Implementação de um Serviço de Notícias numa Rede Veicular

Nuno Areal(A74714), Mário Silva(A75654)

Arquiteturas Emergentes de Redes - MiEI
Universidade do Minho

Abstract. Neste relatório explicamos de que forma é que realizamos as diferentes partes do trabalho prático 2. Este trabalho tem como base o trabalho 1 sendo efetuadas alterações de modo a que funcione eficientemente neste tipo de rede, uma rede tolerante a atrasos (DTN). Como no primeiro explicaremos quais os protocolos que utilizamos, de que forma os utilizamos, como foram projetados, qual a estratégia utilizada e de que forma são uma boa solução para a rede Veicular, focando principalmente a estratégia escolhida para a sua implementação.

1 Introdução

Este trabalho prático baseia-se na implementação de um protótipo de um serviço de notícias numa rede veicular através de um protocolo de encaminhamento e de um protocolo aplicacional. Esta implementação supõe as seguintes funcionalidades:

- Cada nó apenas conhece os seus vizinhos através do envio de receção de mensagens HELLO por UDP no endereço FF02::1 em *multicast* na porta 9999.
- O serviço de notícias será uma aplicação de utilizador que comunicará através de TCP com a aplicação de encaminhamento, tendo como objetivo retornar noticias através de uma mensagem GET_NEWS_FROM e obtê-las no formato NEWS_FOR.
- Cada nó é capaz de armazenar estes tipos de mensagens e destingui-las univocamente com o objetivo de as entregar ao destino no momento de receção ou num futuro próximo.
- Ser capaz de evitar a distribuição epidémica das mensagens, utilizando alternativas.

2 Estratégia de Reenvio

Nesta secção explicaremos qual a estratégia de encaminhamento que utilizamos para efetuar a troca de noticias numa rede do tipo DTN e que melhoramentos são efetuados em relação ao método clássico denominado epidémico.

2.1 Conceção

A estratégia de encaminhamento concebida é baseada numa já existente denominada *two-hop relay*, em que o emissor efetua N cópias e envia-as para os primeiros N nós com que contacta, passando haver N + 1 nós da rede responsáveis por fazer a mensagens chegar ao destino. Esta estratégia tem uma boa probabilidade de entrega ao destinatário e o flooding é muito controlado evitando assim o grande problema de técnicas de encaminhamento, como a epidémica. Apesar das melhorias significativas no desempenho, passa agora a existir a probabilidade de as mensagens nunca serem entregues se nenhum dos N + 1 nós contactarem com o destinatário, por isto adaptamos o metodo de escolha de nós, introduzindo alguma aleatoriedade e avaliação qualitativa de modo a escolher nodos que aumentem a probabilidade de entrega.

2.2 Implementação

Na adaptação desenvolvida passamos a utilizar a tabela de encaminhamento que armazena uma variável que permite avaliar o tempo que um nó fica continuamente próximo do emissor. Quando for necessário escolher os nós para onde serão enviadas cópias, terão prioridade os que estão conectados ao emissor à menos tempo. Esta escolha qualitativa aumenta a probabilidade de fazer chegar mensagens a destinatários fora da area do emissor. Para o teste da estratégia foram escolhido 2 nós pelo método descrito acima e 3 para os 3 próximos nós a conectarem-se pela primeira vez.

2.3 Avaliação

A estratégia desenvolvidade efetua melhoramentos significativos de desempenho e eficiência de utilização de recursos em relação ao epidémico, sendo baseado em *two-hop relay* e ainda efetua otimizações a este, aumentando a probabilidade de entregua a nós mais distantes e que permanencem constantemente fora da sua área. Apesar disto continua a existir a probabilidade de não serem entregues mensagens especialmente a nós mais isolados, que apenas tem contacto com um número reduzido de nós na rede. Esta estratégia é mais eficiente quanto menor for a área total em que um nó pode circular e quanto maior for área que cada nó circula.

3 Implementação

Tudo foi desenvolvido tendo como ponto de partida o primeiro trabalho prático, modificando algumas partes e removendo outras desnecessárias para uma DTN. Caso disso são os ROUTE_REQUEST e ROUTE_REPLY que obviamente não são utilizados em redes DTN.

3.1 PDUs

Existem três tipos diferentes de PDUs na nossa aplicação, passando a descrever o seu formato.

HELLO Para o envio é criada uma thread que a cada 500 milisegundos envia uma mensagem HELLO através de multicast.

