Universidade do Minho
Departamento de Informática

Mestrado Integrado em Engenharia Informática

Arquiteturas Emergentes de Redes

Entrega de Conteúdos P2P em Redes Móveis Espontâneas

a82529 Carlos Afonso
a86617 Gonçalo Nogueira
a74806 João Amorim

Braga
junho, 2021

Conteúdo

Lista de Figuras

1 Introdução

Este trabalho foi proposto pelos docentes da unidade curricular de Arquiteturas Emergentes de Rede e tem como objetivo desenhar, implementar e testar uma solução de entrega de conteúdos P2P sobre redes móveis espontâneas.

Para tal, foi utilizada a plataforma CORE e a toda a implementação foi feita usando a linguagem JAVA.

Este projeto passou por duas fases diferentes, com algumas alterações na segunda fase. Durante este relatório, toda a implementação será explicada, destacando as alterações feitas.

2 Especificação dos Protocolos

Relativamente aos protocolos, foram implementados vários, sendo estes importantes para o desenvolvimento do trabalho.

2.1 Primitivas de comunicação

De forma a atingir o bom funcionamento da rede, foram implementados protocolos de passagem de mensagens por UDP e TCP.

Durante toda a primeira fase, apenas foi utilizado o protocolo UDP, pois achamos que era a melhor forma de atingir velocidade e eficácia na rede. No entanto, durante a segunda fase, devido ao facto de não conseguirmos implementar um controlo de perda de dados, utilizamos o protocolo TCP. Apesar de esta não ser a melhor forma, é sim a melhor forma para garantir que os conteúdos são enviados de nodo para nodo, uma vez que o controlo de perdas de dados é tratado pelo protocolo TCP.

2.2 Formato de mensagens

Dois tipos de mensagens foram implementadas durante o desenvolvimento desta arquitetura.

Para passar pedidos, foi simplesmente criada uma string, que toma a seguinte forma:

<tipo de pedido>:<ficheiro/interesse>:<identificador do nodo que fez o pedido>:<nodos por onde passou a mensagem>:ttl:<identificador de pedido>

Como podemos ver acima, esta string contem toda a informação necessária para os nodos perceberem bem como processar e o que fazer.

- <tipo de pedido>: Tipo de pedido permite ao nodo saber como processar o pedido.
- <ficheiro/interesse>: Qual o interesse ou qual o ficheiro que o nodo pretende
- <identificador do nodo que fez o pedido>: O nodo que pediu o conteudo originalmente
- <nodos por onde passou a mensagem>: Nodos por onde a mensagem passou até ao momento
- <ttl>: Time-to-Live do pacote
- <identificador do pedido>: Identificador do pedido para que os nodos não recebam pedidos que já receberam.

Para passar conteudo, foi criado o objeto 'Mensagem' que encapsula todos os dados necessários sobre o conteúdo e tem a seguinte estrutura:

```
public class Mensagem implements Serializable {
    private String nomeFile;
    private Integer idNodo;
    private Integer idOriginal;
    private String descricao;
    private byte[] conteudo;
    private Set<Integer> nodosAtr;
    private String pedidoCounter;
```

Figura 1: Mensagem

Este objeto contém informação relativa aos seguintes aspetos:

- <nomeFile>: Ficheiro que está a ser enviado.
- <idNodo>: Identificador do nodo que enviou o conteúdo originalmente.
- <idOriginal>: O nodo que pediu o conteudo originalmente.
- <descrição>: ficheiro ou interesse pedido.
- <conteudo>: Ficheiro em bytes.
- <nodosAtr>: Nodos por onde o conteúdo tem de passar para voltar ao nodo original.
- <pedidoCounter>: Identificador do pedido de conteúdo.

2.3 Interações

Quanto às interações entre os nodos, podemos dividir em duas fases.

