Universidade do Minho
Escola de Engenharia

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Redes Móveis

Projeto laboratorial

Grupo 1

Gil Morim

A65214

Rui Silva

A69987

Sara Pereira

A75494

Guimarães, 5 de junho de 2018

Índice

Introdução

O mundo das redes móveis tem evoluído no sentido de dotar os veículos automóveis da capacidade de comunicarem diretamente entre si. Esta comunicação será útil para que veículos transmitam informação sobre as estradas e caminhos por onde circulam.

A informação a ser transmitida poderá ser variada, desde dados sobre as condições da estrada a informação sobre o tráfego ou mesmo sobre condições atmosféricas.

O presente relatório irá debruçar-se sobre um cenário onde os veículos trocam informação sobre acidentes ocorridos no mapa que percorrem. Carros que sofreram um acidente e se encontram na estrada irão gerar mensagens com o intuito de serem espalhadas pela rede e de informarem os restantes veículos que um acidente ocorreu naquele local, criando uma oportunidade de serem evitados mais acidentes ou de evitar o tráfego que é gerado por estes.

Todo o trabalho será baseado em simulações do *The One*. Na sua implementação o grupo alterou, não extensivamente, o código de 3 classes do simulador.

Mapa da área simulada

O mapa escolhido para a elaboração deste projeto foi a cidade de Dortmund situada na Alemanha. Como podemos observar pela Figura 1, esta escolha deveu-se à sua vasta dimensão assim como o grande número de estradas existentes. O facto de ser uma cidade grande, também influenciou esta escolha, uma vez que nestas áreas há maior probabilidade de ocorrências de acidentes e maior densidade de tráfego, sendo este o objetivo deste projeto.

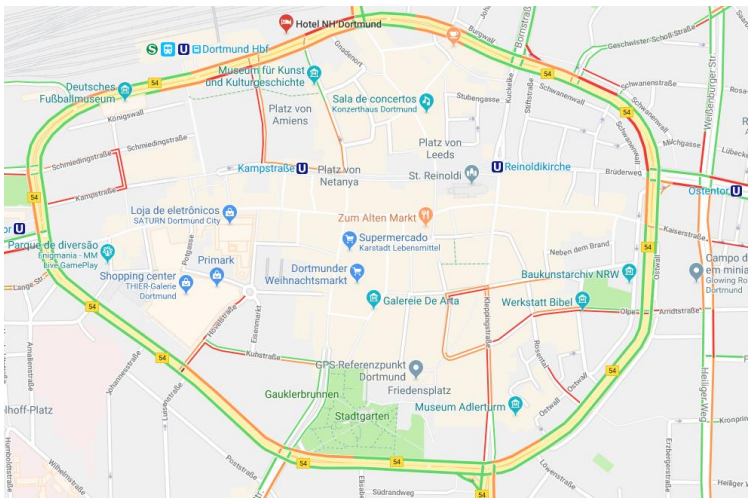

Figura 1 - Mapa da área simulada

Através de *sites* de informação de trânsito em tempo real e do Google Maps foram estabelecidos os lugares com mais densidade de trânsito e os lugares onde ocorriam acidentes. Assim, foram colocados dez *beacons* nos dez sítios com caraterísticas mais propícias a ocorrerem acidentes, como se pode observar na Figura 2.

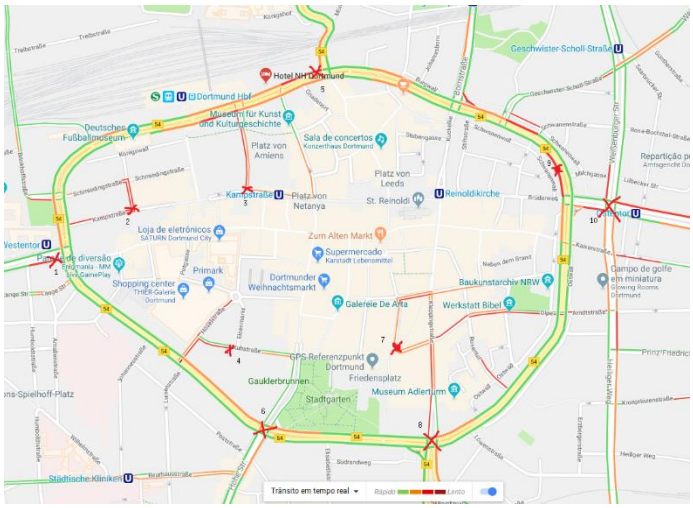

Figura 2 - Colocação dos *beacons*

Modelos de mobilidade

A fim de garantir a mobilidade dos vários veículos desta simulação, foram utilizados dois modelos diferentes: um para os carros e outro para os *beacons*.

