso a essa stream, o *Servidor* não receberá uma notificação para começar a transmitir pois o *RP* ira controlar esse fluxo e garantir que o segundo *Cliente* acede ao mesmo conteúdo do primeiro. Esta métrica é garantida visto que no caminho contido no pacote (indicado pelo RP) irá a informação que o *Nodo* ligado aos dois que terá de encaminhar o pacote para ambos os *Clientes*.

Numa segunda fase garantimos que se nenhum *Cliente* estiver a assistir a uma transmissão o *RP* deve notificar o *Servidor* para parar essa transmissão.

Nesta etapa modificamos o nosso sistema passando a enviar e a trocar os pacotes com a informação acerca dos frames de vídeo segundo o **protocolo UDP**, uma vez que assim aproximamos o nosso sistema a um sistema real de transmissão de vídeo em direto. Contudo, este protocolo não garante que todos os pacotes enviados pelos servidores são recebidos pelos clientes. Para tal, implementamos do lado do cliente um sistema de controlo de frames que é abordado mais à frente, onde só é atualizada a imagem na interface gráfica se forem recebidos os dois pacotes referentes à primeira e segunda parte da totalidade de um frame em bytes. Caso ocorra a perda de uma das partes, esse frame não será mostrado na interface gráfica do *Cliente*, no entanto ,não afetará a funcionalidade da transmissão para os outros frames.

6 Etapa 5: Construção dos Fluxos finais

Com um serviço já bastante complexo, decidimos expandir a nossa topologia para a versão final da mesma (figura 3) onde apresentamos uma topologia muito próxima à realidade, com uma *rede Underlay* estruturada e a *rede Overlay* assinalada com coloração diferente para facilitar a compreensão da mesma (amarelo os nós Overlay e a vermelho o RP).

Nesta fase, passamos a informar o *RP* sobre caminhos possíveis na rede, que até aqui estavam presentes no ficheiro de configuração, através de uma construção de rede dinâmica, onde no arranque do serviço é efetuado um flush pelo *RP* que depois é propagado de nó em nó notificando-os com um pedido de estabelecimento da *rede Overlay*, que tem como consequência um envio de uma mensagem também em flush com o seu próprio *IP* para que possa ser recebida no *RP* como um caminho possível do *Nodo* ate ao mesmo, uma vez que os nodos vizinhos escrevem também o seu IP nessa mensagem.

Esta construção de rede funciona da seguinte maneira, ao arrancar o *RP* é apresentada uma interface gráfica que quando selecionada inicia o primeiro flush do *RP* para os nodos vizinhos. Assim podemos iniciar todos os nodos da nossa topologia pois só depois de selecionar a opção da interface gráfica é que a rede é construída.

O primeiro flush é enviado pelo *RP* aos seus vizinhos com uma mensagem de "Update Network"que é interpretado por eles como uma notificação de construção de rede e então mal a recebem guardam a informação que já receberam esse pedido e reenviam essa mensagem ao seus vizinhos para que eles sejam também saibam que a rede esta a ser criada. Por fim enviam uma mensagem aos seus vizinhos com o seu *IP* com o objetivo de essa mensagem chegar ao *RP* com os caminhos ate ao mesmo. Este caminho é modificado a cada nodo que recebe esta mensagem uma vez que os vizinhos apos a receção de uma mensagem com um caminho verificam se o seu *IP* já se encontra na mensagem, se se encontrar a mensagem não é reenviada aos seus vizinhos, se não estiver presente o mesmo adiciona o seu *IP* á mensagem recebida e envia essa mensagem aos seus vizinhos.

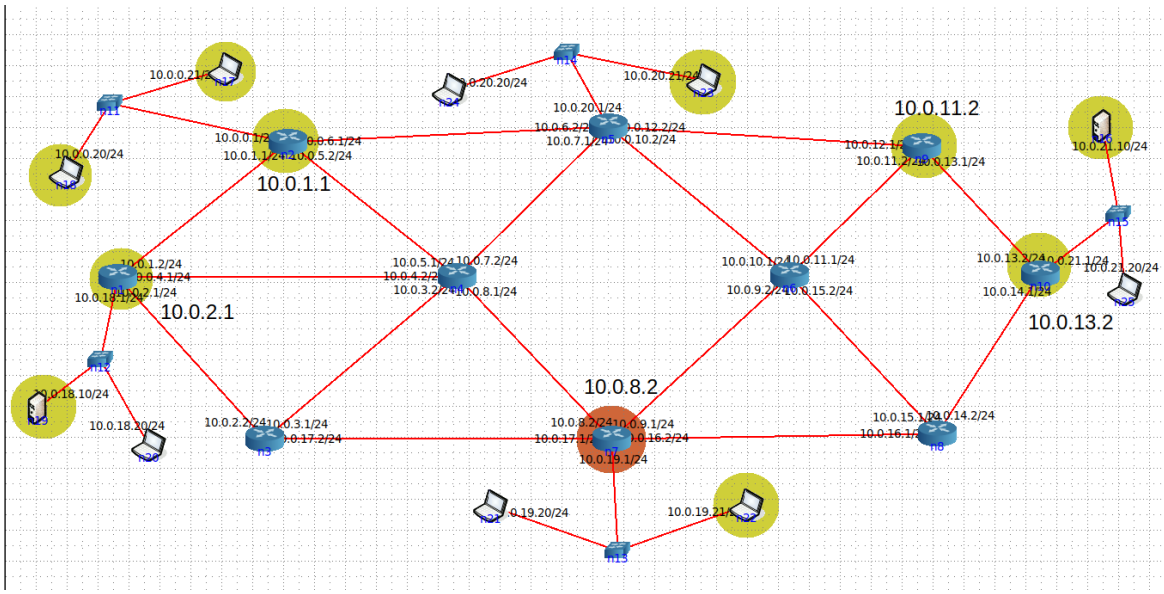


Figura 3: Topologia final.

7 Etapa 6 – Definição de um método de recuperação de falhas e entradas de nós adicionais

Por fim implementos um serviço de recuperação de falhas, assim como uma métrica de gestão de caminhos seguindo uma ótica temporal, onde o caninho de entrega de pacotes até ao cliente é escolhido segundo o menor intervalo de tempo de entrega, que é atualizado também ele dinamicamente.

De maneira a prevenir a falha aquando ocorrem quebras na *rede Overlay*, reestruturamos o nosso código de maneira a que após o *flush* inicial mencionado na etapa anterior, um novo *flush* seja efetuado apos um intervalo de tempo constante. Nesta reestruturação tivemos de modificar a mensagem proveniente do *RP* com uma numeração indicando a que *flush* corresponde permitindo a todas as estruturas de rede perceberem se devem ou não iniciar a troca de mensagens com caminhos ou não.

