Projeto Laboratorial

Redes Móveis

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Eduardo Miguel Mendes da Silva (A70216)

Joana Isabel Taveira Abreu (A68471)

João Pedro Mendes Pereira (A68454)

Guimarães, junho de 2017.

Índice

Índice de acrónimos

DTN - Delay Tolerant Networks

Índice de figuras

1. Introdução

Na unidade curricular de Redes Móveis foi proposto efectuar um trabalho laboratorial, que consiste em avaliar a fiabilidade da monitorização do tráfego rodoviário nos espaços urbanos, utilizando sensores por radiofrequência. A monitorização do tráfego rodoviário é bastante importante, uma vez que ajuda no planeamento do espaço e das redes de transporte. E por isso, hoje em dia essa monitorização é efectuada através de dois tipos de sensores, sendo eles, os anéis inductivos embebidos no solo e as câmaras de vídeo. Contudo, é necessário procurar alternativas eficientes que consigam atingir os mesmos objectivos que os sensores utilizados no presente, e por isso, iremos avaliar a utilização de sensores por radiofrequência.

Para chegar ao pretendido, irão ser colocados em vários locais da cidade os dispositivos fixos, *beacons*, que irão emitir mensagens por rádio (*Bluetooth*) que contenham a identificação do local e a hora. Para além disso, noutros locais da cidade também serão instalados dispositivos fixos, sendo estes os portais, que permitirão aos veículos entregar as mensagens recolhidas para a *Internet*. Isto é, sempre que um veículo passar pelo de um *beacon* recolhe uma suas mensagens geradas e entrega-as aos portais, adicionando à mensagem a sua identificação.

Desta forma, iremos avaliar a fiabilidade destes sensores, determinado a percentagem de mensagens recolhidas dos *beacons* que são enviados para a *Internet*, e o atraso máximo e médio ocorrido desde a criação de uma mensagem por um *beacon* até ao seu envio para a *Internet*, que será interpretado como uma entrega a um portal.

Todo esta solução será abordada em simulação, e para isso, iremos utilizar o projeto *open-source*.

De forma a descrever todo o trabalho desenvolvido apresentamos o presente relatório, que apresentará em primeiro lugar uma parte relativa ao enquadramento teórico onde abordamos alguns conceitos importantes para o desenvolvimento do trabalho. De seguida, irá apresentar uma parte relativa ao desenvolvimento do trabalho onde dividimos em duas fases de trabalho, e por último, a parte relativa aos resultados experimentais obtidos.

2. Objectivos

Para o desenvolvimento deste trabalho, pretende-se atingir os seguintes objectivos:

- Conhecer o funcionamento das DTNs (Delay Tolerant Networks);
- Conhecer os diferentes protocolos de encaminhamento;
- Conhecer os diferentes modelos de mobilidade;
- Avaliar a fiabilidade da solução utilizando os sensores por radiofrequência em ambiente de simulação;
- Determinar o atraso médio e máximo das mensagens desde a recepção de um *beacon* até ao seu envio para a *internet*, e a probabilidade da entrega das mensagens aos portais.

3. Enquadramento teórico

Nesta secção, iremos abordar alguns conceitos importantes, como, as DTNs, os procolos de encaminhamento e os modelos de mobilidade.

3.1. DTNs

Delay Tolerant Networks tem o potencial para conectar dispositivos e locais do mundo que não estão ligadas pelas redes tradicionais. As DTNs permitem a comunicação tirando vantagem das conexões temporárias para retransmitir informação de uma forma similar à rede de correio, em vez de requerer uma ligação de rede *end-to-end* disponível.

Estas redes são desenhadas para operar sobre grandes distâncias, tais como, as comunicações espaciais ou até numa escala interplanetária. Nesse ambiente é inevitável a latência ser medida em horas ou até dias.

Os atuais protocolos TCP/IP operam segundo o princípio de que existe uma ligação *end-to-end* utilizando a concatenação de tecnologias diferentes da camada de ligação. Estes protocolos da Internet não funcionam bem em ambientes tolerantes a atrasos e comunicações previsivelmente interrompidas ao longo de longas distâncias devido aos pressupostos fundamentais construídos na arquitetura da Internet que se seguem:

- Um caminho *end-to-end* existe entre a fonte e o destino durante uma sessão de comunicação;
- Para uma comunicação realista, a retransmissão baseada no *feedback* dos recetores deve ser composta para reparação de erros;
- A perda de pacotes *end-to-end* é pequena;
- Todos os *routers* e estações terminais suportam protocolos TCP/IP;
- As aplicações não precisam de se preocupar com a *performance* da comunicação.

Numa rede tolerante a atrasos, os pressupostos da Internet são flexíveis. E os princípios da formulação de uma DTN são resumidos aos seguintes pontos:

- Mensagens com tamanho variável vão existir como abstração da comunicação para facilitar à rede a habilidade de tomar decisões em relação ao agendamento ou seleção de caminho;
- Uma sintaxe que suporte uma ampla gama de convenções de nomenclatura e endereçamento aplicada para aumentar a interoperabilidade;

- Armazenamento dentro da rede com o objetivo de suportar operações *store-and-forward* sobre múltiplos caminhos e potencialmente longos períodos de tempo;
- Mecanismos de segurança são necessários para proteger a infraestrutura de utilizadores não autorizados descartando o tráfego o mais rápido possível.

Estas redes estão a ser utilizadas para investigação na educação, telecomunicações, serviços governamentais, monitorização ambiental, comunicação veicular e espacial.

3.1.1. Características das DTNs

Existem alguns cenários de rede onde os atuais protocolos da Internet não funcionam bem, como as missões espaciais para Marte e testes de qualidade da água de um lago em áreas rurais. Ambos os cenários têm em comum dispositivos que incorporam computação e tecnologia de rede em ambientes de rede menos tradicionais. As redes de computadores nesses ambientes enfrentam novos desafios, novas técnicas e protocolos que são necessários. De entre os desafios podemos destacar os seguintes:

- Conectividade intermitente

Se não houver um caminho *end-to-*end entre a origem e o destino, a comunicação *end-to-end* usando os protocolos TCP/IP não funcionará, e por isso, serão necessários novos protocolos para suportar as comunicações sem um caminho *end-to-end*.