Para os carros foi utilizado o modelo **ShortestPathMapbasedMovement** onde os carros têm um destino aleatório e seguem o caminho mais curto para alcancá-lo. Alcançando este, aguardam um instante enquanto escolhem outro destino aleatório.

Para os *beacons* foi utilizado o modelo **StationaryMovement** onde, como o nome indica, é adequado para nós da rede que se pretende que não tenham movimento. A fim de similar um acidente, a cada *beacon* foi atribuído as coordenadas geográficas de acordo com a Figura 2 e a velocidade foi introduzida com o valor 0.0 Km/h.

Protocolo de encaminhamento

Para a transmissão de mensagens na rede foi utilizado principalmente o protocolo **EpidemicRouter**. Este protocolo fornece entrega de mensagens para destinos aleatórios baseado nas pressuposições mínimas de topologia e conectividade da rede. A entrega de mensagens depende apenas da conectividade periódica entre outros nós.

Por outras palavras, **EpidemicRouter** baseia-se no *flooding*, ou seja, cada nó está constantemente a enviar e replicar mensagens para os seus vizinhos que ainda não possuam estas.

O protocolo **SprayAndWait** foi também utilizado para comparar resultados. No entanto, este protocolo não se adequa ao intuito do projeto, ou seja, não é apropriado para *broadcast* mesmo sendo baseado em *flooding*.

Os restantes protocolos não baseados em *flooding* não se adequavam nem à solução do grupo nem à necessidade do projeto. Assim, o grupo não se focou nestes.

Interfaces de rede

Como referido no enunciado, em todos os nós foram implementadas interfaces Bluetooth. A fim de determinar qual versão do Bluetooth seria a mais indicada para este projeto, o grupo pesquisou acerca das configurações destas. Como o **Bluetooth 4** e **Bluetooth 5** são as versões mais recentes, foi decidido que seriam estas as versões utilizadas nos testes experimentais. Assim, para o 4 temos um alcance de 20 metros e um *datarate* de 1 Mbps equanto que na versão 5 temos um alcance de 40 metros para um *datarate* de 2 Mbps.

Geração de tráfego

Nas simulações corridas o grupo decidiu que apenas os *beacons* iriam gerar mensagens. Os restantes nós iriam limitar-se a servir de *relays/*destino. O grupo pretendeu desta maneira conseguir obter dados apenas relativos aos atrasos das mensagens relacionadas com o acidente, e não dos restantos nós.

Para cada beacon foi então associado um *MessageEventGenerator*. Este *Event* irá ser responsável por gerar as mensagens relativas a cada acidente. Para garantir que as mensagens eram enviadas de 10 em 10 minutos, o grupo apenas conseguiu que estas começassem a ser geradas também a partir do momento 600 da simulação (10 minutos), e que todos os nós começassem a gerar em tempos próximos uns dos outros. Este fator retira algum realismo da simulação – 10 acidentes acontecerem ao mesmo tempo é improvável.

> Se não se fosse tão rígido com os intervalos de 10 minutos então seria possível fazer com que os acidentes acontecessem em diferentes momentos e assim, que enviassem mensagens em intervalos variados.

Cada mensagem é gerada com um campo *payload*, uma String que irá conter as coordenadas do *beacon* que a gerou. Este campo foi adicionado à classe *Message,* juntamente com um método getPayload e uma linha de código em que, na instanciação de uma mensagem, são obtidas as coordenadas do *host* que a criou.

Abordagem do Problema

Este projeto pretende simular redes veiculares e, mais especificamente, ocorrências de acidentes e notificação da rede destes. O grupo pretendeu aproximar o cenário simulado a uma ocorrência real.

Numa situação real seria de esperar que os carros acidentados enviassem mensagens com destino a todos os nós da rede, ou seja, em *broadcast*. Isto implica, na visão do grupo, que cada mensagem não seja criada com destino a um nó específico, mas sim com um destino a todos os nós. O The One não suporta esta funcionalidade de *broadcast*, mas o grupo alterou o código de maneira a fazer com que as mensagens criadas fossem percorrer toda a rede.