Esta estratégia permite que a *rede Overlay* seja atualizada dinamicamente e esta atualização permitirá recuperar a rede quando ocorre uma falha num dos nodos. Para além disso se se tratar de um *Cliente*, este insere na mensagem que é criada com o seu *IP*, o *timestamp* atual para que quando recebido no *RP* este possa calcular o intervalo de tempo do envio de mensagens por esse caminho.

Resultante desta última implementação é possível aceder à informação constante e dinamicamente atualizada no *RP* quanto ao caminho mais rápido para chegar a cada cliente. Sempre que esta informação é alterada, isto é, ocorre uma atualização de rede que informa o *RP* de um caminho mais rápido para enviar pacotes a um determinado *Cliente*, o *streamControler* responsável por enviar a transmissão a esse cliente modificará o caminho que insere na mensagem de forma a respeitar a métrica de envio dos pacotes segundo o menor intervalo de tempo possível.

Fluxogramas das estruturas

Para além de uma explicação textual decidimos incluir vários fluxogramas que demonstram o comportamento de cada uma das entidades presentes na nossa estrutura. Desta forma é apresentada a informação de um modo visualmente mais apelativo, bem como é apresentado de forma sequencial e temporal as ações de cada componente.

8 Starter

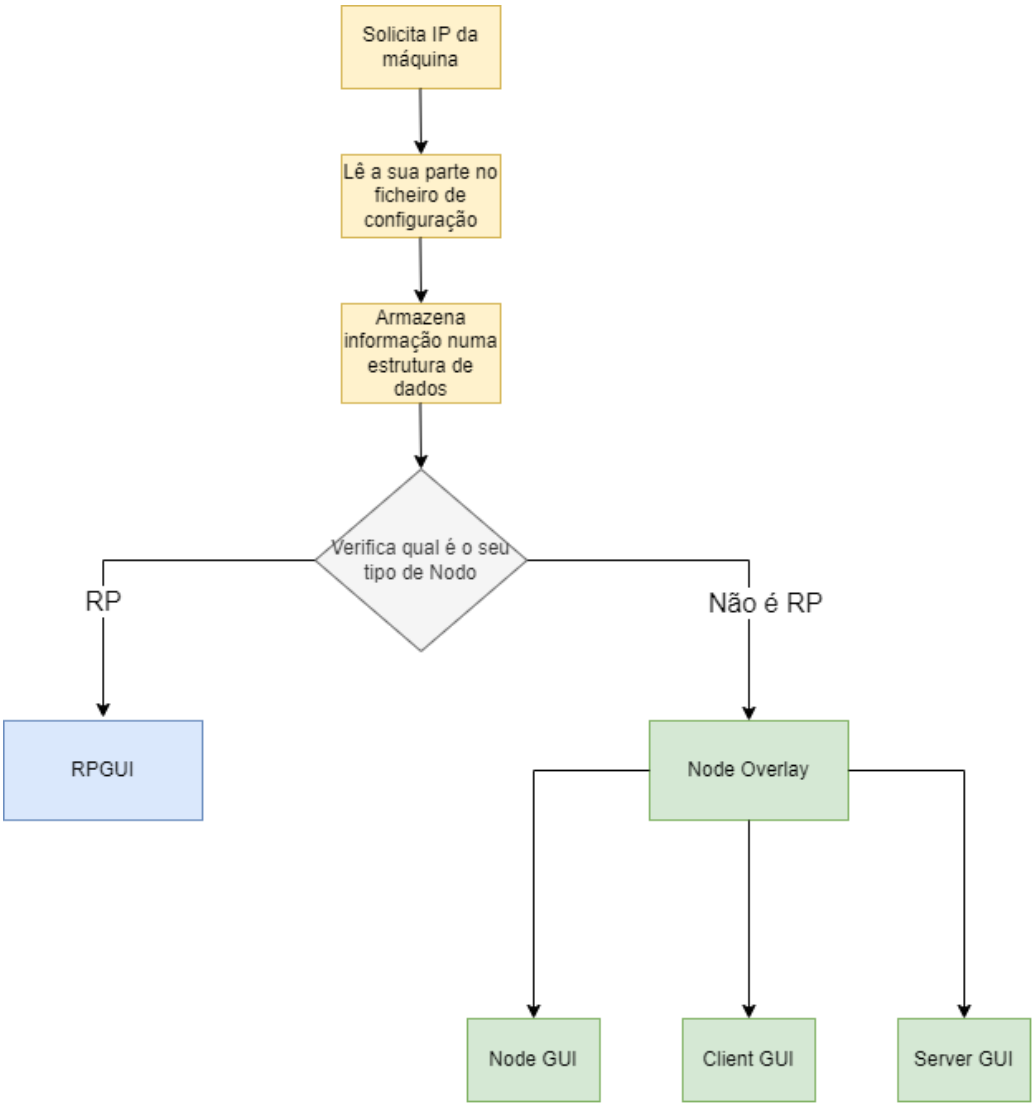


Figura 4: Fluxograma do Starter.

Todas as estruturas que não são o Rendezvous Point são consideradas NodeOverlay uma vez que esta estrutura é usada apenas para garantir a manutenção da Rede Overlay, permitindo ao RP saber sempre quais nodos estão conectados e quais os caminhos possíveis na rede assim como a latência dos mesmos.

9 NodeOverlayGUI

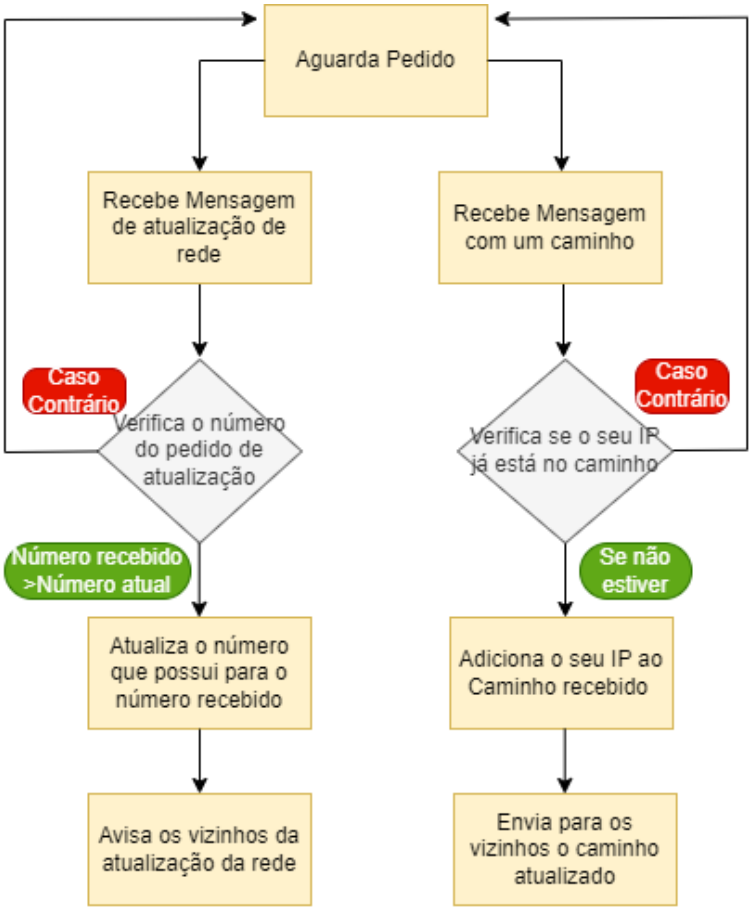


Figura 5: Fluxograma de um Nodo Overlay.

