

Engenharia de Serviços em Rede - TP2

Júlio Gonçalves^[PG50537], Henrique Alvelos^[PG50414], and Duarte Cerquido^[PG50469]

Departamento de Informática, Universidade do Minho, Braga, Portugal

Abstract. Este documento sumariza o desenvolvimento de um projeto no âmbito da unidade curricular de Engenharia de Serviços em Rede. Com este, pretendemos desenvolver um serviço para entrega de multimédia em tempo real.

Keywords: OTT, RTP Protocol, Redes de Comunicação

1 Introdução

Ao longo do tempo, é inevitável observar uma mudança de paradigma na partilha e transmissão de informação nas redes. Essa mudança é devida aos serviços Over The Top (OTT), que resolvem a entrega massiva de conteúdos e oferecem serviços específicos desenhados sobre a camada aplicacional.

Como objetivo neste trabalho prático, é nos proposto o desenvolvimento de um protocolo para a entrega de áudio, vídeo e texto em tempo real através de um serviço Over the Top(OTT), promovendo a eficiência e otimização de recursos para uma melhor qualidade da experiência por parte do utilizador.

2 Diretrizes do Projeto

Em seguida, serão expostas algumas diretrizes presentes no enunciado prático, sobre quais o desenvolvimento do projeto deve seguir:

- A partir de um servidor de conteúdos deve ser possível enviar áudio/texto/vídeo para um conjunto de N clientes;
- Um conjunto de nós pode ser usado no reenvio dos dados, como intermediários, formando entre si a rede de overlay aplicacional;
- Quando um nó da rede overlay começa a sua execução, precisa de conhecer pelo menos um outro nó qualquer da rede a quem se ligar;

Os seguintes etapas deverão ser abordados durante o desenvolvimento do projeto de modo a assegurar o correto funcionamento do serviço. É sobre a natureza destas etapas que o nosso protocolo será desenvolvido:

- **Construção da Topologia Overlay;**
- **Serviço de Streaming;**
- **Monitorização da Rede Overlay;**
- **Construção de Rotas para a Entrega de Dados;**
- **Ativação e Teste do Servidor Alternativo;**

3 Arquitetura da Solução

A linguagem Java foi a escolhida para o desenvolvimento deste serviço devido às inúmeras bibliotecas que esta possui no que toca a transmissão de dados;

O nosso protocolo será desenvolvido tendo em conta o protocolo de transporte UDP, uma vez que oferece uma melhor velocidade de transmissão.

Os intervenientes do nosso serviço podem ser distinguidos entre **Servidor** (nodo que efetua a transmissão de conteúdos), **Cliente** (nodo que faz o pedido de transmissão de conteúdos), **Nodo** (nodo integrante da rede, cuja função é encaminhar mensagens).

4 Especificação do Protocolo

Para existir uma correta comunicação entre os vários nodos, de forma a garantir a integridade dos vários serviços diretos e indiretos que o nosso projeto suporta, foi necessária a criação e diferenciação de diversos tipos de pacotes de dados.

4.1 List Neighbors Request Packet

Este pacote tratado do pedido da lista de vizinhos aquando da inicialização de um nodo.

- **OPCODE** - Identificador do Tipo de Pacote;
- **idOverlay** - Endereço do servidor bootstrapper;

List Neighbors Request Packet	
OPCODE	1
idOverlay	String

4.2 Download Request Packet

Este pacote tratado do pedido de transmissão de conteúdos.

- **OPCODE** - Identificador do Tipo de Pacote;
- **Filename** - Identificador do ficheiro a pedir ao Servidor;

Download Request Packet	
OPCODE	2
Filename	String

4.3 Data Packet

Este pacote permite fazer a transmissão de dados.

- **OPCODE** - Identificador do Tipo de Pacote;
- **BlockN** - Numerador do Pacote;
- **Length**- Tamanho do Pacote;
- **Data** - Informação a ser transmitida;

Data Packet	
OPCODE	3
BlockN	int
Length	short
Data	byte[]

4.4 Acknowledgment Packet

Este pacote permite fazer a confirmação da recepção de pacotes.

- **OPCODE** - Identificador do Tipo Pacote;
- **BlockN** - Número do Pacote confirmado;

Acknowledgment Packet	
OPCODE	4
BlockN	int

4.5 Probe Packet

Este pacote permite fazer a monitorização periodicamente das condições da rede.