O que o grupo fez foi alterar na classe *MessageRouter* as linhas de código que verificavam se um nó era o destino final. O *boolean* que indicava que a mensagem tinha chegado ao seu destino foi definido como *false*, a não ser que essa mesma mensagem tivesse já percorrido os 200 nós da rede. Assim sendo, o grupo não implementou um *broadcast* propriamente dito, mas sim uma impossibilidade de a mensagem ser apagada se ela chegasse ao seu destino, continuando a ser propagada quasi-infinitamente, não fosse o TTL das mensagens. Adicionalmente, para garantir que as mensagens não eram reencaminhadas (e também que não consideradas recebidas) por nós que já a receberam num momento anterior, o grupo impôs uma verificação de se um dado nó já tinha recebido essa mensagem ou não: se o nó já estava incluído nos nós por onde a mensagem passou, então a mensagem era descartada por este. Esta verificação de já ter recebido

Com os métodos descritos, o grupo não consegue garantir uma taxa de entrega de 100% (nem seria possível num cenário real), mas garante tentar aproximar o mais possível a simulação a um cenário real. Esta solução implica, no entanto, que protocolos dependentes de um destino final não funcionem corretamente, ao invés de protocolos de *flooding*, que são os únicos capazes de garantir que a mensagem poderá chegar a todos os nós.

Para testar a sua solução o grupo criou uma série de testes em que variava um conjunto de fatores. Numa tentativa de obter a maior variedade de resultados possível e para obter uma melhor compreensão do problema, 18 simulações diferentes foram testadas. Os parâmetros que foram testados são os seguintes:

- **Tipo de interface de rede:** O grupo simulou a utilização de Bluetooth 4.0 e de Bluetooth 5.0, nomeadamente os seus respetivos *datarates* e alcance;
- **Tamanho dos *buffers* dos nós:** A capacidade que os nós têm para manter mensagens mais antigas é um fator a ser considerado;

- ***Time-to-live* das mensagens:** O tempo que uma mensagem fica "viva" na rede irá influenciar os atrasos que estas sofrem;
- **Tempo de duração da simulação:** O grupo pretendeu verificar se o tempo que a simulação corre tinha qualquer efeito relevante nos atrasos;
- **Tipo de *routing:*** Tal como mencionado, apenas protocolos de *flooding* correspondem à solução pensada pelo grupo e à situação real imaginada.

Os 18 testes criados basearam-se à volta do protocolo de *routing Epidemic* e uma duração de 5 horas da simulação. 9 desses testes foram feitos simulando Bluetooth 4.0, com 1 Mbps de *datarate* e um alcance de 20 metros.

	Buffer Size (Mbits)	TTL (minutos)
SIM1	5	10
SIM2	10	10
SIM3	1	10
SIM4	5	30
SIM5	10	30
SIM6	1	30
SIM7	5	60
SIM8	10	60
SIM9	1	60

Os restantes 9 testes foram feitos simulando Bluetooth 5.0, com 2Mbps de *datarate* e um alcance de 40 metros.

	Buffer Size (Mbits)	TTL (minutos)
SIM10	5	10
SIM11	10	10
SIM12	1	10
SIM13	5	30
SIM14	10	30
SIM15	1	30
SIM16	5	60
SIM17	10	60
SIM18	1	60

Um dos finais parâmetros do simulador configurado foi o rngSeed. É a partir deste parâmetro que os elementos aleatórios da simulação (como os caminhos tomados pelos nós) são gerados. Sem a configuração do rngSeed, em sucessivas simulações, cada nó faria exatamente o mesmo percurso, perdendo-se assim algum valor na fiabilidade da simulação. Esta caraterística permite, no entanto, testar a fiabilidade de um dado protocolo ou confirmar valores obtidos, o que será especificado na fase de testes do relatório.

O grupo recorreu também aos relatórios que o simulador gerava para tirar as suas conclusões e estatísticas sobre a simulação. Em particular 2 relatórios foram utilizados: o *EventLogReport* e o *CreatedMessagesReport*. Foi criado um *parser* que recorreu aos dois relatórios para fazer os cálculos de atraso.

Na classe do *EventLogReport* foram feitas algumas alterações de modo a tornar o relatório mais legível e menos pesado para o parser. Neste relatório são guardados praticamente todos os acontecimentos do simulador e, em particular, todas as conexões que se criam entre dois nós e que posteriormente se desfazem. Todas estas entradas sobre conexões, que eram a esmagadora maioria do relatório, foram removidas. Para fazer isto o grupo comentou as linhas de código que iriam fazer *print* das conexões, tanto no método *hostConnected* como no método *hostDisconnected*.

