Redes Móveis



Projeto Laboratorial

Difusão de Avisos de Acidentes Rodoviários em Redes Veiculares

Mestrado Integrado em Engenharia de Telecomunicações e Informática

4º Ano

5 de maio de 2018

Constituição do Grupo



Carlos Carvalho, a75401, [carlosc.96@gmail.com](mailto:carlosc.96@gmail.com)



Nuno Tavares, a73985, [nuno.f.o.tavares@gmail.com](mailto:nuno.f.o.tavares@gmail.com)

Sérgio Fernandes, a76374, [sergio\_mfernandes@hotmail.com](mailto:sergio_mfernandes@hotmail.com)

Conteúdo

[Contexto 4](#_Toc515826497)

[Objetivo 4](#_Toc515826498)

[Criação do Mapa da Área Simulada 4](#_Toc515826499)

[Configuração dos Hosts 5](#_Toc515826500)

[Colocação dos Beacons 6](#_Toc515826501)

[Configuração dos veículos 7](#_Toc515826502)

[Configuração da interface Bluetooth 8](#_Toc515826503)

[Eventos 9](#_Toc515826504)

[Reports 10](#_Toc515826505)

[MessageStatsReport 11](#_Toc515826506)

[DeliveredMessagesReport 12](#_Toc515826507)

[MessageDelayReport 13](#_Toc515826508)

[MessagesReportRM 14](#_Toc515826509)

[Programa Java 15](#_Toc515826510)

[Resultados obtidos 16](#_Toc515826511)

[Análise critica 18](#_Toc515826512)

[Conclusão 19](#_Toc515826513)

Contexto

No futuro, os veículos irão comunicar entre si, como tal estes passarão a estar munidos de interfaces de rede do tipo *Wireless Access Vehicular Environment* (WAVE) e serão tratados como nós de redes veiculares. Esta comunicação entre veículos permite prevenir acidentes rodoviários, simplesmente utilizando este tipo de redes para difundir alarmes caso ocorram acidentes, alertando os outros carros ligados à rede.

Este projeto é realizado no âmbito da Unidade Curricular de Redes Móveis do Mestrado Integrado de Engenharia de Telecomunicações e Informática. Para efeitos de testes e implementação de algo em concreto foi utilizado o simulador *open source* “The ONE” para simular redes oportunistas, em particular, redes veiculares.

# Objetivo

O objetivo é avaliar a viabilidade das redes veiculares para distribuírem informação sobre acidentes rodoviários. Para isso é necessário medir o tempo que uma mensagem que anuncia a ocorrência de um acidente demora a chegar até aos outros veículos.

# Criação do Mapa da Área Simulada

O grupo utilizou a ferramenta OpenStreetMap para recolher um ficheiro com a extensão “.osm” de um mapa da área da Rotunda da Boavista (Porto, Portugal).

De seguida foi utilizada a ferramenta “osm2wkt” para converter o ficheiro “. osm” para um ficheiro com extensão “. wkt”, por ser este o formato suportado pelo “The ONE”. Para tal foram utilizados o seguinte comando:

**java -jar ./osm2wkt.jar rotunda.osm -o rotunda.wkt**

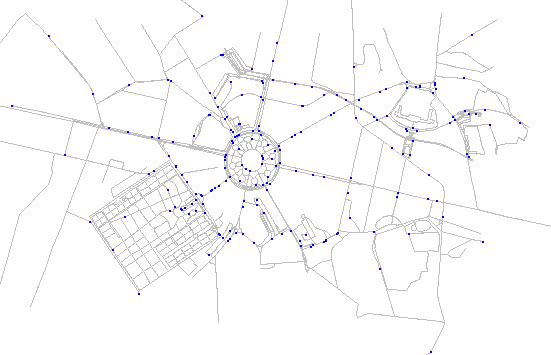


Figura 1 - Mapa simulado da Rotunda da Boavista no "The ONE"

Configuração dos Hosts

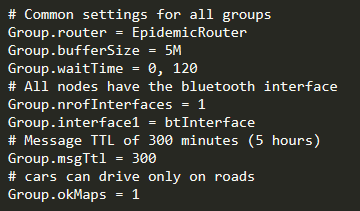


Figura 2 - Configuração geral dos hosts

Como podemos observar na figura para a configuração geral dos hosts foi necessário ter em conta os seguintes parâmetros:

* Encaminhamento *(router):*
  + O modelo escolhido para encaminhamento foi “*EpidemicRouter”*. Este modelo foi o escolhido porque foi o que obteve melhor resultado nos vários testes que o grupo realizou aos restantes modelos. Estes resultados serão apresentados *a posteriori.*
* Tamanho do buffer *(bufferSize):*
  + O valor atribuído é de *“5M”* que representa 5 Megabytes visto que este valor é aceitável para uma memória real.
* Tempo de espera *(waitTime):*
  + O valor atribuído a este parâmetro foi *“0,120”* para simular o tempo de paragem de um veiculo numa situação real.
* Número de interfaces *(nrofInterfaces):*
  + A cada host foi atribuída apenas uma interface.
* Interface *(interface1)*:
  + A interface atribuída foi do tipo “*btInterface”* que será especificada *a posteriori.*
* Tempo de vida da mensagem *(msgTtl)*;

Colocação dos Beacons

De acordo com os requisitos do projeto, para simular a ocorrência de acidentes foram colocados 10 nós fixos (*beacons*) que transmitem uma mensagem a cada 10 minutos.

O posicionamento destes *beacons* teve em conta que a existência de acidentes rodovários tem mais probabilidade de acontecer nos acessos à rotunda e nos cruzamentos. Assim sendo, foram posicionados 5 *beacons* na Rotunda da Boavista e os restantes 5 nos cruzamentos envolventes, como se pode verificar na Figura 2.

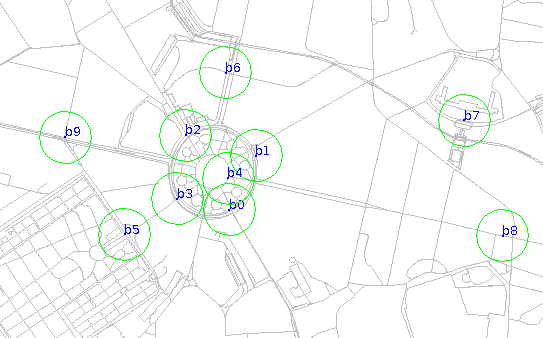


Figura 3 - Posicionamento dos Beacons no simulador "The ONE"

O grupo decidiu para cada beacon definir um grupo específico. Em baixo podemos observar as configurações de um desses grupos, onde é definido um movimento estacionário e a sua localização geográfica. Os beacons foram colocados nas saídas da rotunda por serem os sítios onde existem mais incidentes.

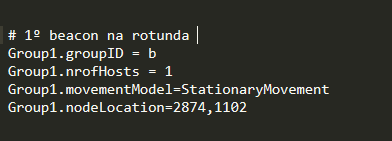


Figura 4 - Exemplo de configuração dos Beacons

Como podemos observar na figura apresentada acima para cada beacon tiverem que ser configuradas as seguintes variáveis:

* Id do grupo (*groupID*):
  + O valor atribuído a este campo é de “*beacon”* que simboliza o papel deste elemento na rede;
* Número de *hosts* (*nrofhosts*):
  + O valor atribuído a este campo é de “1”, visto que para cada localização existe apenas 1 beacon;
* Modelo de movimento (*movementModel*):
  + O modelo escolhido foi “*StationaryMovement”* pois representa a ausência de movimento dos beacons;
* Localização (*nodeLocation*):
  + O valor atribuído a este campo está de acordo com a localização estratégica de cada beacon:

Como já foi referido anteriormente, foi necessário definir um grupo para cada beacon o que faz com que os mesmos estejam definidos entre o grupo 1 e o grupo 10. O único parâmetro que varia de grupo para grupo é a sua localização visto esta ser diferente para cada beacon.

# Configuração dos veículos

Os veículos têm como função transmitir entre si as mensagens geradas pelos beacons e propagá-las na rede de modo a atingir o máximo de veículos possível. Para isso, e como requisito do projeto, foram colocados 200 veículos e configurados da seguinte forma:

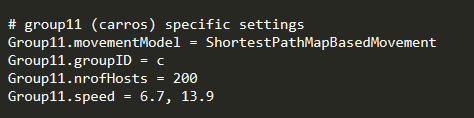


Figura 5 - Configuração dos veículos

Para a configuração dos veículos foi necessário a definição de certos parâmetros dos quais:

* Número de veículos *(nrofHosts);*
  + Este campo, como o nome indica, é o número de elementos de Host deste grupo. No caso dos carros, este valor é de 200.
* Id do grupo *(groupID)*:
  + O id atribuído a este grupo é “c” que representa a inicial de carros.
* Velocidade (*speed):*
  + A velocidade a que este grupo se vai movimentar está entre 6,7 e 13,9 m/s (25-50 Km/h). Estes valores foram escolhidos tendo por base as velocidades médias dentro de uma localidade e também nas rotundas.
* Modelo de movimento *(movementModel)*:
  + O modelo escolhido foi *“ShortestPathMapBasedMovement”*. Este modelo consiste na escolha aleatória de um ponto destino traçando a rota mais curta para o mesmo. A escolha deste modelo baseou-se no facto de o ponto de destino ser aleatório para simular o comportamento real de um condutor, e por ser baseado no mapa utilizado.

Configuração da interface Bluetooth

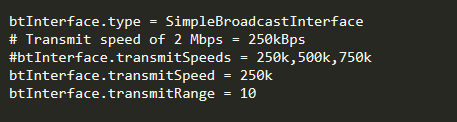


Figura 6 - Configuração da Interface Bluetooth

Para a configuração da interface de Bluetooth que cada veículo possuía foi necessário a definição de certos parâmetros dos quais:

* Tipo (*btInterface.type):*
  + O tipo de interface escolhido foi *“SimpleBroadCastInterface.* Este tipo foi o escolhido visto haver a necessidade de chegar a todos os veículos que se encontrem na rede, por essa mesma razão este tipo de interface satisfaz todos os requisitos do sistema.
* Velocidade de Transmissão *(transmitSpeed):*
  + A velocidade eescolhida foi de *“250k”* que tem como valor real 250kBps o que equivale a 2Mbps (MegaBytes por segundo). Esta valor foi escolhido visto ser o valor que mais se assemelha ao valor real que uma interface Bluetooth é capaz de obter como velocidade de transmissão.
* Alcance *(transmitRange)*:
  + O valor escolhido para o alcance que esta interface deve ter foi de *“10”* visto ser o necessário para cumprir os requisitos do nosso sistema.

Eventos

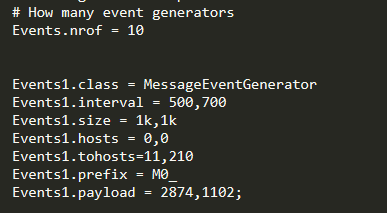


Figura 7 - Configuração dos Eventos

Para a configuração dos eventos foi necessário definir um evento para cada *beacon*, e para isso foi necessário a definição de vários parâmetros, dos quais:

* Número de eventos *(nrof):*
  + O valor escolhido foi de *“10”* porque necessitávamos de um evento por cada *beacon*. Visto terem sido colocados 10 *beacons,* era mandatário ter 10 eventos;
* Classe *(class)*:
  + O valor atribuído foi de *“MessageEventGenerator”* visto que era requisito do projeto a geração de mensagens aquando da ocorrência de acidentes.
* Intervalo *(interval)*:
  + O valor escolhido foi de *“500,700”,* este parâmetro representa o intervalo de criação para as mensagens, e é definido em segundos. Este parâmetro faz com que cada beacon gere um evento com uma frequência variável aleatória, visto que, apesar de nos requisitos ser pedido a criação a cada 10 minutos, o grupo achou que este valor se aproximava melhor à realidade, pois os acidentes não acontecem sempre com intervalo fixo.
* Tamanho *(size)*:
  + O valor escolhido foi de “*1k”* e representa o tamanho máximo das mensagens.
* Hosts:
  + Este campo associa um evento aos host que o geram. Nomeadamente, o “Events1” vai ser gerado pelo beacon cujo número é 0”
* Destinos (*toHosts)*:
  + O intervalo escolhido foi de “*11,210”* visto que os carros apresentam-se com esta gama de números. Assim, o beacon ao criar a mensagem associa como destinatário um determinado carro, e a rede faz o flooding desse pacote na rede.
* Prefixo *(prefix)*:
  + Prefixo associado a cada mensagem gerada por cada evento para distinção entre os mesmos. Este valor associa cada mensagem ao seu beacon, sendo que as mensagens com o id M0\_X são associadas ao beacon 0.
* *Payload:*
  + Este valor foi atribuído consoante as coordenadas do *beacon* ao qual o evento estava atribuído. Contudo, pode ser customizado com a mensagem que se pretender difundir na rede.