- **OPCODE** - Identificador do Tipo de Pacote;
- **idServidorRemetente** - Endereço do servidor que difundiu a mensagem;
- **numHops** - Número de nodos percorridos até à receção do pacote;
- **timestampEnvio** - Tempo em que o pacote foi enviado;
- **stringRotaPercorrida** - Endereços dos nodos percorridos até à receção do pacote;
- **atrasoMs** - Diferença de tempos entre o envio e a receção do pacote;

Probe Packet	
OPCODE	5
idServidorRemetente	String
numHops	int
timestampEnvio	Timestamp
stringRotaPercorrida	String
atrasoMs	long

4.6 RTP Packet

Este pacote permite a comunicação em tempo real, seguindo o protocolo Real-time Transport Protocol.

- **Version** - indica a versão do protocolo RTP que está sendo usado.
- **Padding** - indica se o preenchimento foi adicionado ao pacote RTP.
- **Extension** - indica se o pacote RTP contém um cabeçalho de extensão.
- **CC** - indica o número de identificadores CSRC incluídos no pacote RTP.
- **Marker** - especifica variantes do protocolo.
- **Payload Type** - indica o tipo de carga transportada no pacote RTP.
- **Sequence Number** - identifica pacotes RTP individuais e permite detectar pacotes perdidos ou fora de ordem.
- **Timestamp** - permite sincronizar o fluxo de mídia e calcular o tempo decorrido entre a transmissão de dois pacotes RTP.
- **Ssrc** - permite identificar a origem de um pacote RTP.
- **Header** - refere os campos no cabeçalho RTP que contém informações sobre o pacote RTP.
- **Payload_size** - tamanho do cabeçalho
- **Payload** - dados transportados em um pacote RTP, excluindo o cabeçalho RTP.

RTP Packet	
Version	int
Padding	int
Extension	int
CC	int
Marker	int
Payload Type	int
Sequence Number	int
Timestamp	int
Ssrc	int
Header	byte[]
Payload_size	int
Payload	byte[]

4.7 StreamRequestPacket Packet

Este pacote permite informar .

- **OPCODE** - Identificador do Tipo Pacote;

StreamRequestPacket Packet	
OPCODE	6

4.8 EndStreamPacket Packet

Este pacote permite informar .

- **OPCODE** - Identificador do Tipo Pacote;

EndStreamPacket Packet	
OPCODE	9

5 Implementação

A implementação do nosso serviço de streaming foi desenvolvido de forma faseada e gradual. A mesma seguiu as seguintes etapas, ordenadas cronologicamente:

5.1 Primeiros Passos & Construção da Topologia Overlay

Nesta primeira fase do projeto, começamos por construir um programa simples que permitia a comunicação entre dois nodos diferentes.

Havendo comunicação, começamos por implementar a construção da rede overlay. A estratégia utilizada é baseada num controlador. Um nodo da topologia funciona como bootstrapper, tendo conhecimento de toda a topologia da rede. Quando um nodo se conecta a topologia, faz um pedido ao bootstrapper a pedir informações sobre os seus vizinhos de modo a formar conexões.

5.2 Serviço de Streaming

Para inicializarmos o serviço de Streaming na rede, o cliente deve efetuar um pedido ao Servidor. Uma vez que os diferentes nodos da rede já sabem qual o melhor caminho até ao servidor, estes encaminham a mensagem para o mesmo, ativando e guardando as rotas ativas para o qual os conteúdos devem ser transmitidos. Assim que o pedido chega ao servidor, o mesmo começa a enviar pacotes de streaming. Quando o cliente decide não receber mais informação da transmissão, envia um pacote ao servidor vizinho, de modo a que a rota de streaming construída para o cliente seja "destruída".

5.3 Monitorização da Rede Overlay

A nossa abordagem a este problema teve como solução apenas existirem mensagens de monitorização a partir dos servidores. Para tal, o servidor envia um pacote de monitorização para a rede a cada 10 segundos. Cada nó que recebe este pacote, atualiza na sua tabela de rotas o melhor caminho para chegar ao servidor remetente do pacote caso as métricas do mesmo forem melhores. De seguida, atualiza o pacote com os seus dados e reenvia aos seus vizinhos exceto àquele do qual recebeu o pacote (inundação controlada).

Desta forma, conseguimos nos adaptar a diferentes cenários de rede permitindo sempre que a transmissão de dados percorra as rotas mais rápidas.