- Atraso longo ou variável

Além da conectividade intermitente, os longos atrasos de propagação entre nós e os atrasos de filas variáveis em cada nó, contribuem para atrasos de caminho *end-to-end* conseguindo assim maiores desempenhos em relação ao protocolos da Internet e aplicações que necessitam de retransmissão rápida de dados.

- Taxas de dados assimétricas

A Internet suporta assimetrias moderadas de taxa de dados bidirecionais para utilizadores com TV a cabo ou acesso DSL assimétrico, mas se as assimetrias forem grandes, os protocolos conversacionais não funcionarão.

- Taxas de erro elevadas

Erros de bit nas ligações de transmissão requerem correção ou retransmissão do pacote inteiro, o que pode resultar no aumento do tráfego da rede. Para uma dada taxa de erro de ligação, menos retransmissões são necessárias para *hop-by-hop* do que para uma retransmissão *end-to-end.*

3.2. Modelos de mobilidade

Os modelos de mobilidade permitem que os nós se movam de uma determinada maneira na simulação. Existem vários tipos de modelos de mobilidade em DTNs, mas apenas iremos abordar alguns deles, sendo estes, o *Shortest Path Movement*, o *Random Waypoint* e o *Map Based Movement Model*.

No modelo *Map Based Movement Model* é necessário predefenir um mapa para que os nós se movam segundo os caminhos do mapa. Este modelo inclui três modelos de mobilidade que são baseados no *Map Based Movement Model*, são eles o *random map based movement*, o *shortest path map based movement* e por último, o *route map based movement*.

O modelo *Shortest Path Movement* é a versão mais sofisticada do modelo *Map Based Movement Model*. Neste modelo, os nós utilizam o algoritmo de *Djisktra* para encontrar o caminho mais curto.

No modelo *Random Waitpoint* os nós movem-se aleatoriamente numa direcção arbitrária. Os nós começam a movimentar-se a partir de uma localização e seguem para qualquer direção deixando os caminhos do mapa de ser revelantes para os nós.

3.3. Protocolos de encaminhamento

Existem vários protocolos de encaminhamento DTN, nomeadamente, *Direct Delivery*, *First Contact, Spray and Wait, ProPhet, Max-Prop* e *Epidemic*.

O *Direct Delivery* e o *First Contact* são protocolos de apenas uma cópia, onde apenas uma cópia de cada mensagem existe na rede. Enquanto que no *Direct Delivery* um nó carrega a mensagem até encontrar o seu destino final, no *First Contact* os nós seguem as mensagens até que ao primeiro nó que encontram, o que resulta numa procura "*Random Walk*" até ao nó de destino.

O *Spray and Wait* é um protocolo de n cópias que limita o número de cópias de mensagens criadas que distribui as cópias das mensagens para os contactos até atingir o número máximo de cópias que pode fazer. Contudo, existem duas variantes do protocolo *Spray and Wait*, sendo elas, a variante normal e a variante binária. A variante

normal um nó dá uma cópia a um contacto, enquanto que na variante binária metade das cópias são encaminhadas.

O protocolo *ProPhet* estima qual o nó que tem maior probabilidade de entregar a mensagem até ao destino final baseando-se no registo de nós que encontrou.

O *Max-Prop* inunda a rede com as mensagens e só as apaga quando uma das cópias das mensagens for entregue ao destino. Envia ainda uma mensagem aos *hosts* com uma determinada ordem que tem em conta o número de *hops* de uma mensagem e as probabilidades das suas entregas consoante os encontros anteriores.

Por último, o protocolo *Epidemic* que replica as mensagens para todos os vizinhos.

4. Desenvolvimento do projeto

A secção de desenvolvimento do projeto está dividida por duas fases. A primeira fase correponde à fase inicial do trabalho em que tinhamos como objectivo conhecer o funcionamento do simulador *The One*, e a segunda fase corresponde à fase em que tinhamos de implementar o que fosse necessário para avaliar a solução proposta.

4.1. Primeira fase de trabalho

Na fase inicial do projeto laboratorial foi necessário conhecer o funcionamento do simulador, e por isso, começamos por exportar um mapa de pequena dimensão da cidade de Guimarães (Figura 1). Para tal, foi preciso utilizar a ferramenta *OpenStreetMap*.

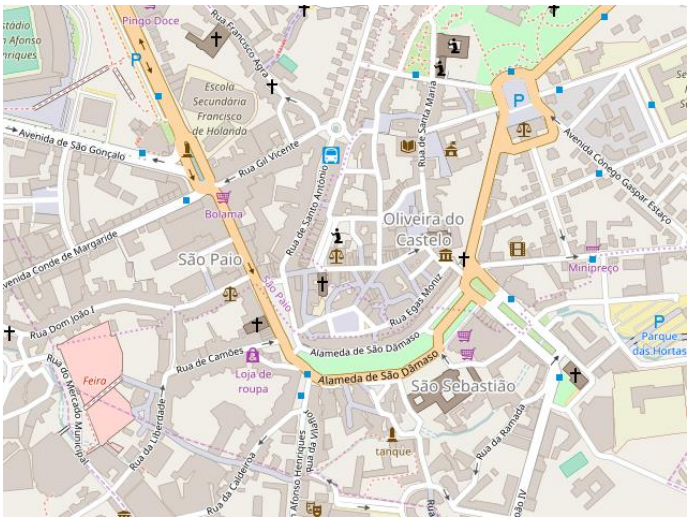

Figura 1- Mapa de pequena dimensão da zona de Guimarães.

Uma vez que era necessário limpar o mapa de forma a deixar apenas as estradas importantes, recorremos à ferramenta *JOSM* para conseguir obter o mapa apresentado de seguida (Figura 2).