10 RPGUI

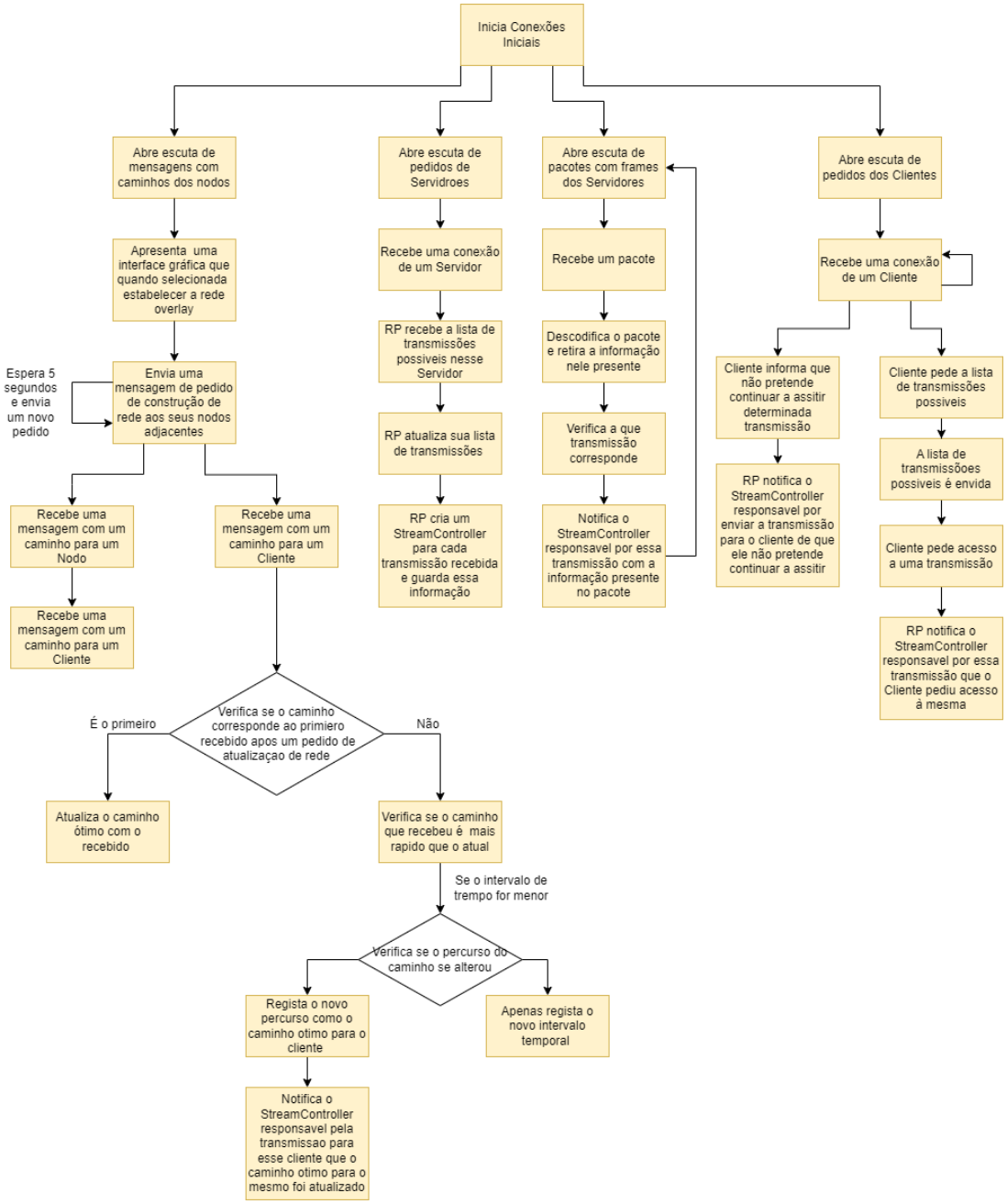


Figura 6: Fluxograma do RP.

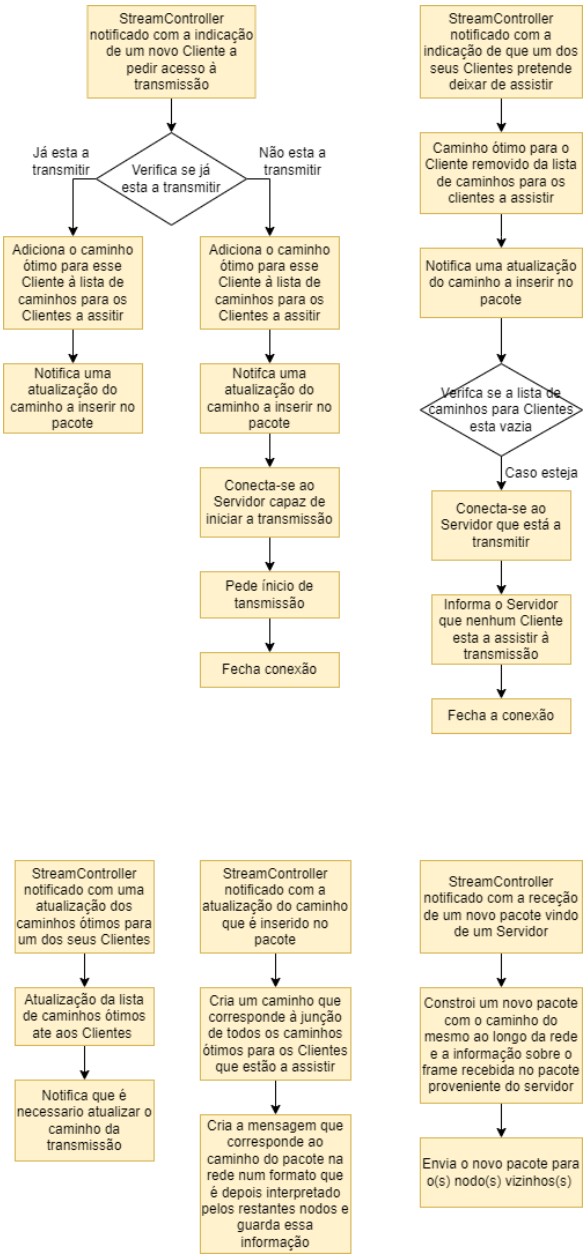


Figura 7: Fluxograma de um Controlador de Stream presente no RP.