5.4 Construção de Rotas para a Entrega de Dados

Para a construção de rotas para a entrega de dados, fazemos uso dos pacotes de monitorização da rede para a construção das rotas até aos servidores remetentes das mensagens. Através destas, construímos tabelas de encaminhamento com informação sobre a Origem/Servidor, Remetente, número de nodos até ao servidor, tempo de envio desde o servidor até ao nodo.

Como fazemos uso de mensagens periódicas para a construção de rotas, asseguramos que as tabelas de encaminhamento estão sempre atualizadas sobre as rotas com prestação de melhores velocidades.

5.5 Ativação e Teste do Servidor Alternativo

Quando o serviço de streaming do servidor é interrompido, um serviço de streaming do servidor alternativo é ativo, garantido a contínua transmissão de dados mesmo em cenários de falha.

Para implementarmos esta solução, fazemos uso novamente dos pacotes de monitorização (Probe Packet). O nodo cliente, assim que deteta uma melhor rota para obter a stream de conteúdo, altera o fluxo da mesma para o novo servidor e manda um novo pedido de streaming.

6 Testes e Resultados

A topologia usada para testes e verificação de resultados é demonstrada na *Figura 1*. O software de suporte à criação e simulação de transmissão de dados foi o *Core*.

Para a construção da topologia, como é possível verificar na *Figura 2*, o nodo O2 e O3 quando iniciados, efetuam um pedido da listinha de vizinhos ao servidor ativando as rotas para os seus vizinhos.

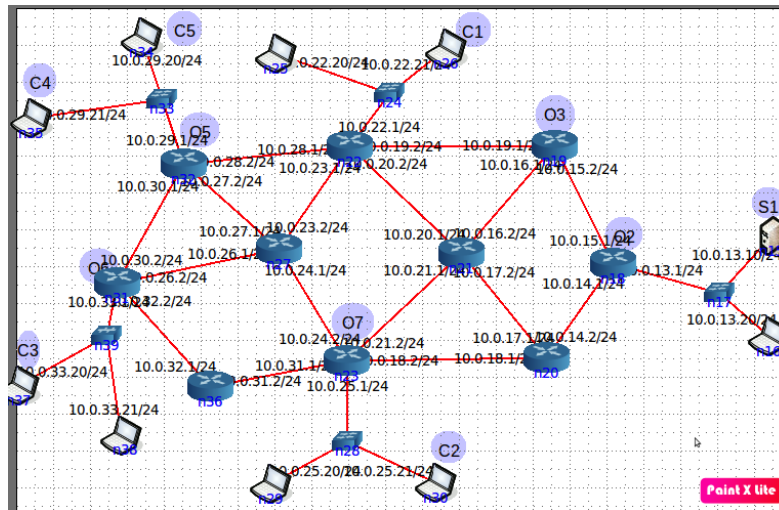


Fig. 1. Topologia da Rede

```

vcmd 03
root@n19:/tmp/pycore.40163/n19.conf# java -jar tp2esr.jar 03 10.0.13.10
args[0]=03 args[1]=10.0.13.10
Init node
Vou começar o processo de stream
[TEXT_DOWNLOADER]: Sent LNRP to (/10.0.13.10,6666)
[TEXT_DOWNLOADER]: Recieved DATA blockN = 1, len = 40
[PEER]: Vizinhos recebidos => O2=10.0.15.1;C1=10.0.22.21;O5=10.0.28.2;

vcmd 02
root@n18:/tmp/pycore.40163/n18.conf#
root@n18:/tmp/pycore.40163/n18.conf# java -jar tp2esr.jar 02 10.0.13.10
args[0]=02 args[1]=10.0.13.10
Init node
Vou começar o processo de stream
[TEXT_DOWNLOADER]: Sent LNRP to (/10.0.13.10,6666)
[TEXT_DOWNLOADER]: Recieved DATA blockN = 1, len = 40
[PEER]: Vizinhos recebidos => S1=10.0.13.10;O3=10.0.15.2;O7=10.0.18.2;

vcmd Servidor
[PEER]: Sou o bootstrapper. Vizinhos => O2=10.0.13.1;
[TEXT_SENDER]: Sent DATA to (/10.0.13.1,43392) content = S1=10.0.13.10;O3=10.0.15.2;O7=10.0.18.2;
[SERVER]: Sent response to LRP with neighbors = S1=10.0.13.10;O3=10.0.15.2;O7=10.0.18.2;
[TEXT_SENDER]: Sent DATA to (/10.0.15.2,52881) content = O2=10.0.15.1;C1=10.0.22.21;O5=10.0.28.2;
[SERVER]: Sent response to LRP with neighbors = O2=10.0.15.1;C1=10.0.22.21;O5=10.0.28.2;

```

Fig. 2. Construção da Topologia Overlay

A verificação de resultados aquando da transmissão de dados no nosso serviço pode ser confirmada como exposto na *Figura 3*.