Na primeira fase temos uma arquitetura rede P2P circular, ou seja, todos os nodos têm vizinhos estipulados. Consideremos agora que um nodo faz um pedido de conteúdo. Quando este pedido é feito, o nodo envia-o para os seus nodos vizinhos, ou seja, o seu sucessor e o seu antecessor. Cada um destes vai ser responsável por enviar ao seu próximo, até que seja encontrado um nodo que tenha esse conteúdo. Quando esse nodo que tem o conteúdo recebe o pedido, a partir da mensagem de pedido, o nodo liga-se diretamente ao nodo que fez o pedido originalmente, e envia-lhe o conteúdo desejado. De destacar que nesta fase, os conteúdos eram enviados "crus", ou seja, não era encapsulado em objeto, mas sim convertido em bytes e enviado ao nodo que fez o pedido.

Na segunda fase, a arquitetura foi alterada, de forma a priveligiar uma rede que concilia P2P, NDN's e DTN's. As alterações mais relevantes são as seguintes. Deixou-se de ter a noção de vizinhos, os conteúdos são enviados por TCP e agora os nodos podem pedir interesses, em vez de um ficheiro em particular. Posto isto, podemos agora explicar com detalhe as interações nesta nova rede.

Considerando o caso inicial acima, onde um nodo faz um pedido. Nesta fase, como os nodos deixam de ter sucessor e antecessor, o pedido é enviado por flooding, ou seja, é enviado a todos os

nodos atingíveis. Estes nodos enviam também por flooding a todos os nodos que consegue atingir e caso não consiga encontrar outro nodo que não seja o que lhe enviou o pedido, vai aguardar até conseguer uma conexão com outro nodo. Quando um nodo que tem conteúdo que bate certo com o pedido de interesse que recebeu, é criado uma mensagem do tipo "Mensagem"acima descrito, que vai encapsular o conteúdo. Essa mensagem é enviada apenas aos nodos por onde o pedido passou, ou seja, caso o nodo consiga se conectar ao nodo original, envia logo para ele. Caso contrário, vai enviando pelos nodos por onde o pedido passou, eventualmente chegando ao nodo original.

3 Implementação

De seguida, iremos abordar toda a implementação feita, com alguns detalhes.

3.1 Pings e estatísticas

Na primeira fase, para ser possível garantir que o nodo nunca fica sem vizinhos, foi aberto um SocketDatagram que vai estar a fazer pings de forma ativa, para perceber se os nodos vizinhos são atingíveis. Caso algum desses vizinhos se desconectar, foi implementado um controlo de vizinhos, que vai garantir que o nodo que perdeu o vizinho rapidamente encontre outro candidato para ser vizinho. Na segunda fase, este controlo torna-se obsoleto, uma vez que agora o envio de mensagens passa por uma abordagem de flooding. Quanto às estatísticas, os nodos são capazes de perceber o quão longe estão de outros nodos, pois é sempre garantido que se sabe o número de hops pelo qual as mensagens passam, para que o nodo que pediu receba o conteudo do nodo mais perto.

3.2 Comunicação

Cada nodo tem portas abertas, isto em ambas as fases, de forma a poder receber vários pedidos ao mesmo tempo e de forma integrada, para que nunca se confundam pedidos e mensagens de conteúdos e, de tal forma, impedir o envio errado de dados.

3.3 Generalização da arquitetura

Abaixo, podemos ver a generalização da arquitetura implementada

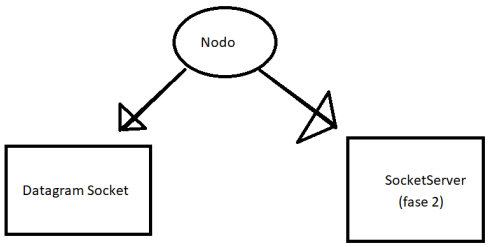

Figura 2: Arquitetura

Como podemos ver, implementamos um "ouvido"por UDP, que na primeira fase servia para passar todo o tipo de dados. Na segunda fase, o grupo decidiu criar um SocketServer para separar por completo a passagem de conteúdo pela passagem de pedidos. De destacar que, caso fosse implementado o controlo de perda de dados, este 'ouvido' TCP seria em UDP, talvez por outra porta, para continuar a separar os pedidos dos conteúdos