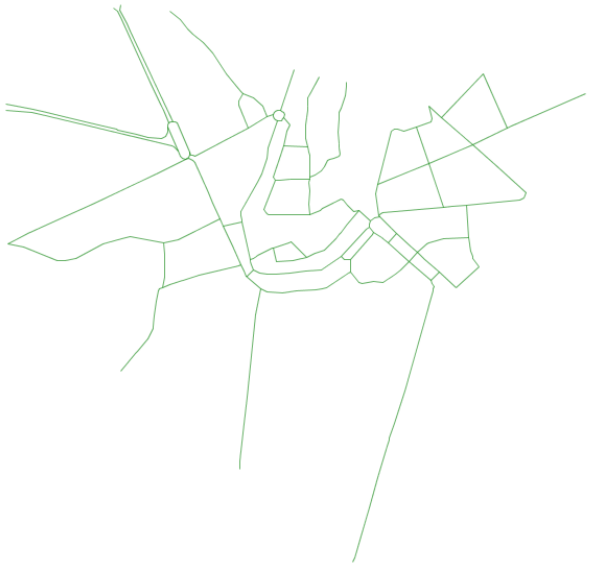

Figura 2- Mapa da pequena zona de Guimarães apenas com as estradas.

Contudo, para iniciar o trabalho no simulador *The One* era preciso ter o mapa em formato *.wkt* pois este estava em formato *.osm*, e por isso, foi necessário utilizar a ferramenta de conversão *osm2wkt* disponível através do *github*.

Tendo o necessário, importamos o ficheiro *.wkt* para o simulador, e começamos a definir três grupos no ficheiro de configuração, sendo um grupo para os *beacons,* outro para os portais e o último para os veículos. Seguindo os detalhes técnicos do projeto, é preciso definir o número mínimo de veículos como sendo de 50, o total de *beacons* como 5 e o número de portais consoante o que o grupo de trabalho achar necessário. Assim, no ficheiro de configuração encontra-se um grupo com o *groupID cars* e o número de *hosts* como 50, um grupo com o *groupID* portais em que o número de *hosts* nesta fase inicial será de 5 e um grupo com o *groupID beacons* em que o número de *hosts* é 5 (Figura 3).

```
Group2.groupID = beacon
Group2.movementModel = MapRouteMovement
Group2.routeFile= projeto/localBeacons.wkt
Group2.routeType = 2
Group2.nrofHosts = 5

Group3.groupID = portais
Group3.movementModel = MapRouteMovement
Group3.routeFile= projeto/localPortals.wkt
Group3.routeType = 2
Group3.nrofHosts = 5

Group4.groupID = cars
Group4.movementModel = ShortestPathMapBasedMovement
Group4.router=EpidemicRouter
Group4.routeType = 1
Group4.waitTime = 10, 20
Group4.speed = 4, 7
Group4.nrofHosts = 50
```

Figura 3-Configuração inicial dos beacons, portais e veículos no ficheiro de configuração.

Tanto os *beacons* como os portais são dispositivos fixos, mas os veículos são dispositivos que vão percorrer o mapa, por isso o modelo de mobilidade para estes será diferente dos primeiros. Os portais e os beacons foram definidos com o modelo de mobilidade *StationaryMovement,* enquanto que os veículos foram definidos com o modelo *MapRouteMovement*. Sendo os *beacons* e os portais dispositivos parados, é preciso definir a localização de cada um deles no mapa, e por isso, será criado um grupo para cada dispositivo fixo necessário. Desta forma, no ficheiro de configuração foram criados 5 grupos que representam cada um dos *beacons* (Figura 4) e 5 grupos que representam cada um dos portais (Figura 5).

```
Group2.groupID = beacon
Group2.movementModel = StationaryMovement
Group2.nodeLocation = 400,315
Group2.nrofHosts = 1

Group3.groupID = beacon
Group3.movementModel = StationaryMovement
Group3.nodeLocation = 844,300
Group3.nrofHosts = 1


Group4.groupID = beacon
Group4.movementModel = StationaryMovement
Group4.nodeLocation = 932,300
Group4.nrofHosts = 1


Group5.groupID = beacon
Group5.movementModel = StationaryMovement
Group5.nodeLocation = 1160,0
Group5.nrofHosts = 1


Group6.groupID = beacon
Group6.movementModel = StationaryMovement
Group6.nodeLocation = 1293,1013
Group6.nrofHosts = 1
```

Figura 4- Configuração dos beacons no ficheiro de configurção.

```
Group8.groupID = portal
Group8.nrofHosts = 1
Group8.movementModel = StationaryMovement
Group8.nodeLocation= 1118,2664
Group8.speed = 0, 0

Group9.groupID = portal
Group9.nrofHosts = 1
Group9.movementModel = StationaryMovement
Group9.nodeLocation= 1377,650
Group9.speed = 0, 0

Group10.groupID = portal
Group10.nrofHosts = 1
Group10.movementModel = StationaryMovement
Group10.nodeLocation= 2062,2885
Group10.speed = 0, 0

Group11.groupID = portal
Group11.nrofHosts = 1
Group11.movementModel = StationaryMovement
Group11.nodeLocation= 3501,679
Group11.speed = 0, 0

Group12.groupID = portal
Group12.nrofHosts = 1
Group12.movementModel = StationaryMovement
Group12.nodeLocation= 2719,2154
Group12.speed = 0, 0
```

Figura 5-Configuração dos portais do ficheiro de configuração.

Posto isto, através do comando *nodeLocation* no grupo de *beacons* foram definidas as coordenadas da localização consoante a estratégia inicial. Sendo esta uma primeira fase de trabalho para que possamos descobrir o funcionamento do simulador, definimos que a estratégia inicial de localização dos *beacons* seria de os colocar nas zonas onde houvesse mais tráfego no mapa da zona que selecionamos. De seguida, é possível visualizar a distribuição dos *beacons* no mapa utilizado (Figura 6).