Reports

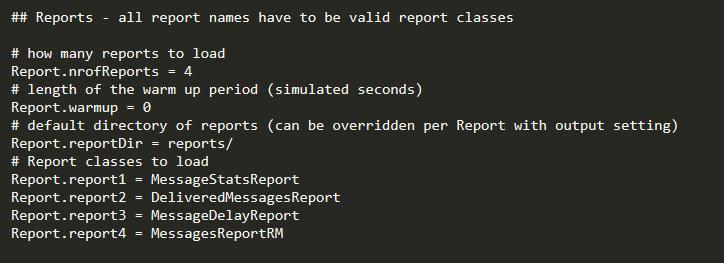


Figura 8 - Configuração dos relatórios

De maneira a obter resultados conclusivos e que representassem de forma real o desempenho da rede, decidimos tirar partido de uma ferramenta que nos é disponibilizada pelo software que são os Relatórios (Reports), para a configuração dos mesmos foi necessário a definição de vários parâmetros, dos quais:

* *Número de relatórios (*nrOfReports)*:*
  + O valor escolhido foi “4” visto ser o número de relatórios que decidimos configurar.
* Aquecimento (warmup):
  + O valor escolhido foi “0” visto que queriamos obter o máximo de informação possivel sobre o nosso Sistema, mal o Sistema “arrancasse” queriamos ter a informação disponivel.
* Diretoria (reportDir):
  + Local onde os relatórios são guardados.
* Relatório (reportX):
  + Este parâmetro foi definido consoante os relatórios que achamos pertinentes para retirar conclusões acerca do desempenho do Sistema. Os relatórios escolhidos foram:
    - MessageStatsReport;
    - DeliveredMessagesReport;
    - MessageDelayReport;
    - MessagesReportRM.

MessageStatsReport

Este relatório foi utilizado para conseguirmos obter dados estatísticos em relação à troca de mensagens na rede.

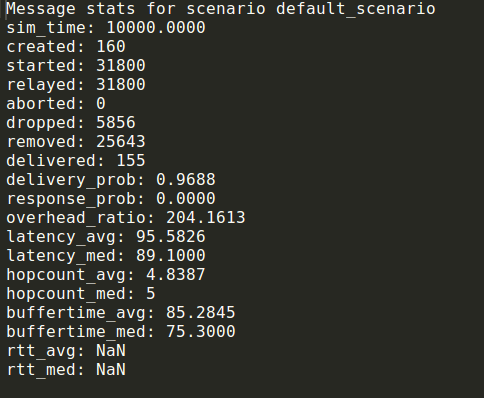


Figura 9 - Relatório de estatistica das mensagens

Como podemos observar na figura acima, com este relatório conseguimos obter informação muito importante para que seja possível observar o que se passa realmente na rede, esta informação é fundamental para a classificação de desempenho da mesma, e podemos salientar os seguintes dados mais relevantes:

* Tempo de simulação;
* Número de mensagens criadas;
* Número de mensagens perdidas;
* Número de mensagens entregues;
* Latência;
* Tempo em buffer.

DeliveredMessagesReport

Este relatório foi utilizado de maneira a que conseguíssemos verificar que cada mensagem gerada pelo respetivo *beacon* era entregue e reconhecida por todos os veículos presentes na nossa rede. Sem duvida um dos relatórios fundamentais para conseguir atingir os objetivos propostos ao longo da realização do projeto.

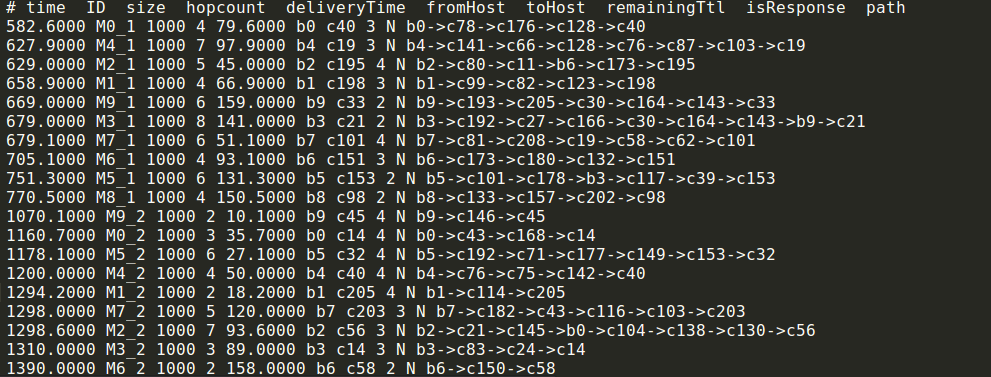


Figura 10 - Relatório de mensagens entregues

Como podemos observar na figura acima, com este relatório conseguimos verificar que cada mensagem chega a todos os veículos da rede, e para isso fornece-nos a seguinte informação:

* Tempo que a mensagem foi gerada;
* ID da mensagem;
* Tamanho da mensagem;
* Contagem de saltos;
* Tempo de entrega;
* Fonte;
* Destino;
* TTL restante;
* É resposta?
* Caminho percorrido.