11 NodeGUI

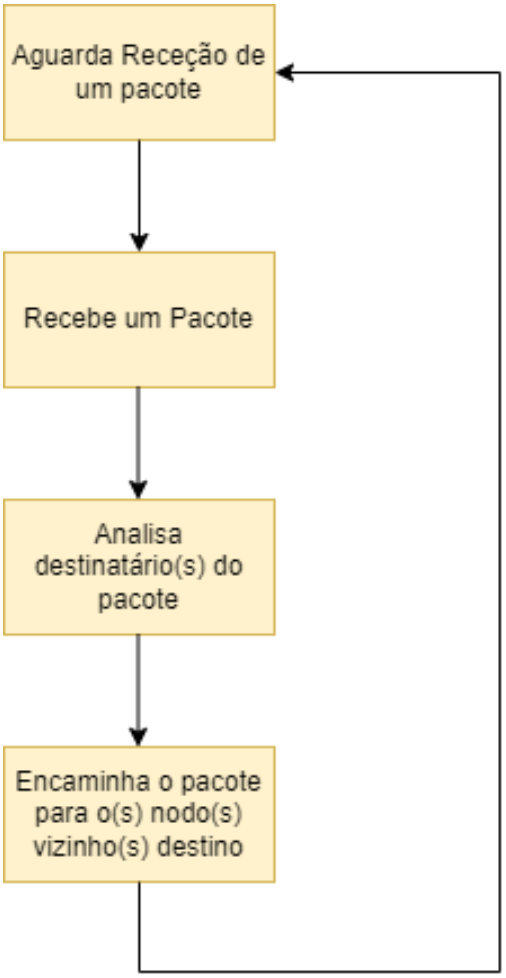


Figura 8: Fluxograma de um Nodo.

12 ClientGUI

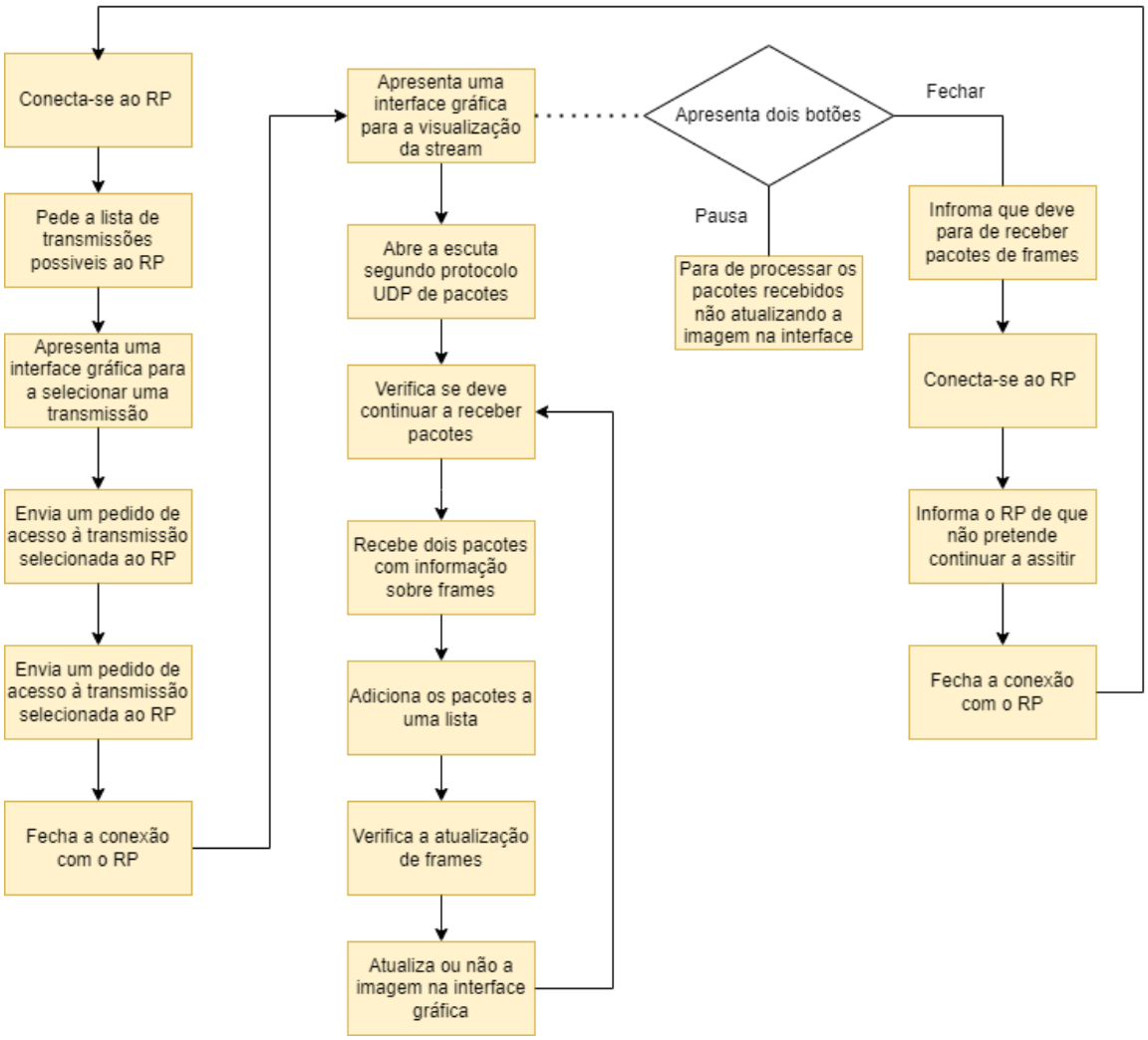


Figura 9: Fluxograma de um Cliente.