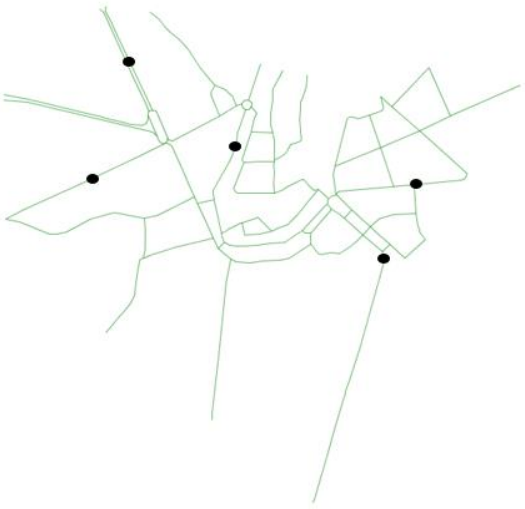

Figura 6- Localização dos beacons.

Apesar de já termos definido o modelo de mobilidade e os grupos, não foi definida uma estratégia de localização para os portais, e por isso, não foi definido nada no comando de *nodeLocation*.

Seguidamente, é preciso garantir que apenas os *beacons* geram mensagens e que apenas os portais as podem receber. No entanto, em primeiro lugar é preciso criar um evento, que representa uma mensagem, para cada um dos *beacons* e definir que de 5 em 5 segundos uma nova mensagem é gerada (Figura 7).

```
# Class of the first event generator
Events1.class = MessageEventGenerator
# (following settings are specific for the MessageEventGenerator class)
# Creation interval in seconds (one new message every 25 to 35 seconds)
Events1.interval = 500
# Message sizes (500kB - 1MB)
Events1.size = 500k,1M
# range of message source/destination addresses
Events1.hosts = 0,0
Events1.tohosts= 60,84
# Message ID prefix
Events1.prefix = M
```

Figura 7-Configuração dos eventos.

Sendo que através do comando *hosts* identifica-se o *beacon* que é responsável por gerar a mensagem, e através do comando *toHosts* definiu-se o ID dos portais aos quais as mensagens devem ser entregues.

Para além do que foi referido, nesta fase inicial não foi efectuado mais nada. Todo o trabalho desenvolvido de seguida, foi efectuado na segunda fase de trabalho, e por isso, será apresentado na secção seguinte.

4.2. Segunda fase de trabalho

Nesta secção, iremos apresentar tudo o que foi desenvolvido depois da fase inicial de trabalho.

4.2.1. Configuração dos dispositivos

Passada a fase em que percebemos o funcionamento do *The One*, deixámos o mapa de pequena dimensão da zona de Guimarães e voltámos a exportar com a mesma ferramenta um novo mapa, mas desta vez de uma zona maior (Figura 8) da cidade de Guimarães.

Figura 8-Mapa de maior dimensão da cidade de Guimarães.

Tal como no primeiro mapa, foi preciso limpar tudo o que não fosse estradas e convertê-lo para o formato *.wkt*. Como agora estamos a utilizar um novo mapa, foi preciso voltar a definir a localização dos dispositivos fixos e a configurar todos os dispositivos consoante o que foi estabelecido para esta fase.

Uma vez que já na primeira fase tinhamos definido o número de *beacons* que os detalhes do projeto laboratorial exigiam e o modelo de mobilidade como o *StationaryMovement* devido ao facto de serem dispositivos fixos, apenas foi necessário voltar a definir a sua localização, e para isso, foi seguida a estratégia da primeira fase em que estes dispositivos iam ser colocados nas zonas de maior tráfego do mapa (Figura 9).

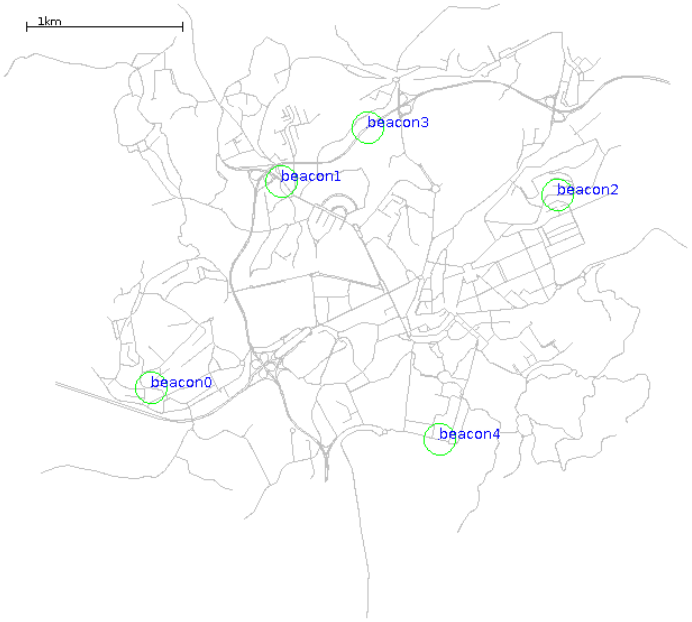

Figura 9-Localização dos beacons.

No ficheiro de configuração, a localização foi definida através do comando *nodeLocation* com as coordenadas do ponto pretendido em cada um dos grupos que tenham como *groupID beacon* (Figura 10).

```
Group1.groupID = beacon
Group1.nrofHosts = 1
Group1.bufferSize = 20k
Group1.movementModel = StationaryMovement
Group1.nodeLocation= 1478,2490
Group1.speed = 0, 0

Group2.groupID = beacon
Group2.nrofHosts = 1
Group2.bufferSize = 20k
Group2.movementModel = StationaryMovement
Group2.nodeLocation= 1769,1128
Group2.speed = 0, 0

Group3.groupID = beacon
Group3.nrofHosts = 1
Group3.bufferSize = 20k
Group3.movementModel = StationaryMovement
Group3.nodeLocation= 3523,1213
Group3.speed = 0, 0

Group4.groupID = beacon
Group4.nrofHosts = 1
Group4.bufferSize = 20k
Group4.movementModel = StationaryMovement
Group4.nodeLocation= 2313,788
Group4.speed = 0, 0

Group5.groupID = beacon
Group5.nrofHosts = 1
Group5.bufferSize = 20k
Group5.movementModel = StationaryMovement
Group5.nodeLocation= 2770,2776
Group5.speed = 0, 0
```

Figura 10-Configuração dos beacons.