O *parser* de texto foi criado em linguagem Java, e utiliza os ficheiros de report gerados pelo simulador para calcular as estatísticas da simulação. O primeiro ficheiro analisado é o *CreatedMessagesReport*. A partir de cada uma das linhas do ficheiro, o programa extrai uma lista das mensagens criadas pelos beacons, e o instante de tempo em que foram criadas, e são pesquisadas ocorrências dessas mensagens no ficheiro *EventLogReport*, nomeadamente os eventos de receção das mesmas. A cada evento de receção, o programa retira informação sobre instante de tempo em que a mensagem foi recebida, nó emissor, nó recetor e nome da mensagem.

Utilizando os tempos de emissão e receção, consegue-se obter o atraso da mensagem, e, comparando os atrasos de cada uma delas, obtem-se o atraso máximo, mínimo e médio. Também conseguiu-se obter a percentagem de nós atingidos por cada mensagem, e através da comparação, obter as mensagens com maior e menor alcance.

Testes experimentais e Análise

Para todos os testes mencionados anteriormente os resultados foram compilados na seguinte tabela:

	Atraso médio	Mediana do atraso	Atraso máximo	Maior % de entrega*	Menor % de entrega**
SIM1	203,26	191,50	599,90	99,52	23,69
SIM2	199,06	188,10	593,50	99,5	12,79
SIM3	197,68	186,40	599,50	99,5	0,40
SIM4	206,33	190,69	1799,69	100	11,30
SIM5	212,29	196,90	1798,50	100	3,70
SIM6	209,70	191,70	1799,69	99,5	1,80
SIM7	207,36	189,40	3592,80	99,5	2,84
SIM8	211,04	194	3599,50	99,5	0,90
SIM9	207,34	188,09	3595,59	99,4	2,88
SIM10	141,16	131,80	598,80	99,5	28,43
SIM11	139,90	132,29	599,90	99,3	34,50
SIM12	135,80	128,40	598,50	100	7,50
SIM13	145,50	131,09	1799,60	100	31,70
SIM14	148,59	133,50	1799,50	100	20,30
SIM15	145,55	130,70	1799,50	100	37,90
SIM16	152,50	135,50	3599,50	100	30,80
SIM17	148,40	131,79	3598,80	100	19,90
SIM18	151,24	133,69	3599,60	100	9,95

* Este campo demonstra o valor da mensagem que obteve maior percentagem de entrega

** Este campo demonstra o valor da mensagem que obteve menor percentagem de entrega

Olhando para os resultados o que sobressai é a variação dos resultados de acordo com a interface de rede utilizada, sendo que os restantes parâmetros não influenciam muito nos atrasos de propagação das mensagens. Seria de esperar que a interface de rede tivesse o maior impacto, no entanto, o grupo não esperava que os tamanhos de *buffer* não tivessem influência nos atrasos.

O atraso máximo na receção das mensagens está associado ao valor de TTL da mensagem, o que significa que existem nós que receberão algumas mensagens perto do fim do tempo de simulação, ou então nem chegarão a recebê-las. Verifica-se ainda que com o Bluetooth 5 há maior probabilidade de as mensagens serem recebidas, obtendo melhores resultados da simulação já mencionados.

Neste tipo de redes, os atrasos esperados são muito grandes quando comparados a uma rede infraestruturada, andando na ordem dos 2 minutos. No entanto para este cenário, os atrasos obtidos são muito bons, pois consegue-se uma boa propagação de informação acerca do acidente.

Um fator importante a notar é que as tabelas do atraso máximo e as tabelas de percentagens de entrega não estão associadas. Ou seja, o atraso máximo foi verificado para a mensagem que, de entre todas aquelas criadas, demorou mais tempo a ser entregue ao seu último destino.

As mensagens com menor percentagem de entrega têm este valor intrínsicamente associada ao tempo de simulação. Esta pode acabar antes da mensagem ser propagada corretamente, causando esta baixa taxa de entrega.

É ainda improvável, num cenário real, que existam mensagens com taxa de 100%, no entanto não é impensável que num cenário desta dimensão existam as altas taxas da ordem dos 80% a 90% que o grupo verificou.

Spray And Wait

De modo a obter uma melhor análise o grupo testou ainda a utilização do protocolo *SprayAndWait*, mesmo este não sendo adequado à solução que o grupo desenvolveu. Para este protocolo foram apenas feitas duas simulações de 2000 segundos em que os *buffers* tinham 10 MB e as mensagens um TTL de 10 minutos. A difença entre os dois testes foi apenas na interface de rede usada. O número de cópias da mensagem gerada era de 6.