13 ServerGUI

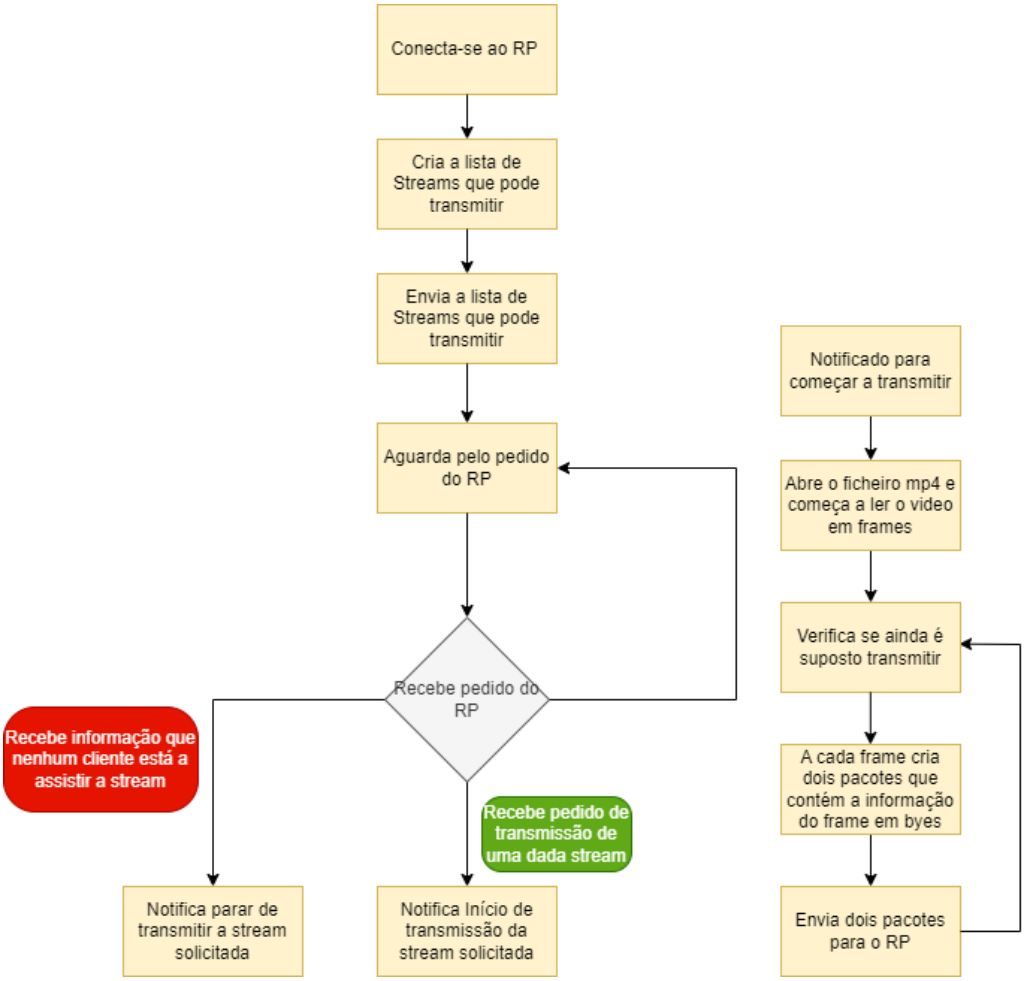


Figura 10: Fluxograma de um Servidor.

Verificação de frames

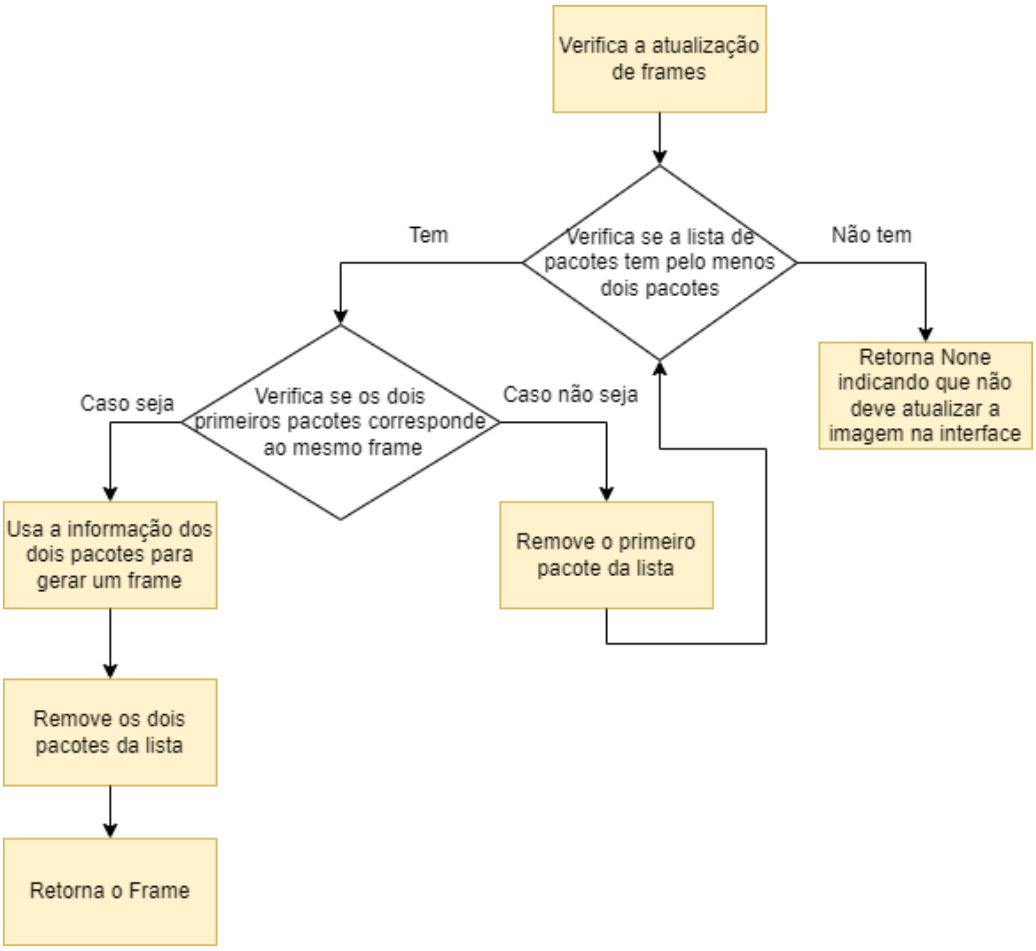


Figura 11: Fluxograma da verificação de Frames.