Para definir os portais começamos por estabelecer que os iamos localizar nas estradas de maior convergência de forma a que possamos cobrir o maior número de estradas utilizando o menor número possível de portais (Figura 11). No entanto, à medida que fomos definindo os portais percebemos que era ainda necessário colocar em alguns pontos importantes que apenas abrangessem uma ou duas estradas.

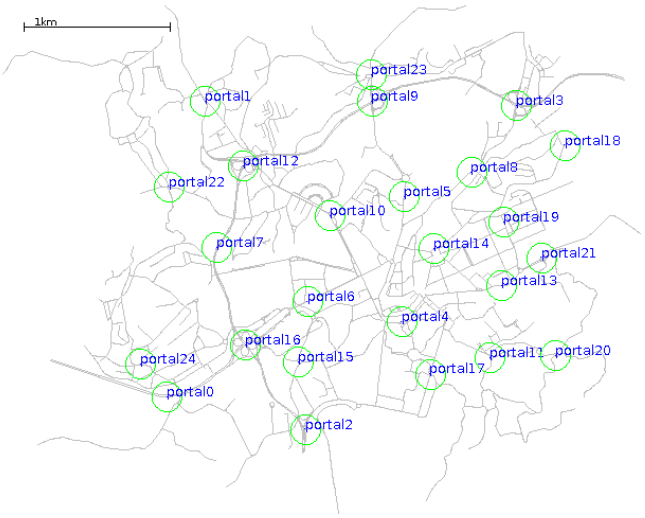

Figura 11-Localização dos portais.

No ficheiro de configuração, a localização dos portais também foi colocada através do comando *nodeLocation*. Através dos pontos que foram analisados, decidimos utilizar 24 portais, sendo que foi criado para cada um deles um grupo (Figura 12). Na figura seguinte apenas apresentados alguns dos portais.

```
Group6.groupID = portal
Group6.nrofHosts = 1
Group6.movementModel = StationaryMovement
Group6.nodeLocation= 1118,2664
Group6.speed = 0, 0

Group7.groupID = portal
Group7.nrofHosts = 1
Group7.movementModel = StationaryMovement
Group7.nodeLocation= 1377,650
Group7.speed = 0, 0

Group8.groupID = portal
Group8.nrofHosts = 1
Group8.movementModel = StationaryMovement
Group8.nodeLocation= 2062,2885
Group8.speed = 0, 0
```

Figura 12-Configuração dos portais.

Como o número mínimo de veículos a utilizar é de 50, especificamos que iriamos utilizar numa primeira fase de testes 55 veículos. Desta forma, foi criado um grupo com o número de *hosts* 55 e com o *groupID* de *cars*. Sendo os veículos dispositivos que se movem ao longo do mapa, definimos que alguns dos veículos iam ter como modelo de

mobilidade o *ShortestPathMapBasedMovement*, e outros teriam como modelo o *MapBasedMovement*. Nestes, serão definidas rotas, de forma a obrigar os veículos a percorrer os caminhos especificados. Serão explicados na próxima secção.

4.2.2. Rotas forçadas

Para testar um ambiente mais próximo da realidade adicionamos rotas forçadas para verificar o quanto estas podem influenciar o desempenho do sistema. Uma vez que os humanos são caracterizados por terem hábitos diários como ir e vir do trabalho, estas rotas procuram simular estes comportamentos nas estradas onde existe maior afluência. Assim também nos permite aumentar a densidade de nós nestas estradas o que permite também testar como retransmissão de mensagens se comporta com os diferentes protocolos de encaminhamento.

Estas rotas foram colocadas com base nas estradas que tem acesso á autoestrada e ao centro da cidade, já que são as vias com boas condições de acesso á cidade. Foram utilizados 20 nós neste processo com o objetivo de ver como se relacionam com os outros nós que não tem uma rota forçada. Assim podemos comparar se estes nós com trajetos pré-definidos tem ou não uma grande influência nos resultados obtidos.

Para adicionar estas rotas ao *theOne*, foi necessário criar um ficheiro *WKT* com as *LIneStrings* das estradas correspondentes e no grupo de nós pretendido colocamos o modelo de movimento como *MapRouteMovement* e indicamos a localização do ficheiro com as *LineStrings* (Figura 13).

```
Group33.groupID = anycast
Group33.movementModel = MapRouteMovement
Group33.router = SprayAndWaitRouter
Group33.routeType = 1
Group33.waitTime = 10, 30
Group33.speed = 50, 70
Group33.nrofHosts = 30
Group33.routeFile = data/guimaRoute.wkt
```

Com as as configurações apresentadas um exemplo da disposição é encontra-se representado na figura 14 , onde se pode verificar uma maior densidade de nós nos locais já referidos.