	Atraso médio	Mediana do atraso	Atraso máximo	Maior % de entrega*	Menor % de entrega**
BT5	61,50	21,69	578	2,80	2,30
BT4	96,40	56,45	575,60	2,80	2,30

* Este campo demonstra o valor da mensagem que obteve maior percentagem de entrega

** Este campo demonstra o valor da mensagem que obteve menor percentagem de entrega

Para os resultados obtidos com *SprayAndWait* verifica-se mais uma vez que apenas a interface de rede influência os resultados. Os valores de atraso encontram-se menores, mas apenas porque um número limitado de mensagens é propagado, baixando obrigatoriamente os atrasos. Todas as mensagens disponíveis são entregues o mais cedo possível. A percentagem de entrega é também coerente com o número de cópias.

Vendo os resultados verifica-se também que as percentagens de entrega dependem, como é óbvio, do número de cópias definido para este protocolo de routing. Isto vem a confirmar que este protocolo não é apropriado para uma situação onde se pretende espalhar um evento por toda a rede. A percentagem de entrega é proporcional ao número de cópias permitidas, o que implica que para obter taxas de entrega altas será necessário ter um número de cópias igual ao número de veículos da rede (isto também porque a solução do grupo não se adequa ao uso deste protocolo) e assim

praticamente replicar o número de mensagens a correr a rede para um número semelhante ao *Epidemic*. Assim sendo, o protocolo *SprayAndWait* não oferece uma melhor capacidade de entrega na situação simulada, mantendo-se o *Epidemic* como a melhor opção.

Fator rngSeed

Um outro campo de testes que o grupo aplicou passou por utilizar o rngSeed. Todos os testes e avaliações anteriormente feitos anteriormente usavam rngSeeds diferentes entre si, de modo a criar diferentes cenários e aproximar as simulações a possíveis situações reais.

No entanto, o grupo também repetiu todas as simulações (Sim1 a Sim18) para um mesmo cenário, ou seja, não alterando a rngSeed. Nesta situação, para a Sim1 e para a Sim2 por exemplo, os nós móveis iriam percorrer exatamente os mesmos caminhos. O que o grupo verificou foi que os resultados obtidos eram exatamente os mesmos quando os tamanhos dos *buffers* mudavam, ou seja, tanto o atraso médio, a sua mediana e atraso máximo, eram iguais. Estes valores apenas mudavam com o TTl da mensagem e com a mudança da interface de rede.

Este fator veio a suportar as conclusões a que o grupo chegou, ao dizer que o maior fator nas simulações era a interface de rede.

Conclusão

Fazendo uma análise crítica dos resultados obtidos pode-se concluir que esta abordagem é de fato uma possível solução para o problema apresentado. Seguindo uma política de *broadcast*, num ambiente não muito congestionado, verificou-se que uma mensagem relativa a um acidente demoraria em média entre 2 a 3 minutos a chegar à maioria dos nós presentes na rede. Tendo em conta este atraso pode-se concluir que os carros seriam avisados da ocorrência de um acidente atempadamente, a não ser que estivessem já muito próximos do acidente. Com o conhecimento do acidente meros minutos após este ocorrer os carros poderiam mudar as suas trajetórias, ou pelo menos saber que no seu caminho iria existir um obstáculo, podendo assim se precaver.

A estratégia que o grupo seguiu para resolver o problema é também uma muito realista e implementável. A maior fraqueza que o grupo verifica é no método que verifica se um nó já recebeu uma mensagem ou não: uma mensagem num cenário real provavelmente não teria uma lista de todos os nós que percorreu, tornando esta verificação impossível. O fato de existir um TTL de mensagem garante, no entanto, que para esta estratégia as mensagens não permaneçam infinitamente na rede, nem que viagem para além do local geográfico onde a sua existência se justifique. Tendo em conta também que o tamanho dos *buffers* não influenciou os atrasos e que esse tamanho pode ser considerado muito reduzido para as capacidades atuais de um dispositivo de memória, o grupo considera que esta estratégia é escalável. Muito facilmente um nó poderia ter *buffers* na ordem dos gigabytes, atenuando o peso incontornável que um protocolo *Epidemic* tem numa rede.

Esta implementação apenas recorreu à mudança de 3 classes: *MessageRouter*, *Message* e *EventLogReport*.

Em suma, o grupo considera que a sua implementação pode ser aplicada num cenário real e que se as condições da simulação realmente se aguentassem então um sistema de difusão da ocorrência de acidentes seria de facto viável.