Estrutura dos pacotes UDP ao longo da rede

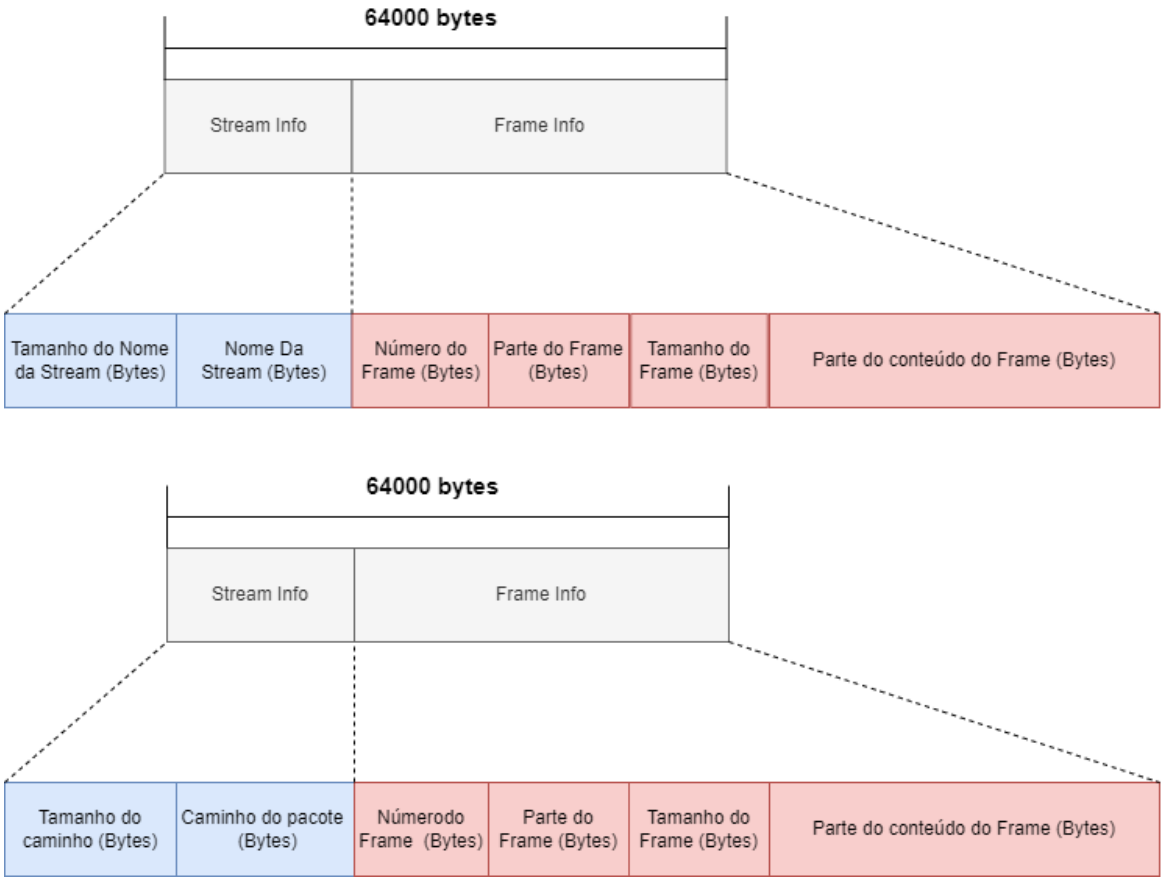


Figura 12: Conteúdo dos Pacotes UDP (em cima servidor para RP, em baixo RP para Nodo).

Como anteriormente abordado o *RP* é que é responsável por monitorizar os percursos dos pacotes desde a sua receção. Aquando da sua receção ele identifica a que transmissão corresponde e informa o *Controlador de Stream* responsável por essa transmissão que recebeu um novo pacote do *Servidor* e que deve fazer com que o mesmo chegue ao *Cliente final*. Para que seja garantida a compreensão, por parte dos nodos, da informação relativa ao caminho que o pacote tem de seguir, o *RP* refaz o pacote adicionando a informação proveniente do servidor ao percurso do caminho.

Comunicações ao longo da rede

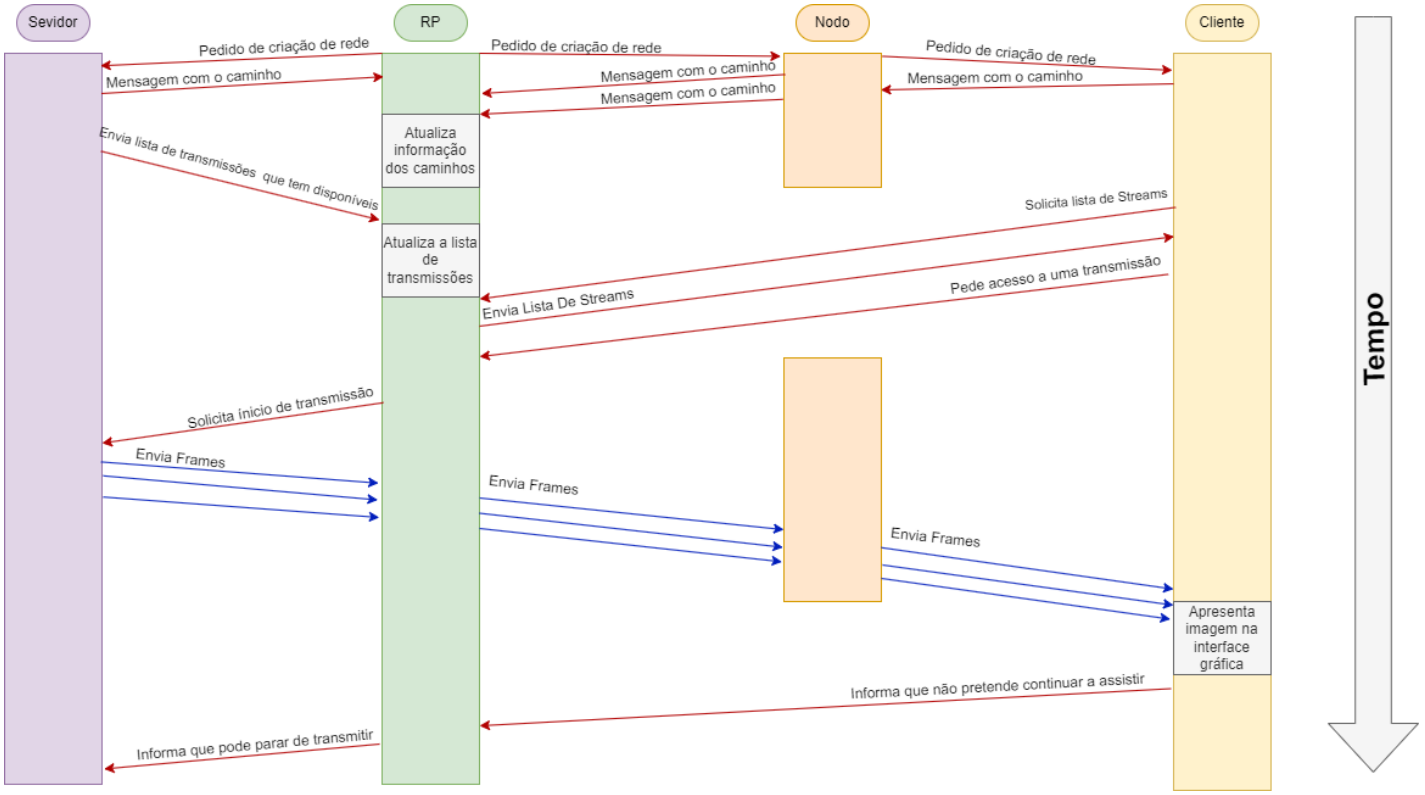


Figura 13: Fluxograma das comunicações na rede.