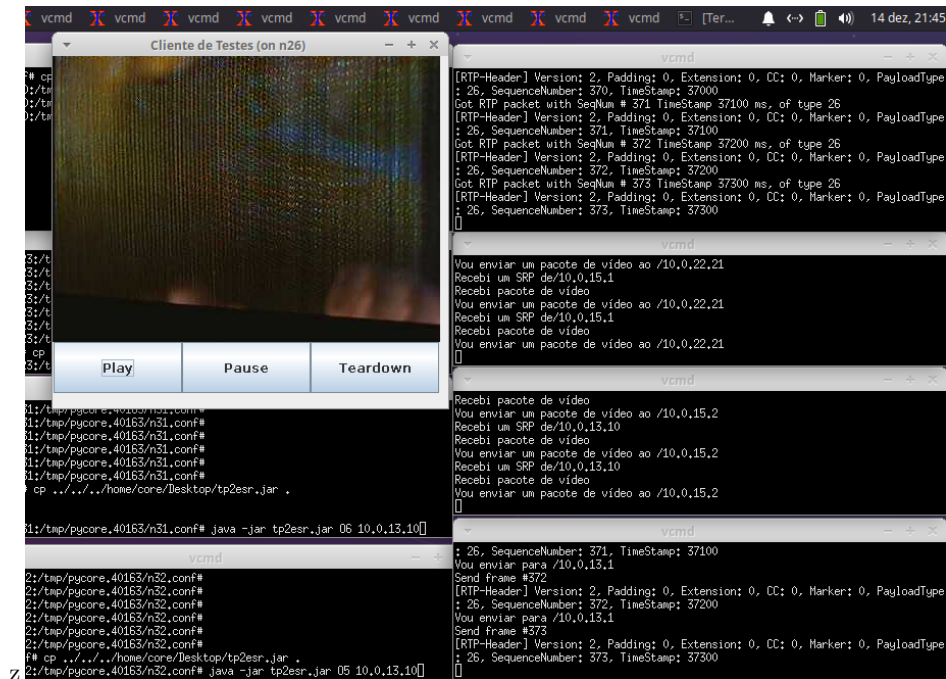


Fig. 3. Streaming no Cliente

Na figura 4 é possível verificar o envio de pacotes de monitorização, e através desses a construção de tabelas de encaminhamento em cada nodo. Nota: Neste teste, foi adicionado um Servidor Alternativo "S2", com ligação ao nodo O5. Esta nova ligação foi adicionada também ao ficheiro bootstrapper.