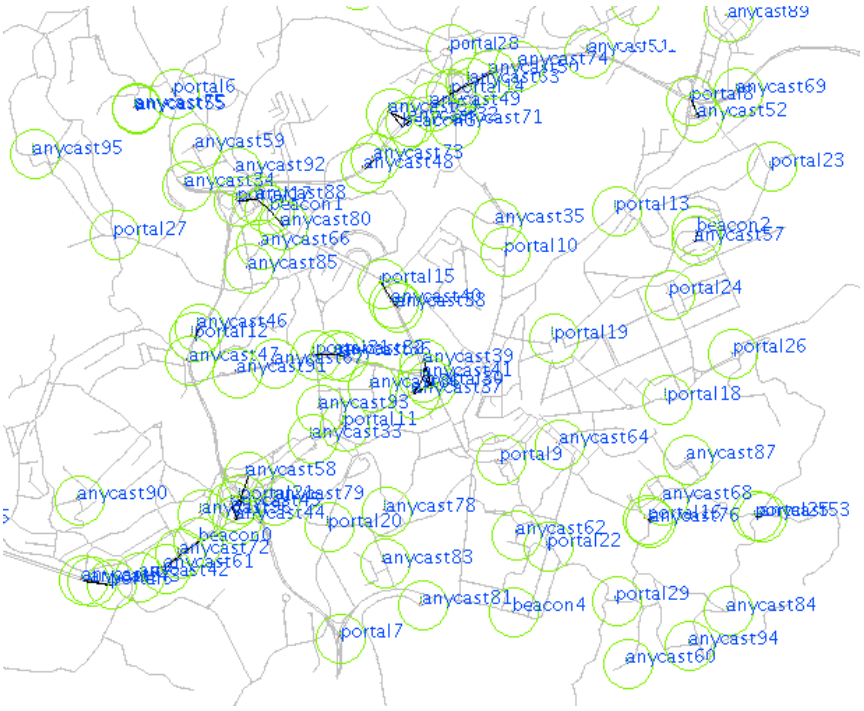

Figura 14 - Disposição de nós com rotas forçadas.

4.2.3. Interface de Comunicação

No estabelecimento de comunicações entre nós, *beacons* e portais foi usada a tecnologia Bluetooth. Esta tecnologia permite a comunicação sem fios de curto alcance entre dispositivos através de ondas rádio, criando redes ad-hoc conhecidas como *piconets*. Quando uma rede é estabelecida, um dispositivo toma a função de *master* enquanto os outros atuam como *slaves*. As *piconets* são estabelecidas dinamicamente e automaticamente à medida que os dispositivos com Bluetooth entram ou saem da área de proximidade rádio. Esta é uma tecnologia de baixo custo uma vez que requer *hardware* mais barato e consome menos energia que o *Wi-Fi (IEEE 802.11).*

A interface de *Bluetooth* ficou configurada com um alcance de 100 m e uma velocidade de transmissão de 2500 kBps (Figura 15). Com este alcance pretende-se que haja uma maior interação principalmente entre nós e entre nós e portais já que depois de um nó passar num *beacon* é importante que a mensagem chegue a um portal. Como as mensagens geradas pelos *beacons* são pequenas a velocidade de transmissão também pode ser pequena otimizando assim a transferência de mensagens.

```
## Interface-specific settings:
# type : which interface class the interface belongs to
# For different types, the sub-parameters are interface-specific
# For SimpleBroadcastInterface, the parameters are:
# transmitSpeed : transmit speed of the interface (bytes per second)
# transmitRange : range of the interface (meters)

# "Bluetooth" interface for all nodes
btInterface.type = SimpleBroadcastInterface
# Transmit speed of 2 Mbps = 250kBps
btInterface.transmitSpeed = 2500k
btInterface.transmitRange = 100
```

Figura 15 - Exemplo de configuração da interface Bluetooth.

4.2.4. Configurações das mensagens

As mensagens são unicamente geradas pelos *beacons* e contêm o instante de tempo (data e hora) e a identificação do local onde foi gerada. Uma mensagem diferente é gerada a cada 5s com um tamanho de 20 *kbytes*. O tamanho das mensagens foi assim definido porque o tempo para transmitir uma mensagem de um *beacon* para um nó é curto e assim uma maior probabilidade da mensagem ser totalmente enviada. Estas mensagens tem como destino final qualquer portal, cada com o prefixo M e tempo de geração ocorre no intervalo de tempo definido em *EventsX.time* (Figura 16).

```
# Class of the first event generator
Events1.class = MessageEventGenerator
# (following settings are specific for the MessageEventGenerator class)
# Creation interval in seconds (one new message every 25 to 35 seconds)
Events1.interval = 5,5
# Message sizes (500kB - 1MB)
Events1.size = 20k,20k
# range of message source/destination addresses
Events1.hosts = 0,0
Events1.tohosts= 5,29
# Message ID prefix
Events1.prefix = MA
 Events1.toGroup= PORTAL
Events1.time = 0 , 15000
```

Figura 16 - exemplo de configuração de geração de mensagens.

4.2.5. Configuração do multicast

De modo a melhorar o ambiente de simulação para ser o mais realista possível, de acordo com o tipo de aplicação que pretendemos desenvolver, foi necessário alterar o código fonte do simulador *The* One.

A alteração do código tem como objetivo adicionar uma funcionalidade ao *The One*, de forma a permitir que uma mensagem não seja entregue apenas a um portal com um ID específico, mas sim a um conjunto de portais definidos através de um ID de grupo. O esquema abaixo (Figura 17) demonstra como será tratada a receção de uma mensagem num nó da rede.

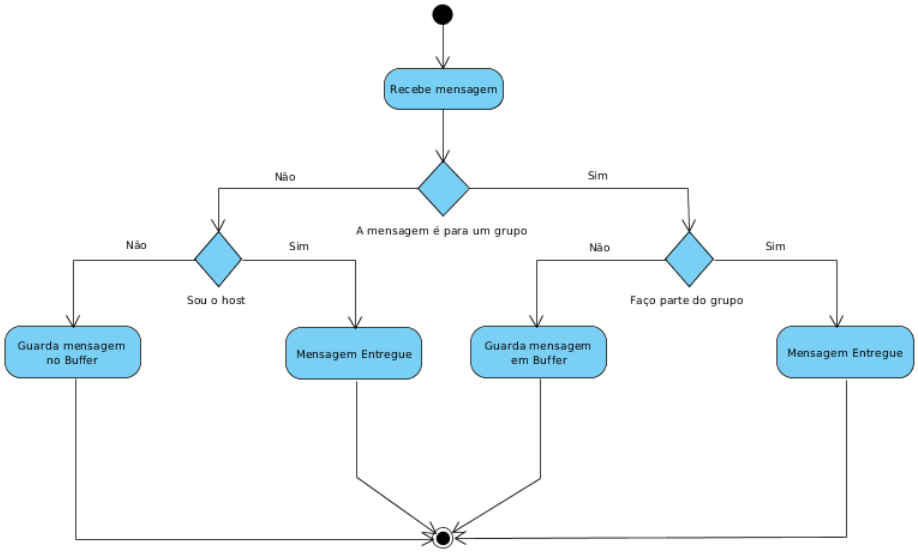

Figura 17-Fluxograma da recepção de uma mensagem.