No exemplo acima descrito apresentamos um exemplo de teste onde temos uma topologia simples de um *Servidor* ligado ao *RP* e um *Cliente* ligado ao um *Nodo* que por sua vez esta ligado ao *RP*. Com este exemplo pretendemos mostrar em quais das interações do nosso sistema usamos cada protocolo de comunicação.

Cenários de Teste

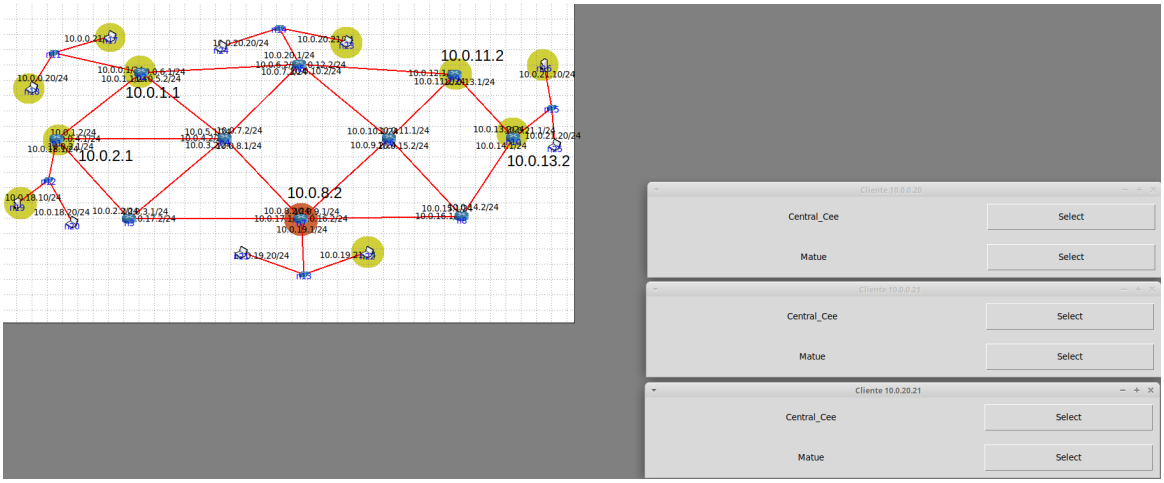


Figura 14: Cenário de Teste com Menus em 3 clientes.

No cenário de teste da Figura 14, apresentamos uma das facetas do nosso sistema que é o facto de suportar multicliente, onde ligamos apenas o *Servidor*, identificado com o IP 10.0.18.10 que é capaz de transmitir duas streams identificadas como "Central_Cee"e "Matue", e os três *Clientes* identificadas pelo seu IP na respetiva interface gráfica, dispersos na rede.

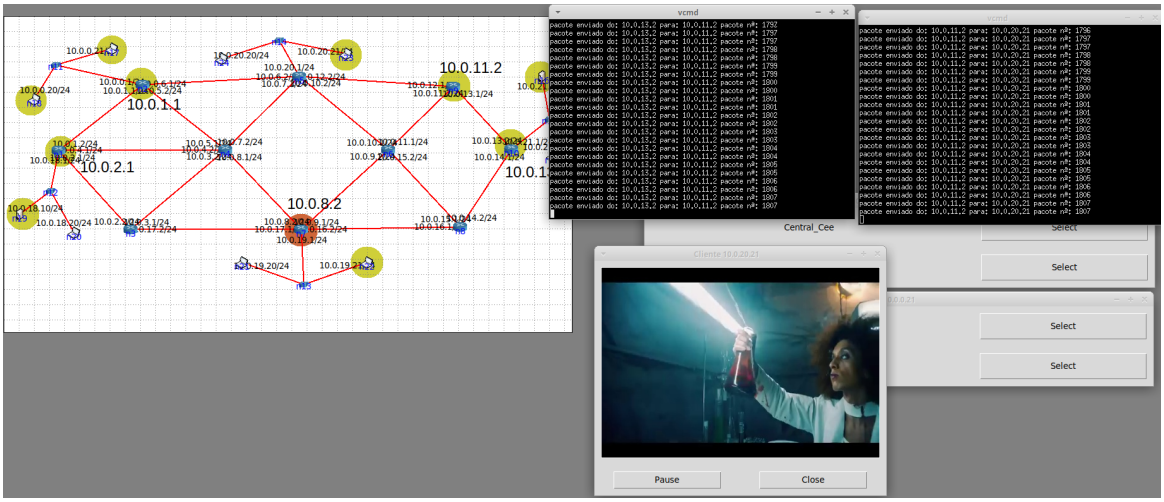


Figura 15: Cenário de Teste com um cliente a consumir a stream.

No cenário de teste da Figura 15, exemplificamos o que um *Cliente* observa quando seleciona uma transmissão . Através dos terminais correspondentes aos *Nodos* da rede Overlay por onde passam os pacotes de vídeo provenientes do *RP*, conseguimos também observar o percurso dos mesmos até ao *Cliente final*.

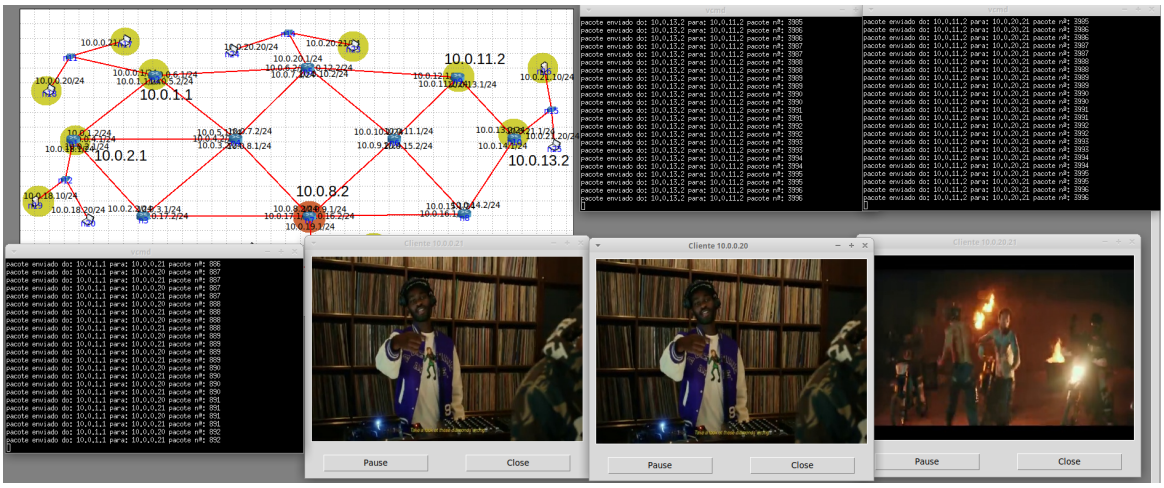


Figura 16: Cenário de Teste com três clientes a consumir a stream.