The figure displays six terminal windows, each showing a sequence of network protocol messages. The messages include 'Sent PROBE packet, next probe in 10s...', 'Received PROBE PACKET', and 'Tabela de Rotas:' (Route Table). The logs show the progression of route discovery and construction across different nodes (S1, S2) and hops (NumSaltos). The route tables show the origin, remitter, number of hops, timestamp, and the path taken (rotaPerCorrida).

```

vcmd
[SERVER]: Sent PROBE packet, next probe in 10s...
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S2', numHops=2, timestampEnvio=2022-1
2-14 22:17:09.543, atraso=1ms, rotaPerCorrida =S2:05:S2;)
Tabela de Rotas:
Origem: S1 | Remetente: 05 | NumSaltos: 4 | Timestamp: 18
Origem: S2 | Remetente: 05 | NumSaltos: 2 | Timestamp: 11
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S1', numHops=4, timestampEnvio=2022-1
2-14 22:17:13.349, atraso=19ms, rotaPerCorrida =S1:02:03:05:S2;)
Tabela de Rotas:
Origem: S1 | Remetente: 05 | NumSaltos: 4 | Timestamp: 18
Origem: S2 | Remetente: 05 | NumSaltos: 2 | Timestamp: 11

vcmd
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S2', numHops=1, timestampEnvio=2022-1
2-14 22:17:09.543, atraso=3ms, rotaPerCorrida =S2:05;)
Tabela de Rotas:
Origem: S1 | Remetente: 03 | NumSaltos: 3 | Timestamp: 16
Origem: S2 | Remetente: S2 | NumSaltos: 1 | Timestamp: 3
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S1', numHops=3, timestampEnvio=2022-1
2-14 22:17:13.349, atraso=16ms, rotaPerCorrida =S1:02:03:06;)
Tabela de Rotas:
Origem: S1 | Remetente: 03 | NumSaltos: 3 | Timestamp: 16
Origem: S2 | Remetente: S2 | NumSaltos: 1 | Timestamp: 3

vcmd
2-14 22:17:09.543, atraso=24ms, rotaPerCorrida =S2:05:03:C1;)
Tabela de Rotas:
Origem: S1 | Remetente: 03 | NumSaltos: 3 | Timestamp: 20
Origem: S2 | Remetente: 03 | NumSaltos: 3 | Timestamp: 24
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S1', numHops=3, timestampEnvio=2022-1
2-14 22:17:13.349, atraso=24ms, rotaPerCorrida =S1:02:03:C1;)
Tabela de Rotas:
Origem: S1 | Remetente: 03 | NumSaltos: 3 | Timestamp: 20
Origem: S2 | Remetente: 03 | NumSaltos: 3 | Timestamp: 24

vcmd
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S2', numHops=2, timestampEnvio=2022-1
2-14 22:17:09.543, atraso=16ms, rotaPerCorrida =S2:05:03;)
Tabela de Rotas:
Origem: S1 | Remetente: 02 | NumSaltos: 2 | Timestamp: 15
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S2', numHops=2, timestampEnvio=2022-1
2-14 22:17:09.543, atraso=14ms, rotaPerCorrida =S2:05:03;)
Tabela de Rotas:
Origem: S1 | Remetente: 02 | NumSaltos: 2 | Timestamp: 15
Origem: S2 | Remetente: 05 | NumSaltos: 2 | Timestamp: 14
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S1', numHops=2, timestampEnvio=2022-1
2-14 22:17:13.349, atraso=4ms, rotaPerCorrida =S1:02:03;)
Tabela de Rotas:
Origem: S1 | Remetente: 02 | NumSaltos: 2 | Timestamp: 4
Origem: S2 | Remetente: 05 | NumSaltos: 2 | Timestamp: 14

vcmd
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S2', numHops=3, timestampEnvio=2022-1
2-14 22:17:09.543, atraso=35ms, rotaPerCorrida =S2:05:03:02;)
Tabela de Rotas:
Origem: S1 | Remetente: S1 | NumSaltos: 1 | Timestamp: 10
Origem: S2 | Remetente: 03 | NumSaltos: 3 | Timestamp: 26
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S1', numHops=1, timestampEnvio=2022-1
2-14 22:17:13.349, atraso=1ms, rotaPerCorrida =S1:02;)
Tabela de Rotas:
Origem: S1 | Remetente: S1 | NumSaltos: 1 | Timestamp: 1
Origem: S2 | Remetente: 03 | NumSaltos: 3 | Timestamp: 26

vcmd
[SERVER]: Sent response to LRP with neighbors = 03:10,0,19,1;C5=10,0,29,20;C4=10
,0,29,21;C6=10,0,30,2;07=10,0,24,2;
[SERVER]: Sent PROBE packet, next probe in 10s...
[SERVER]: Sent PROBE packet, next probe in 10s...
[SERVER]: Received PROBE PACKET
ProbePacket(opcode=5, idServidorRemetente='S2', numHops=4, timestampEnvio=2022-1
2-14 22:17:09.543, atraso=33ms, rotaPerCorrida =S2:05:03:02:51;)
Tabela de Rotas:
Origem: S2 | Remetente: 02 | NumSaltos: 4 | Timestamp: 39
[SERVER]: Sent PROBE packet, next probe in 10s...

```

Fig. 4. Flooding e Construção de Rotas

7 Conclusões

Fazendo uma análise final do projeto desenvolvido tendo em conta os objetivos propostos, demonstramos bastante agrado para com o mesmo. Sentimos que foi um trabalho desafiante e complexo, que pediu muita discussão e atenção aos pormenores uma vez que as alternativas de soluções são múltiplas e incubiamos analisar os prós e contras de cada uma, o que, para além de desenvolver competências técnicas permitiu aumentar as capacidades de trabalho de grupo e de espírito crítico.

Sentimos que embora o nosso projeto cumpra todos os objetivos propostos, à medida que fomos desenvolvendo o serviço fomos nos apercebendo de melhores estratégias que poderiam ser implementadas no que toca ao suporte e funcionamento do serviço, mas que por limitações de tempo foram descartadas.