4 Testes e resultados

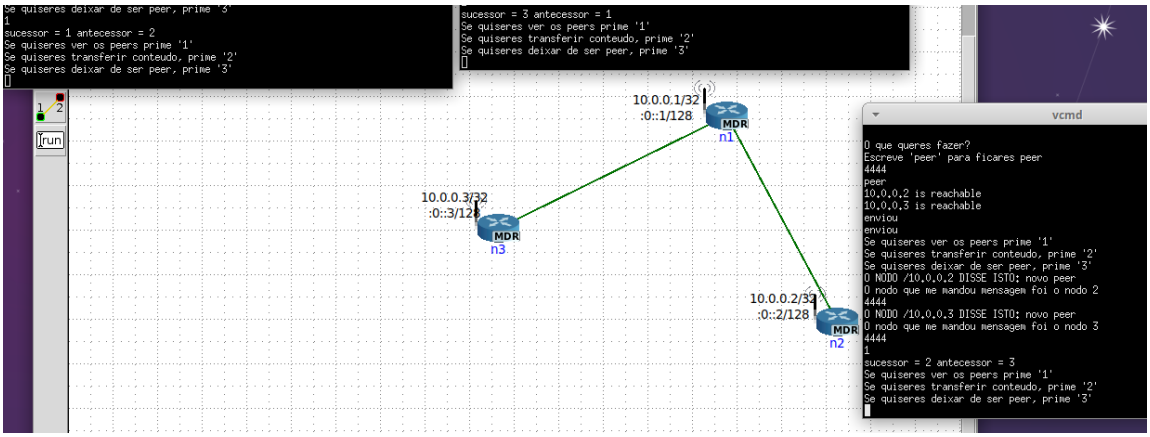

Figura 3: Vizinhança da rede P2P circular - Fase 1

Uma vez que a transferência de dados na primeira fase é um pouco simples, não achámos necessário apresentar testes de transefrência de dados, sendo que decidimos que o mais importante seria apresentar o sistema de vizinhos.