Numa primeira fase adicionamos uma nova opção aos grupos e outra aos eventos no ficheiro de configuração. Para os grupos adicionamos uma configuração onde definimos o grupo a que pertencem através do comando *Group.eventGroup= <nome do grupo>* (Figura 18). Com este comando definimos o grupo a que este nó pertence , de forma a que ele possa receber mensagens que sejam para ele ou para o grupo.

```
Group22.groupID = portal
Group22.nrofHosts = 1
Group22.movementModel = StationaryMovement
Group22.nodeLocation= 1653,2313
Group22.speed = 0, 0
Group22.eventGroup = Portal
```

Figura 18-Comando Group.eventGroup.

Através do comando *Event.toGroup = <nome do grupo>* (Figura 19) específicamos o grupo para o qual as mensagens serão enviadas quando o gerador de eventos as criar.

```
Events2.class = MessageEventGenerator
Events2.interval = 5,5
Events2.size = 20k,20k
Events2.hosts = 1,1
Events2.tohosts= 5,29
Events2.prefix = MB
Events2.toGroup= Portal
Events2.time = 0 , 15000
```

Figura 19-Comando Event.toGroup.

Para ser possível ao *The One* ler estas novas configurações foram alteradas as classes "*SimScenario*" e "*MessageEventGenerator*". A classe "*SimScenario*" é responsável por ler e fazer as inicializações das configurações dos grupos, e de alguns parâmetros da simulação que estão definidos no ficheiro de configuração. Desta forma, adicionamos nesta leitura o parâmetro "*eventGroup*" (Figura 20), que toma o valor de *Default* quando não está definido no ficheiro de configuração.

```
if(s.contains(EVENT_GROUP)){
    eventGroup= s.getSetting(EVENT_GROUP);
}else{
    eventGroup = "Default";
}
```

Figura 20-Parâmetro "Event_Group".

A classe *MessageEventGenerator* é onde é efectuada as leituras das configurações dos eventos. Deste modo, foi efectuada a leitura do parâmetro "*toGroup*"(Figura 21), que toma o valor de *Default* quando não está definido no ficheiro de configuração.

```
if(s.contains(TO_GROUP)){
    this.toGroup=s.getSetting(TO_GROUP);
}else{
    this.toGroup="Default";
}
```

Figura 21-Parâmetro "To_Group".

Após termos acesso às variáveis que foram definidas no ficheiro é necessário atribuí-las aos nós da rede e ao gerador de eventos. No caso dos nós, estes são representados pelo objecto *DTNHost*. Assim, foi adicionado a este objecto a variável "eventGroup", que é o grupo a que o nó pretende, é ainda criado o método *getEventGroup()* para ser possível ter acesso ao valor desta variável. No *SimScenario* são inicializados os nós da rede, portanto é associado o "*eventGroup*" específicado no ficheiro de configuração aos *DTNHost*.

No caso das mensagens teve que ser adicionado à classe *Message* uma variável "*toGroup*", de forma a que cada mensagem contenha o identificador do grupo para onde estou a ser enviadas. A classe *responsável* pela geração de eventos (*MessageCreateEvent*) , recebe como parâmetro o "*toHost*" para conseguir gerar as mensagens com o grupo especificado no ficheiro de configuração.

Após as mensagens e o nós terem todas as variáveis necessárias para podermos distinguir se um nó pertence a um determinado grupo, e a que grupo é destinada uma determinada mensagem, é necessário garantir que podemos encaminhar para um determinado grupo ou para um determinado nó, ou ambos em simultâneo. Posto isto, alterámos a classe responsável pela gestão do encaminhamento das mensagens

(*MessageRouter*), de modo a entregar uma mensagem a um determinado grupo caso no ficheiro de configuração seja definido um grupo destino das mensagens. Se não se encontrar definido o grupo, será então feita a verificação pelo nós.

Para tal, em cada mensagem (*aMessage*) (Figura 22) é verificado o grupo a que esta pertence. Se pertencer ao grupo "*Default*" (valor caso a variável não tenha sido definida no ficheiro de configuração), então só é destino se o nó for o próprio definido na mensagem. Caso contrário, é feita a verificação através da conparação do grupo destino da mensagem e do grupo a que o nó pertence.

```
if(!aMessage.getToGroup().equals("Default")){
    isFinalRecipient = aMessage.getToGroup().equals(host.getEventGroup());

}else{
    isFinalRecipient = aMessage.getTo() == this.host;
}
```

Figura 22-Condição para verificar o destinatário.

5. Resultados experimentais

Nesta secção, iremos apresentar os testes efectuados com diferentes protocolos de encaminhamento e com diferentes valores de TTL, entre outros.

Como protocolos foram utilizados o protocolo de encaminhamento *Epidemic* e o *Spray And Wait* este com duas versões: Vanilla e Binary com 32 cópias. Das outras variáveis, o TTL variou entre 40 e 110 minutos, o número de nós entre 65 e 130 e a velocidade de cada nó entre 30 e 70. Todos cenários simulavam uma duração de 33 minutos com o *buffer* dos nós fixo em 5MB.

Dos testes realizados os melhores resultados foram obtidos com os protocolos *Epidemic* (Figura 24) e *Spray And Wait* versão *Binary* (Figura 23) ambos com TTL de 60 minutos, 100 nós e velocidade dos nós entre 30 e 55.