GET_NEWS_FROM Encontra-se no formato

```
GET_NEWS_FROM <IP origem> <IP destino> <Timestamp>
```

exceto quando são enviadas do cliente para o protocolo de encaminhamento onde não contêm o timestamp.

NEWS_FOR Encontra-se no formato

```
NEWS_FOR <IP origem> <IP destino> <Timestamp> <Notícias>
```

3.2 Classes e interações

Nesta secção iremos explicar o que faz cada classe do projeto e que tipo de interações têm com os PDUs e de que forma os usam e constroem.

No Esta classe representa um nó que estará na tabela de encaminhamento e contém os parâmetros endereço, saltos até chegar ao endereço, número de hellos recebidos, uma blocking queue utilizada para adicionar os pacotes HELLO recebidos, para posterior processamento.

Message Esta classe contém a especificação do conteudo de uma mensagem do tipo GET_NEWS_FROM e NEWS_FOR com os seguintes parâmetros:

- ipFrom - IP origem da mensagem
- ipTo - IP destino da mensagem
- mess - Noticia
- timestamp - Tempo de criação da mensagem
- type - True=¿GET_NEWS_FROM; False=¿NEWS_FOR

DTN é a responsável por toda a execução do protocolo de encaminhamento. Efetua o start às varias threads necessárias para o funcionamento, nomeadamente a que efetua o envio de HELLO, HelloSendThread, a que efetua o processamento de pacotes recebidos por multicast, MulticastReceiveThread, a que processa as conexões TCP, TCPThread, a que trata de todos os pacotes recebidos por unicast, UnicastReceiveThread e da que efetua a limpeza de mensagens, MessageCleanerThread. Esta também efetua o print de uma interface de opções que permite ao utilizador imprimir a tabela de encaminhamento através da thread PrintThread.

HelloSendThread é uma das threads iniciadas pela DTN e esta envia as mensagens de HELLO por multicast, chegando a todos os nós que tenham uma conexão com o nó que as envia.

MulticastReceiveThread recebe todos os pacotes que são enviados por multicast, ou seja os HELLO. Os procedimentos mais relevantes são a criação de uma thread, HelloReceiveThread, por cada vizinho direto e redirecionamento de todos os pacotes HELLO seguintes para esta thread e também o envio de mensagens para o nó que enviou a mensagem, caso este seja um novo nó.

SendMessagesThread Esta thread é utilizada para enviar as mensagens a um novo nó que se conecta, se estas existirem. São também enviadas mensagens de GET_NEWS_FROM e NEWS_FOR que foram criadas pelo nó, que passam a ser também distribuidas pelo nó que se ligou. Esta thread foi criada devido a isto ser feito na thread MulticastReceiveThread que está à escuta de pacotes multicast. Se não fosse feito numa thread separada poderiam ser perdidos pacotes devido ao processamento que é necessário fazer para o envio de mensagens.

UnicastReceiveThread recebe todos os pacotes que são enviados por unicast, nomeadamente GET_NEWS_FROM e NEWS_FOR. Esta cria um thread da classe *HandleUnicastPacket* que irá efetuar todo processamento deste pacotes.

HandleUnicastPacket responsável por processar todos os pacotes recebidos por unicast do tipo GET_NEWS_FROM e NEWS_FOR.Os procedimento mais relevantes são o reenvio dos pacotes para os destinos por unicast caso este esteja diretamente conectado ou para o seu servidor se o próprio for o destino final assim como o armazenamento de mensagens para entregar no futuro.

HelloReceiveThread tem como função a implementação do parâmetro dead interval, que declara o vizinho inatingível caso este não envie nenhuma mensagem de HELLO num determinado período de tempo, eliminando-o da tabela de encaminhamento.

TCPThread Responsável pelo processamento de pacotes que chegam através de TCP. Esta verifica se se trata de um GET_NEWS_FROM ou de um NEWS_FOR. Qualquer pacote deste tipo que seja recebido por esta thread deve ser entregue no localhost, logo esta passa a reenviar os pacotes por TCP para o cliente ou servidor.
É também nesta thread que são criadas as duas threads HandleSocket, uma para o cliente e outra para o servidor. Isto é feito através da receção de mensagens SERVER e CLIENT enviadas pelo servidor e cliente, respetivamente. De seguida é criada a thread que processa toda a informação recebida por estes.

HandleSocket é responsável por tratar de toda a comunicação TCP entre cliente e protocolo de encaminhamento e também entre o servidor e o protocolo de encaminhamento. Toda a comunicação é feita através de TCP. Aquando da receção de mensagens estas são reenviadas por unicast UDP quer para o destinatario, se disponível, como para nós vizinhos através do método explicado anteriormente.