MessageDelayReport

Este relatório foi utilizado para verificar o atraso que estava associado à entrega de cada mensagem, não é dos relatórios fundamentais, mas é um acréscimo de informação que complementa as estatísticas apresentadas pelos outros.

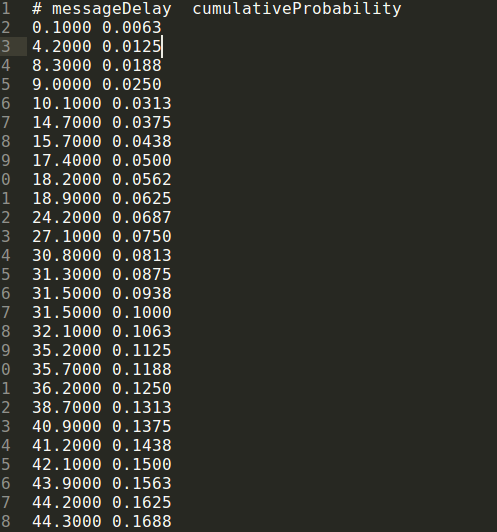


Figura 11 - Relatório do atraso associado à entrega das mensagens

Como podemos observar na figura acima, com este relatório conseguimos observar o atraso associado à entrega de cada mensagem e fornece-nos a seguinte informação:

* Atraso;
* Probabilidade cumulativa.

MessagesReportRM

Este relatório foi utilizado para verificar o estado de cada mensagem, não é dos relatórios fundamentais, mas é um acréscimo de informação que complementa a informação disponibilizada pelos outros.

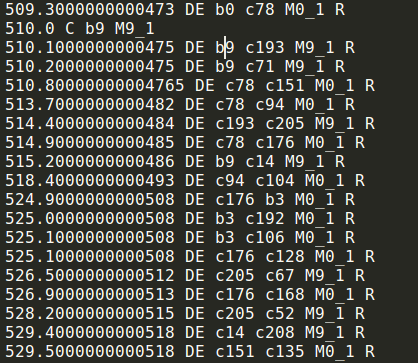


Figura 12 - Relatório que demonstra o estado de cada mensagem

Como podemos observar na figura acima, este relatório dá nos a seguinte informação:

* Marca temporal da mensagem;
* Nós pelos quais a mensagem passou;
* Se foi entregue / reencaminhada.

Programa Java

Para que fosse possível interpretarmos de uma maneira mais simplificada e coerente os resultados provenientes de todos os relatórios gerados pelo nosso sistema decidimos criar um programa Java.

O programa tem como finalidade apresentar todos os dados estatísticos que são pertinentes e requisitos do projeto, assim como, apresentar alguma informação adicional que foi relevante para a análise de todos os resultados obtidos dos diferentes protocolos de encaminhamento testados afim de selecionar o melhor para o nosso sistema.

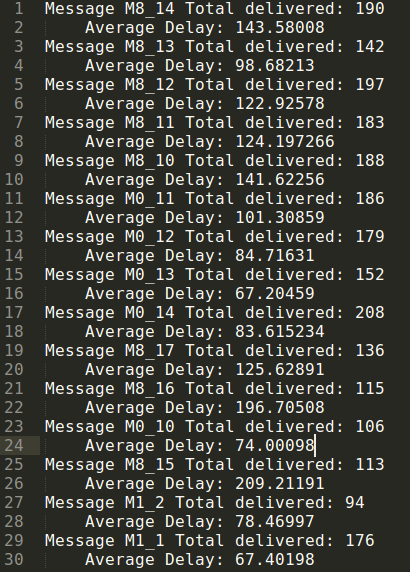


Figura 13 - Output programa - I

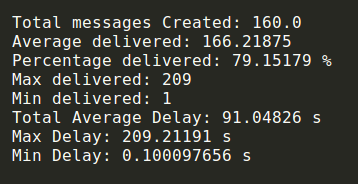


Figura 14 - Output programa - II

Como podemos observar nas figuras apresentadas acima, o output do nosso programa consiste na geração de um ficheiro que apresenta todos os dados estatísticos relevantes para o nosso sistema, e é constituído por:

* A respetiva mensagem e quantas da mesma foram entregues, assim como o atraso médio associado a esta entrega;
* Total de mensagens criadas;
* Média de mensagens entregues;
* Percentagem de entrega;
* Máximo de mensagens entregues;
* Mínimo de mensagens entregues;
* Atraso médio total na entrega das mensagens;
* Atraso máximo na entrega das mensagens;
* Atraso mínimo na entrega das mensagens.

Resultados obtidos

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Protocolo Router* | *TTL* | *Alcance* | *Mensagens criadas* | *Média de mensagens entregues* | *Percentagem de mensagens entregues* | *Máximo de mensagens entregues* | *Mínimo de mensagens entregues* | *Atraso médio total* | *Atraso máximo* |
| Direct  Delivery | 5 | 10m | 160 | 0.3125 | 0.14880952% | 1 | 0 | 27.483755s | 253.1001 s |
| First  Contact | 5 | 10m | 160 | 56.51875 | 26.91369% | 137 | 1 | 58.03515s | 150.60645s |
| Life  Router  Mncount  1,5 | 5 | 10m | 160 | 13.2375 | 6.303571% | 31 | 0 | 82.1805s | 169.75098s |
| MaxProp | 5 | 10m | 160 | 166.21875 | 79.15179% | 209 | 1 | 91.04826s | 209.21191s |
| Passive  Router | 5 | 10m | 160 | 0 | 0 | 0 | 0 | 0 | 0 |
| Prophet  Router  15seconds | 5 | 10m | 160 | 203.225 | 96.77382% | 208 | 129 | 70.60033s | 145.06079s |
| SprayWait  nrOfCopies 2,  binaryMode 1 | 5 | 10m | 160 | 1.41875 | 0.6755953% | 2 | 1 | 24.170313s | 140.05005s |
| SprayWait  nrOfCopies 5,  binaryMode 1 | 5 | 10m | 160 | 4.71875 | 2.2470238% | 5 | 3 | 17.842371s | 76.93994s |
| WaveRouter  immunTime 60,  custodyTime 1 | 5 | 10m | 160 | 207.88126 | 98.99107% | 209 | 188 | 30.458557s | 85.245605s |
| EpidemicOracle | 5 | 10m | 160 | 208.0625 | 99.07738% | 210 | 137 | 30.988373s | 89.76514s |
| EpidemicOracle | 5 | 30m | 160 | 205.94376 | 98.06846% | 209 | 90 | 69.45978s | 133.55371s |
| Epidemic  Oracle | 5 | 60m | 160 | 207.88126 | 98.99107% | 209 | 188 | 30.343094s | 85.288086s |
| Epidemic  Oracle | 5 | 100m | 160 | 208.35 | 99.214294% | 209 | 196 | 10.396374s | 41.643433s |
| Epidemic  Oracle | 10 | 10m | 160 | 207.49374 | 98.80654% | 209 | 76 | 106.48325s | 275.23828s |
| Epidemic  Oracle | 10 | 60m | 160 | 208.79375 | 99.4256% | 209 | 188 | 31.596302s | 86.10913s |

Análise critica

O grupo de modo a conseguir realizar uma decisão em qual protocolo de encaminhamento fosse mais adequado, assim como os respetivos parâmetros, decidiu construir uma tabela com os resultados obtidos dos vários testes a que o nosso sistema foi sujeito. Esta foi a única maneira de chegar a uma conclusão de modo a que o nosso sistema fosse o mais eficiente e o mais fiável possível, para isso foi preciso analisar os seguintes parâmetros:

* TTL;
* Alcance;
* Total de mensagens criadas;
* Média de mensagens entregues;
* Percentagem de mensagens entregues;
* Máximo de mensagens entregues;
* Mínimo de mensagens entregues;
* Atraso médio total;
* Atraso máximo.

De acordo com estes parâmetros formamos a nossa solução, sendo assim o protocolo de encaminhamento escolhido foi o “*EpidemicOracle (EpidemicRouter)”.* Este protocolo cujas mensagens atingem o maior número de carros e assim, o que difunde melhor a mensagem sobre os acidentes num determinado local.

Conclusão