Serviço de Notícias em Rede Veicular

Bruno Chianca Ferreira, pg33878
João Rui de Sousa Miguel, a74237
Paulo Jorge Machado Guedes, a74411

Universidade do Minho, Gualtar, Braga

Resumo. Neste trabalho foi desenvolvido um serviço de notícias numa Rede Veicular tolerante a atrasos. Isto significa que os veículos usados para transportar essas mesmas notícias poderão não estar em constante comunicação. Foram desenvolvidos vários algoritmos de transmissão de pacotes, evitando ao máximo o *flooding* da rede, mas garantindo uma taxa de entrega elevada, combinando aspetos de aleatoriedade com inspeção de padrões de comunicação entre os nodos. Foram criados vários métodos, 6 em específico, de modo a que as notícias pudessem ser pedidas e entregues. Para tal, guarda-se informações relevantes para evitar reenvio de pacotes para o mesmo destino. De forma a que os pedidos e entregas de mensagens fossem apagadas logo que o seu propósito fosse cumprido, e de modo a evitar *flooding*, obtendo informações sobre a rede, desenvolveu-se um algoritmo no qual a entrega de pedidos e mensagens seja o mais eficiente possível. Decidiu-se sacrificar espaço de disco de modo a melhorar a eficiência do algoritmo, visto que nos dias de hoje o segundo aspeto é o mais importante e difícil de obter.

Keywords: UDP, Rede Veicular, IPv6.

1 Introdução

Foi-nos solicitado o desenvolvimento de um Serviço de Notícias numa Rede Veicular (Rede Oportunista/Tolerante a Atrasos). Deste modo, tivemos de tomar em conta vários aspetos, como a comunicação intermitente de veículos, onde tem de ser calculado o melhor trajeto para o veículo destino através de padrões de comunicação e/ou aleatoriedade, evitando ao máximo a técnica de *flooding* da rede (o envio de todas as mensagens para todos os nodos, num algoritmo epidémico).

Inicialmente foi desenvolvido no âmbito do protocolo, um método para notificar os veículos que conseguem comunicar uns com os outros, o método "hello". Este pacote deverá ser enviado periodicamente, notificando os veículos próximos da possível troca de dados entre estes. De seguida foram desenvolvidos mais dois tipos de métodos, "GET" e "NEWS", cujo objetivo será, respetivamente, o pedido das notícias e a entrega destas mesmas. Estas duas mensagens serão entregues dependendo de um algoritmo, explicado posteriormente, que combina uma entrega pré-determinada e aleatória de mensagens, dependendo das comunicações entre nodos. Para evitar a contínua propagação destas duas mensagens foi criado o método "delete", com o obje-

tivo de remoção de mensagens cuja propagação já não é necessária. Este método em específico não provocará *flooding* na rede, pois a partilha desta será calculada de uma forma determinística, não possuindo nenhuma componente aleatória. Para uma propagação mais precisa das mensagens "GET" e "NEWS", foram criados mais dois métodos, "conhece?" e "conheço", no qual o primeiro interroga sobre as comunicações de outro veículo, e segundo serve como resposta a esta pergunta, fornecendo informações sobre os veículos conhecidos.

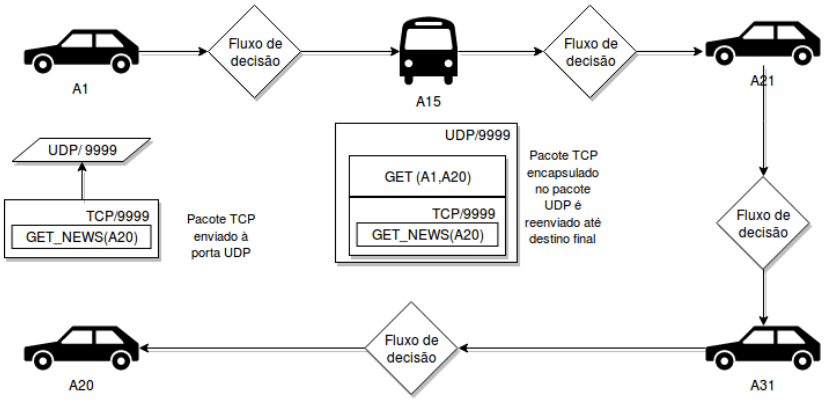

Figura 1 – Difusão de mensagens

Foi aceite como *trade-off* que seria sacrificado espaço de disco sendo guardada muita informação sobre as comunicações do veículo, compensando pelo aumento da eficiência do algoritmo de procura e entrega de notícias, visto que espaço de disco é facilmente obtido em detrimento à eficiência.

Todas as mensagens, com exceção do pedido inicial e da entrega final das notícias, foram realizadas através de UDP por IPv6, e o TCP foi usado como simulação de interface entre camada de rede e camada de aplicação.