MessageCleanerThread efetua limpeza, a cada 5 minutos, das mensagens a enviar diretamente a outros nós e das mensagens a fazer "flood".

PrintThread efetua o print para o ecrã do estado da tabela atual com os parâmetros endereço do nó vizinho e numero de saltos.

Cliente representa o protocolo aplicacional que é independente do programa que implementa o protocolo de encaminhamento. Esta é composta por uma interface que permite efetuar um pedido de GET_NEWS_FROM a um outro nó através do seu endereço IP, criando uma conexão TCP com o programa que esta a efetuar o protocolo de encaminhamento no host, enviando este pedido e ficando à espera de uma resposta. Se esta não chegar dentro do tempo limite é escrita no ecrã uma mensagem de erro, avisando que o pedido expirou.

GetNewsThread Responsável pelo establecimento da conexão entre o cliente e o protocolo de encaminhamento e de todo o processo de envio de GET_NEWS_FROM e da receção da resposta ao pedido realizado. Também é esta que impõe o time-out TCP de 5 minutos e caso este limite seja atingido informa o cliente que o pedido expirou.

Server é a segunda parte do protocolo aplicacional. Permite fazer o envio de noticias aquando de uma receção de um pedido. Inicialmente é enviada a mensagem SERVER, para identificação e posteriormente fica à escuta de mensagens. Assim que recebe uma, responde com um NEWS_FOR para o IP origem com as noticias.

HandleRequest Ao receber uma mensagem do tipo GET_NEWS_FROM por TCP é enviada uma uma resposta do tipo NEWS_FOR com o ip de origem, ip destino, timestamp de criação assim como a noticia atual do servidor.

3.3 Parâmetros

- MulticastSocket: 9999 (UDP)
- DatagramSocket: 6666 (UDP)
- Multicast Group: FF02::1
- ServerSocket: 9999 (TCP)
- Hello Interval: 500 milisegundos
- Dead Interval: 700 milisegundos
- Timeout TCP: 5 minutos
- Limpeza de Mensagens: 5 minutos

3.4 Biliotecas de funções

- MulticastSocket
- InetAddress
- DatagramPacket
- BlockingQueue
- ArrayBlockingQueue
- NetworkInterface
- BufferedReader
- Socket
- InputStreamReader
- DataOutputStream
- PrintWriter
- ServerSocket
- DatagramSocket

4 Testes e Resultados

Para testar o nosso programa efetuamos testes com o *sample4.imn* fornecido pelo CORE em conjunto com o modelo de movimento. De seguida, executamos um servidor no nó n4 e um cliente no n8, fazendo um pedido de noticias logo no inicio da simulação onde os dois nós nao estão conectados, obtendo resposta quando eles se ligam um ao outro pois acontece logo de seguida. Testamos também quando os dois nós estão diretamente conectados onde obtivemos resposta instantânea. Outra situação foi a de os dois nós estarem desconectados, fazer o pedido e este ser reencaminhado por outros nós. A entregas das mensagens também se verificou. Por último a caducação de mensagens pode ser verificada se não correrrmos a simulação, nunca se encontrando nós suficientes para fazer circular a mensagem, uma vez que implementamos *two-hop relay*.

5 Conclusões e trabalho futuro

Com este trabalho podemos perceber melhor o funcionamento de redes DTN, neste caso especifico uma rede Veicular, especialmente as diferenças que existem entre este tipo e as redes Adhoc, realizado no primeiro trabalho, a denotar a necessidade de armazenar estado tanto das mensagens a entregar como das entregues de modo a reduzir o *flooding* na rede, que é o principal problema do método epidémico.
Penso que atingimos os objetivos pretendidos para o trabalho de fazer a implementação de um protocolo de encaminhamento que funcionasse corretamente numa rede DTN e que apresenta-se melhoramentos significativos ao problema de *flooding* típicos de uma estratégia epidémica.
No entanto poderiam ser feitos ajustes no método de escolha de nós assim como dos parâmetros apresentados para um caso concreto onde fossem descritas mais detalhadamente informações sobre os nós, como o tipo de movimento, a área abrangida, velocidade, etc.
Tendo como base o que aprendemos com a realização deste trabalho poderíamos no futuro desenvolver um protocolo de encaminhamento para redes DTN mais eficiente, evitando ainda mais os pacotes desnecessários a circular na rede, aumentar as probabilidades de chegada ao destino assim como outros que não foram focados neste projeto como minimização do tempo necessário para receber a resposta a um pedido.