No cenário de teste da Figura 16, é mostrado o comportamento do nosso sistema para o caso onde três *Clientes* solicitam acesso a transmissões e as respetivas interfaces gráficas, assim como os terminais dos nodos por onde os pacotes passam ficando percetível o caminho dos mesmos. Neste exemplo é ainda possível observar o comportamento do nosso código para o caso em que dois *Clientes* ligados ao mesmo nodo solicitam acesso à mesma transmissão em que o nodo adjacente aos mesmos recebe um pacote e envia a mesma informação para os dois *Clientes* ao mesmo tempo, implementando assim um serviço multicast.

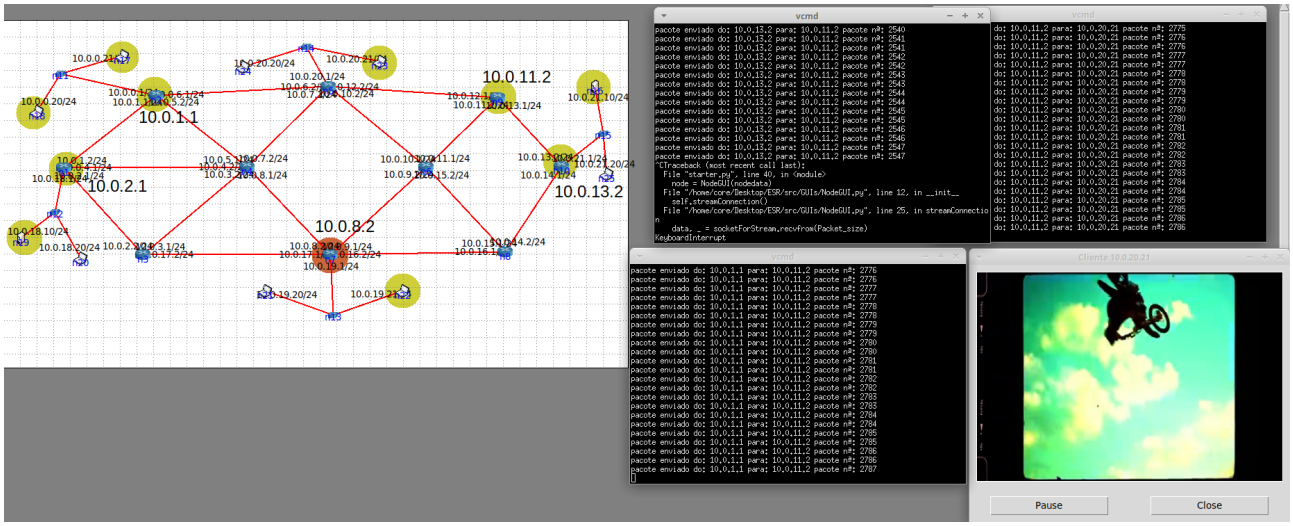


Figura 17: Cenário de Teste com quebra num nodo que estava contido na rota.

No cenário de teste acima apresentado mostramos como o nosso sistema suporta falhas na rede, onde o *Nó* com o IP 10.0.13.2 é quebrado, e observamos que ,após um novo flush de construção de rede, a mesma é restaurada e os pacotes voltam a ser reencaminhados para o Cliente por um novo percurso.

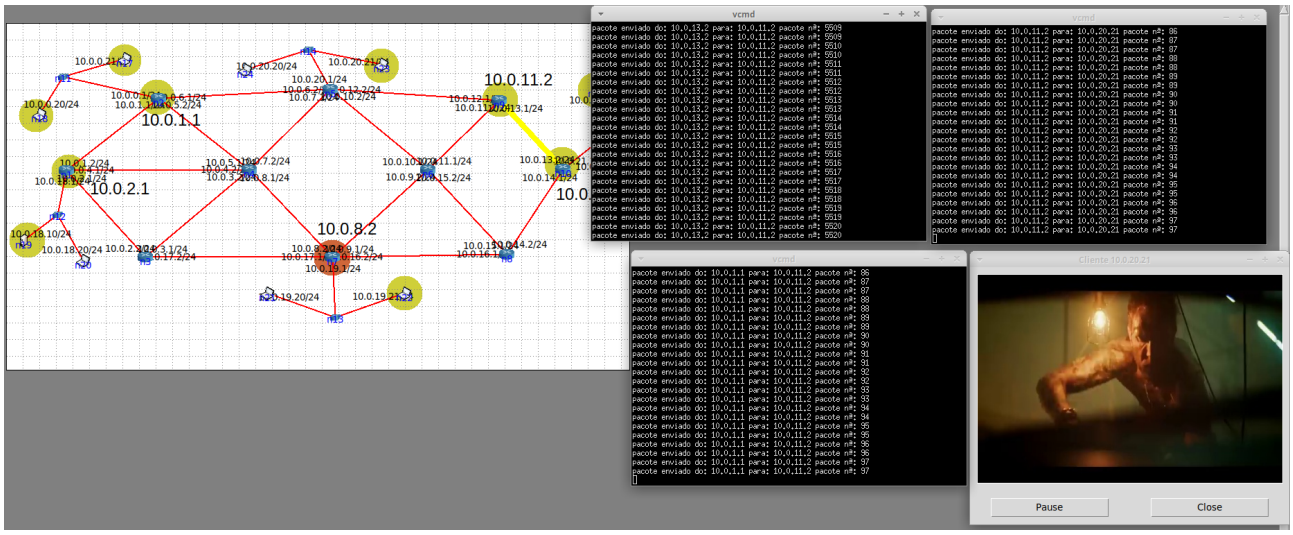


Figura 18: Cenário de Teste com latencia adicionada numa ligação.

No cenário de teste acima apresentado mostramos como o sistema se comporta quando adicionamos delay numa ligação entre dois nodos. Neste caso o colocamos um delay de 5 ms na ligação assinalado com coloração amarela e verificamos que apesar de ter o nodo com o IP 10.0.13.2 aberto o sistema prefere o caminho alternativo uma vez que é menos demorado, garantindo assim a métrica de escolha do caminho com menor latência.