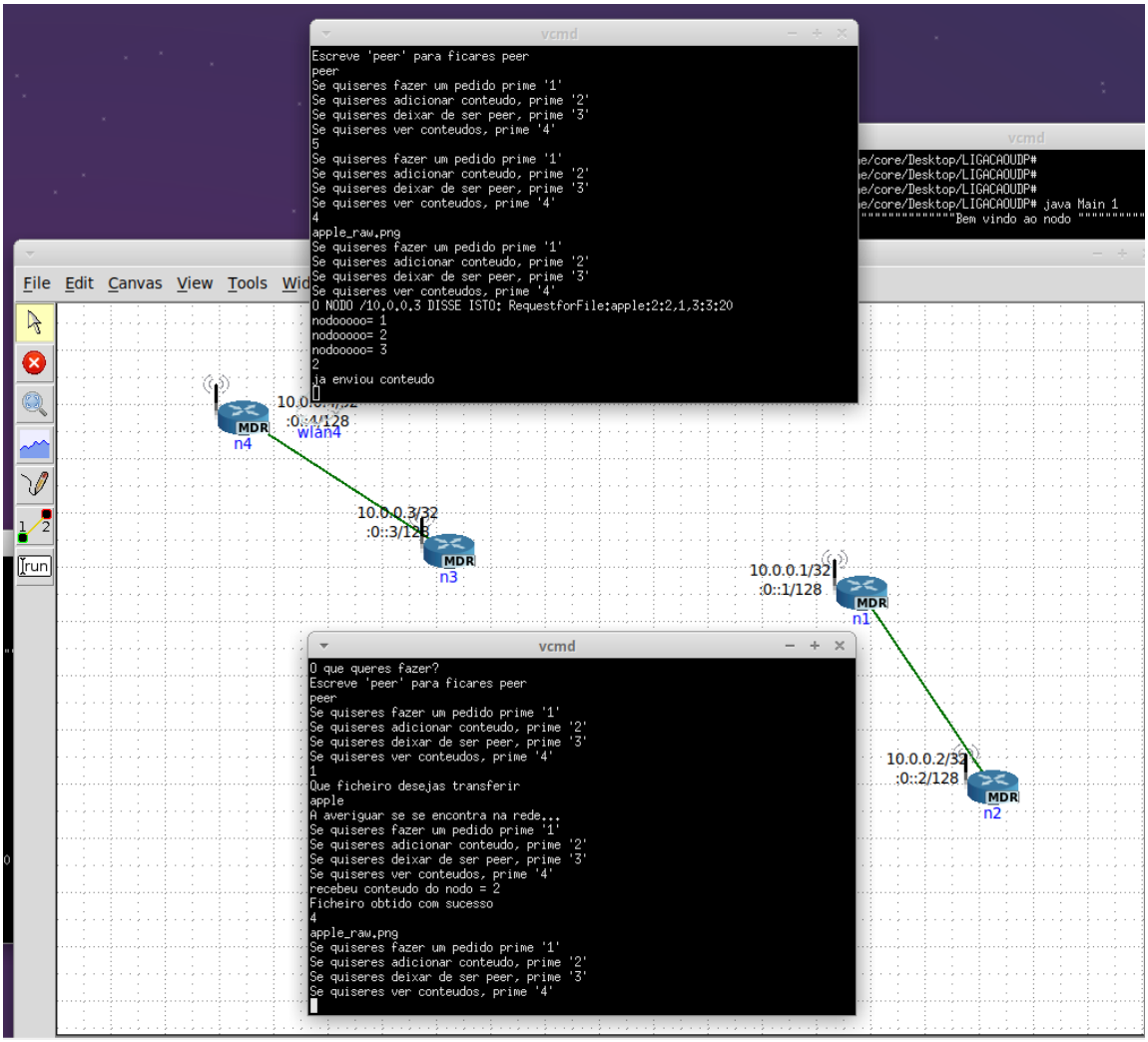

Figura 4: Pedido por transferência por parte do nodo 2 - Fase 2

Na imagem acima, podemos ver uma tentativa de pedido do pacote de interesse "apple", que, como acima foi explicado, chegou ao nodo 4 e, pelo caminho contrário, chegou ao nodo 2 o ficheiro "apple raw.png".

5 Conclusão

Após a realização deste trabalho prático, novos conhecimentos foram adquiridos sobre as redes P2P, NDN's e DTN's. Relativamente à primeira parte, o grupo considera que fez um bom trabalho, pois garante todas as condições pedidas. Achamos que o sistema elaborado permite eficiência e escalabilidade, caso fosse necessário.

Quanto à segunda fase, também consideramos que foi feito um trabalho satisfatório, no entanto, temos alguns pontos a salientar sobre a modelação desta arquitetura DTN. Quanto ao facto de não existir um controlo de perda de dados entre os nodos quando se passa conteúdo, o grupo percebe que isto é uma falha, pois assim não se garante que quando um nodo se desconecta enquanto estava em comunicação com outro para passar conteúdo, possa voltar a enviar dados, a partir do momento em que se desconectou. Consideramos este o nosso maior problema relativo à segunda fase, no entanto, o facto de os nodos poderem sair da rede, enquanto não estão em comunicação, voltar e perceber a quem devem enviar os dados era a funcionalidade maior, que foi concluída com sucesso. O facto de agora se utilizar pacotes de interesse, derivado da implementação NDN's, tornou a arquitetura mais elegante e mais fácil de pedir conteúdos.

De uma forma geral, o grupo considera que fez um bom trabalho atingindo praticamente todos os objetivos a que nos propusemos.