2 Definição do Protocolo

2.1 Registos e Armazenamento

Para o cálculo do algoritmo de reencaminhamento de pacotes, foi decidido que seria guardada informação sobre a comunicação com outros veículos de modo a detetar padrões de comunicação, para poder prever com máxima eficácia quando será a próxima comunicação com um veículo específico.

Sendo assim, foi criada uma tabela que guarda o histórico das comunicações com outros veículos. Nesta tabela guarda-se os *timestamp* das cinco últimas comunicações com o veículo. É considerada uma nova comunicação se entre a comunicação atual e a última comunicação guardada na tabela, houver uma diferença superior a 20 segundos (apenas para efeitos de demonstração, caso o algoritmo fosse aplicado a uma escala real, a diferença teria de ser maior). Esta tabela servirá para calcular um padrão

de comunicação com um determinado veículo. Este padrão será calculado através da diferença média entre comunicações, podendo assim prever quando se realizará a próxima comunicação. Foram apenas guardados 5 *timestamps* para efeitos de demonstração, mas caso seja necessário aumentar a precisão, aumentar-se-ia o número de *timestamps* por veículo.

CarID	TimeStp1	TimeStp2	TimeStp3	TimeStp4	TimeStp5
nd0001	123456789	123456789	123456789	123456789	123456789
nd0012	123456789	123456789	123456789	123456789	123456789
nd0023	123456789	123456789	123456789	123456789	123456789
nd0011	123456789	123456789	123456789	123456789	123456789

Figura 2 – Tabela exemplo relativa ao histórico de contactos

Foi criada uma outra tabela, que guarda os veículos com quem comunicou recentemente (para efeitos de demonstração, os veículos cuja comunicação foi realizada num máximo de 20 segundos). Este valor deverá ser aproximadamente igual à diferença mínima entre *timestamps* da tabela de histórico, para garantir que apenas seriam guardados novos dados). Esta tabela guarda também o número de segundos que esteve em comunicação, sabendo assim a direção dos dois carros quando estabeleceram comunicação (se a comunicação foi breve, podemos afirmar que os dois carros iriam em sentidos opostos, e vice-versa), e/ou se circulam na mesma área.

CarID	ContactTime (s)	TimestampDelta
nd0001	5	3234
nd0041	15	34768
nd0021	17	98544
nd0201	2	754

Figura 3 – Tabela de contactos recentes

Estas duas tabelas serão utilizadas para a verificação determinística da comunicação do veículo com o veículo destino da mensagem, mas poderá haver ocasiões em que nem este, nem nenhum em redor a este, alguma vez comunicaram com o nodo destino da mensagem. Para compensar este facto, foi adicionada uma vertente aleatória a cada veículo, o *score*. O *score* é um valor que irá aumentar e diminuir dependendo da atividade do veículo. É aumentado quando: existe comunicação com outro nodo que não tenha conhecido recentemente; recebe uma mensagem "GET" ou "NEWS" que ainda não tenha recebido; quando entrega alguma mensagem de "GET" ou "NEWS" ao destino ou a algum veículo de forma determinística (não baseado no *score*). Este *score* é diminuído de acordo com o tempo (para efeitos de demonstração, é diminuído para metade a cada minuto).

As mensagens ainda não entregues serão guardadas numa tabela, enquanto que todas as mensagens entregues por este veículo serão guardadas noutra tabela. Isto permite que um veículo possa descartar uma mensagem de imediato caso já tenha entregue essa mensagem, ou se ainda a distribui.

2.2 Algoritmo de Entrega

Para além dos registos enunciados acima, para o algoritmo de entrega de mensagens "GET" e "NEWS" foi tomado em conta o tempo de *timeout* da mensagem, caso esta exista.

Sendo assim, o algoritmo de entrega de mensagens "GET" e "NEWS" é o seguinte:

- Caso um veículo tenha o nodo destino na sua tabela de comunicações recentes, é lhe enviada a mensagem;
- É calculada a diferença média entre comunicação da tabela do histórico de comunicações, entre o veículo destino, e caso a diferença seja menor, é lhe enviada a mensagem;
- Em cada mensagem, é guardado o *score* máximo do veículo ao qual a mensagem já foi transmitida. Assim, para cada veículo, é verificado se o seu *score* é maior que o *score* do veículo possuidor da mensagem e se é maior que o *score* guardado na mensagem. Caso se verifique, é lhe enviada a mensagem, e atualizado o *score* da mensagem em específico. O score guardado em cada mensagem, é diminuído também consoante o tempo (para efeitos de demonstração, é diminuído para metade a cada minuto), para garantir a continua propagação da mensagem, caso não seja entregue.

Para cada mensagem, é verificado o tempo de *timeout* da mensagem, não sendo entregue caso este já tenha expirado, e é eliminado da tabela de mensagens a ser entregue.