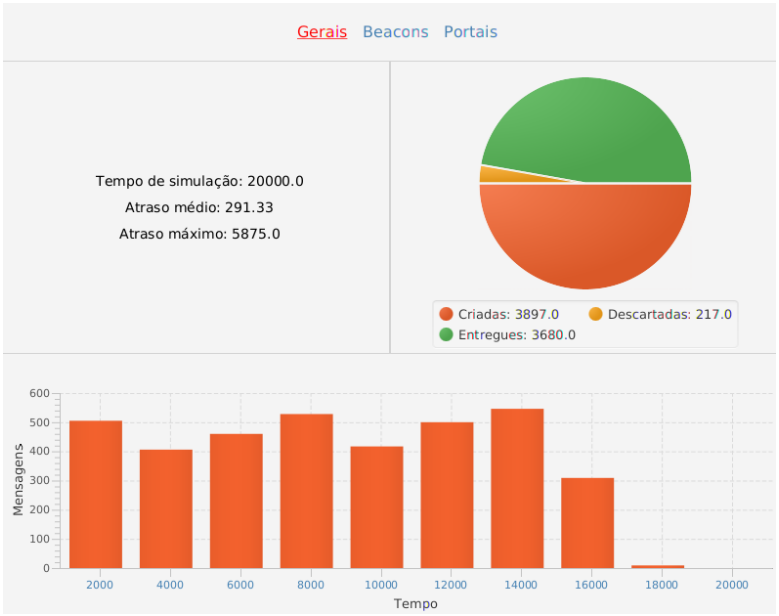

Figura 23-Resultado com o Spray and Wait modo binário (Estatísticas Gerais).

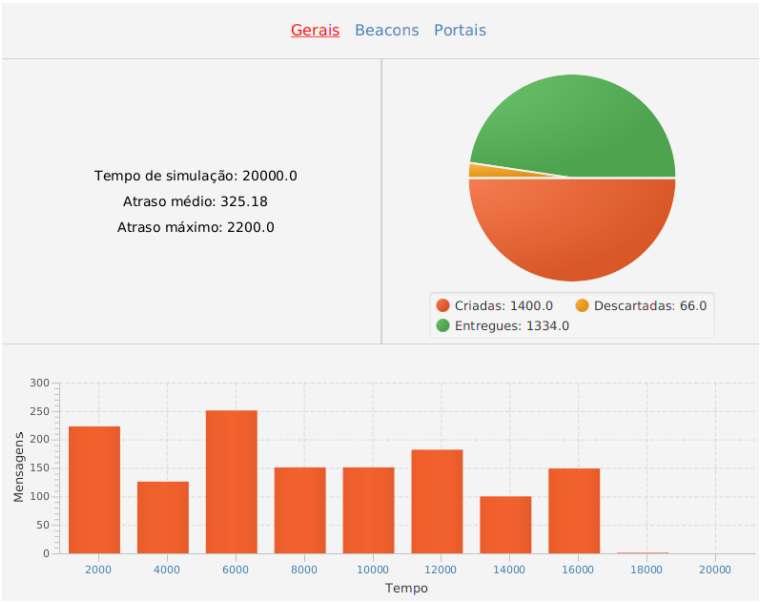

Figura 24-Resultado com o Epidemic.

De acordo com os resultados obtidos a taxa de entrega é aproximadamente de 95% com *Binary Spray And Wait* e aproximadamente 93.3% com o *Epidemic*, o que nos diz que para os parâmetros escolhidos estes protocolos apresentam um excelente desempenho com quase todas as mensagens que foram criadas a serem entregues a um portal. Já com o *Vanilla Spray And Wait* os resultados apresentavam uma grande discrepância em relação ao *Binary* com um enorme número de mensagens criadas e descartadas sendo que a melhor percentagem de entrega obtida foi de apenas 35% o que não é uma solução viável.

6. Ferramentas utilizadas

Nesta secção iremos abordar algumas das ferramentas utilizadas ao longo do desenvolvimento do trabalho, como o simulador *The One*, o *JOSM* e o *Open Street Map*.

- *The One*

O simulador *The One* é uma ferramenta de *software open-source* que foi desenvolvida com o intuito de possibilitar a análise do encaminhamento DTN.

- *JOSM*

A ferramenta *JOSM* permite limpar o mapa de forma a que possamos retirar todos os elementos não desejados, como edifícios, zonas verdes, entre outros.

- *Open Street Map*

É uma ferramenta de *software* que permite exportar os mapas de uma determinada zona com o formato *osm*.

7. Conclusão

Neste projeto encontramos o conceito das redes oportunistas, ao qual o grupo teve que se basear para simular um ambiente de monitorização de tráfego rodoviário. Para tal foi utilizado o *software* The One com qual nunca tivemos contacto.

Primeiramente, foi necessário conhecer este software já que não havia grande documentação disponível, o que levou algum tempo uma vez que não tinha implementado encaminhamento *multicast* e que foi necessário implementar já que no contexto do nosso projeto melhorava o desempenho do sistema. Além disto, foi útil estudar os vários protocolos existentes e testar quais o que de acordo com a sua teoria iriam adaptar melhor ao nosso problema.

Seguidamente, a disposição dos *beacons* e dos portais foram um grande desafio já que a localização destes tem uma grande influência no desempenho da rede. O número limite de *beacons* dificultou a escolha da sua disposição uma vez que tinham de ser colocados em locais estratégicos para que apanhem o máximo número de nós possíveis. Já os portais apesar de haver limite no número a escolher havia o problema de alguns nunca receberem mensagens de um nó, o que levou a fazer uma escolha moderada e que garantisse a cobertura necessária da cidade para receber todas as mensagens geradas pelos *beacons*.

Por fim, conseguimos chegar a um modelo que conseguisse obter uma taxa de entrega de mensagens viável e de acordo com os requisitos propostos. Com este projeto o grupo obteve uma melhor perceção da realidade da construção de uma rede oportunista e quais as dificuldades, tecnologia e o impacto que no futuro estas podem ter na área das comunicações.

8. Referências

[1] *The One Tutorial*, Disponível em: http://delay-tolerant-networks.blogspot.pt/p/one-tutorial.html, Consultado em: Maio de 2017.

[2] *The One Tutorial*, Disponível em: http://www.margalho.pro.br/subsites/theone.html, Consultado em: Maio de 2017.

[3] *The One* Simulator for DTN Protocol Evaluation, Disponível em: https://www.netlab.tkk.fi/~jo/papers/2009-03-simutools-one.pdf, Consultado em: Maio de 2017.