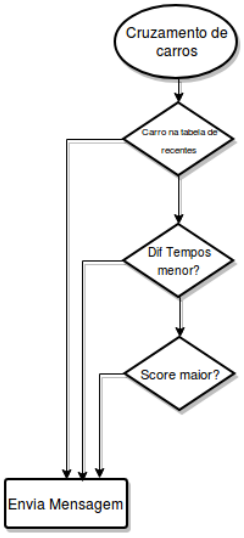

Figura 4 – Fluxo de decisão

Este algoritmo combina ambas as partes de entrega de mensagens dependo da comunicação da rede e a parte de aleatoriedade necessária para comunicações a grandes distâncias, para veículos desconhecidos. É ainda possível customizar o nível de pro-

pagação aleatória das mensagens individualmente, modificando o *score* associado a cada mensagem da maneira desejada. De seguida são explicados as 6 PDUs criadas para que se cumpra o algoritmo, e impeça o *flooding* da rede.

3 Definição das PDUs do Protocolo

3.1 PDU Hello

Serve para notificar todos os veículos por dentro do seu grupo IPv6, que este se encontra disponível para a troca de mensagens. É enviado o seu nome, em conjunto com o score do seu veículo, em *multicast* para o grupo IPv6.

Table 1. PDU "Hello"

0	nome	score

- Tipo do protocolo
- Nome de quem enviou a mensagem
- O *score* de quem enviou a mensagem

3.2 PDU GET

Serve para realizar o pedido de notícias a um nodo destino. Este contém um identificador, para impedir que sejam transmitidas mensagens duplicadas, e guarda o caminho percorrido até ao momento, impedindo que este seja transmitido a um veículo que já tenha essa mensagem. A mensagem é guardada na tabela de mensagens a ser transmitido caso não seja o destino, para poder reencaminhar para outros veículos. Caso contrário é realizado um pedido por TCP, a solicitar as notícias desse veículo. Também se guarda o *timeout* do pedido, assim como o *timestamp* de quando o foi realizado, não transmitindo após expirado esse *timeout*.

Table 2. PDU "GET"

1	nome	id	"MSG"	destino	timeout	timestamp	caminho	TCP	score

- ID da PDU
- Nome de quem enviou a mensagem
- ID da mensagem (juntamente com o nome de quem enviou, forma um identificador único para qualquer mensagem)
- "MSG"
- Destino da mensagem
- *Timeout* da mensagem
- *Timestamp* de quando a mensagem foi criada
- Caminho percorrido pela mensagem
- Pacote TCP, a enviar para a parte aplicacional, que contêm o tipo de mensagem ("GET" ou "NEWS"), a origem e o destino da mensagem, o *timeout* e o *timestamp*
- *Score* da mensagem

3.3 PDU NEWS

Utilizado para responder a um protocolo "GET", com as notícias do veículo. Contém os mesmos campos que o protocolo "GET", mas com valores novos (exceto o *timeout*, que é atualizado para quanto tempo falta desde o ponto em que criado o pacote "NEWS" até ao tempo limite de entrega). Realiza o mesmo procedimento que o protocolo "GET".

Table 3. PDU "NEWS"

2	nome	id	"MSG"	destino	timeout	timestamp	caminho	TCP	score

- ID da PDU
- Nome de quem enviou a mensagem
- ID da mensagem (juntamente com o nome de quem enviou, forma um identificador único para qualquer mensagem)
- "MSG"
- Destino da mensagem
- *Timeout* da mensagem
- *Timestamp* de quando a mensagem foi criada
- Caminho percorrido pela mensagem
- Pacote TCP, a enviar para a parte aplicacional, que contêm o tipo de mensagem ("GET" ou "NEWS"), a origem e o destino da mensagem, e as notícias
- *Score* da mensagem

3.4 PDU Delete

Criado para impedir a contínua propagação das mensagens. É criado dando um identificador, como as mensagens "GET" e "NEWS", indicando também a origem e o identificador da mensagem que deseja apagar naquele/s veículos. É também acompanhado pelo *timeout* da mensagem que deseja apagar, para o caso de, quando o *timeout* da mensagem a ser apagada expirar, e os veículos que estejam a reencaminhar essa mensagem não sejam atingidos pela mensagem "delete", esta ser apagada também.

Esta mensagem apenas é transmitida pelos veículos que contêm essa mensagem (veículos que pertencem ao caminho a percorrer ou percorrido da mensagem "delete"). A mensagem é assim propagada pois o algoritmo de reencaminhamento de notícias não contém um fator enorme de *flooding* da rede, logo a mensagem "delete" não é muito importante ao ponto de esta também dar *flood*.

Table 4. PDU "delete"

3	nome	id	origem	idd	a percorrer	percorrido	timeout

- ID da PDU
- Nome de quem enviou a mensagem
- ID da mensagem (juntamente com o nome de quem enviou, forma um identificador único para qualquer mensagem)
- Origem da mensagem a apagar

- ID da mensagem a apagar
- Caminho a percorrer
- Caminho percorrido
- *Timeout* da mensagem a apagar

3.5 PDU Conhece

Contém uma tabela de pares: nome de um veículo e um *timeout.* Caso o veículo questionado consiga chegar ao(s) veículo(s) da mensagem recebida, é criada uma mensagem "conheço" (a discutir a seguir), com os nomes dos veículos que chega a tempo. Este cálculo é realizado calculando a diferença média entre os valores da tabela que contém o histórico de comunicações com o destino. Caso este seja menor que o *timeout*, é adicionado à mensagem "conheço".

Table 5. PDU "conhece"

4	nome	timeout

- ID da PDU
- Lista de pares:
  - Nome do veículo
  - *Timeout* da mensagem que se destina a esse veículo

3.6 PDU Conheço

Contém o nome de quem enviou a mensagem, para adicionar ao caminho percorrido pela mensagem que irá ser enviada, juntamente com a lista de veículos que consegue comunicar com os veículos da mensagem "conhece".

Após receber esta mensagem, serão verificadas quais as mensagens, na sua tabela de mensagem a transmitir, cujo destino está contido da lista de veículos da mensagem "conheço".

Table 6. PDU "conheço"

5	nome	nomes

- ID da PDU
- Nome de quem enviou a mensagem
- Lista de nomes aos quais, quem enviou a mensagem, consegue comunicar dentro do *timeout*

4 Testes e Resultados

Para efeitos de teste utilizou-se um modelo de rede no emulador CORE. Definiu-se uma topologia com 16 nodos distintos, onde maior parte se encontra em movimento constante, sendo que apenas dois nodos estão imóveis. Esta abordagem visa abranger

mais casos de teste. Um ponto extra desta topologia é o facto de existir uma componente repetitiva, sendo que a cada 60 segundos a topologia reinicia. Isto ajuda a explorar uma das componentes do algoritmo, relativa à periodicidade entre contactos.

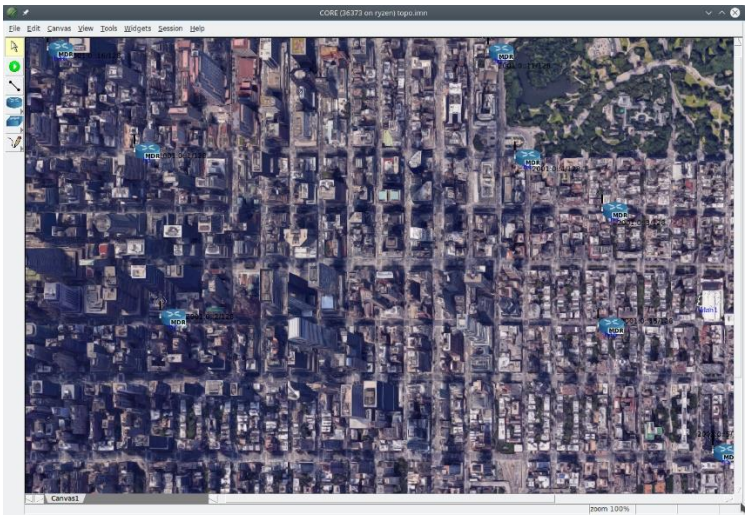

Figura 5 - Representação do canvas do Core

Nesta topologia foi possível obter comunicação/noticias entre vários nodos, sendo que para alguns deles é necessário esperar significativamente mais tempo (visto dependerem da componente aleatória). Assim, todas as comunicações aconteceram tal como esperado, demonstrando uma boa implementação do algoritmo.

5 Conclusões

O foco deste trabalho foi tentar evitar ao máximo a abordagem *flood*, e tentar criar algo que possua tanto um aspeto determinístico que nos permita percorrer com o máximo de precisão possível o caminho mais rápido, caso fosse possível, e adicionando sempre um aspeto aleatório para casos em que o caminho não é conhecido, sendo esse aspeto aleatório, num trabalho futuro, customizável, dependendo da distância da origem ao destino.

Poderá também ser feita uma melhoria, guardando mais informação na tabela de mensagem apagadas (não só a origem e o id), de modo a avaliar, quando uma nova mensagem chega, se não foi entregue uma mensagem semelhante (com a mesma origem, destino, por exemplo), com um *timestamp* mais atualizado, indicando que essa mensagem, embora que nunca tivesse sido transmitida pelo veículo, consegue-se identificar que está desatualizada, e não deverá ser retransmitida, impedindo ainda mais o *flood* da rede.