Mestrado em Engenharia Informática
Dissertação/Estágio
Relatório Final

CHOPIN-MANET II - Comunicação Ad Hoc para Robótica de Busca e Salvamento

André Filipe Simões Gomes
afgomes@student.dei.uc.pt

Orientadores:
Filipe Araújo
Rui Rocha

Data: 6 de Julho de 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Júri

Luís Miguel Machado Lopes Macedo
Pedro Miguel Machado Lopes da Cunha Abreu

Resumo

As redes Ad Hoc são compostas por dispositivos móveis, que comunicam através de ligações sem fios, por broadcast. São projectadas, para áreas onde não existem infraestruturas físicas que possibilitem uma administração centralizada. Os nós são responsáveis pela estruturação da rede, trocando mensagens de controlo, que lhes permite construir um mapa da rede.

A mobilidade aleatória dos nós, obrigou à criação de novos protocolos de encaminhamento, que garantissem, a reconfiguração de rotas e assim, contornar a dinâmica da rede. Para tal, a rede é inundada com mensagens de sinalização, gerando problemas ao nível da escalabilidade da rede.

O presente trabalho, foca-se num protocolo de encaminhamento Ad Hoc, o CHOPIN. Este protocolo, foi desenvolvido para a gestão de redes móveis, em situações de emergência, sendo necessário garantir a cooperação entre as equipas de socorro humanas e robôs. Em tais cenários, torna-se complicado a implementação de infraestruturas físicas. O protocolo tem de assegurar a comunicação entre os diferentes nós da rede e que todos os intervenientes, humanos e robôs, mantém um mapa da rede actualizado.

A simulação, é a ferramenta escolhida, para analisar a performance do CHOPIN, tempo de convergência e quantidade de sinalização gerada. São implementadas técnicas que permitam superar os problemas encontrados.

Palavras-chave: MANET; protocolo de encaminhamento; redes de comunicação sem fios Ad Hoc; robótica de busca e salvamento; simulador Adevs; simulador NS-3.

Abstract

Ad Hoc networks are composed by mobile devices that communicate through wireless broadcast. They are designed for areas where there isn't any physical infraestructures that allow a centralized administration. Nodes are responsible for building the network topology, by exchanging control messages that allow them to build a network map.

Nodes's random mobility, forced the creation of new routing protocols, who guaranteeing, routes reconfiguration to go around network's dynamic. To be able to do that, network is flooded by control messages, creating problems to network's scalability.

Present work, focus on a Ad Hoc routing protocol, the CHOPIN. This protocol, was developed to manage mobile networks, in critical situations, where cooperation among humans and robots, is needed. In such scenarios, is very complicated to build physical infrastructures. The protocol must to assure communication among nodes and that all intervenients, robots and humans, maintain an updated network map.

Simulation, is the framework chosen, to analyze CHOPIN's performance, convergence time and amount of signaling generated. Techniques to overcome encountered problems are implemented.

Keywords: MANET; routing protocol;wireless Ad Hoc networks communication; robotics fo search and rescue; Adevs simulator; NS-3 simulator.

Índice

1 Introdução....1
1.1 Contexto....1
1.2 Motivação....2
1.3 Contribuições....3
1.4 Estrutura do documento....3
2 Estado de Arte....5
2.1 Redes de Sensores sem Fios....5
2.2 Protocolos de Encaminhamento....5
2.2.1 Distance-vector vs Link-state....6
2.2.2 Pró-activos....7
2.2.3 Reactivos....8
2.2.4 Híbridos....8
2.3 Protocolos Ad Hoc....8
2.3.1 Optimized Link State Routing....8
2.3.2 Ad-Hoc On-Demand Vector....12
2.3.3 Dynamic MANET On-demand....13
2.3.4 Dynamic Source Routing....15
2.3.5 Zone Routing Protocol....16
2.3.6 Babel....16
2.3.7 TREBOL....16
2.3.8 CHOPIN....18
3 Simulador Adevs....25
3.1 Simulador de Eventos Discretos....25
3.2 Análise de Ferramentas....26
3.3 A Discret Event Simulator....27
3.3.1 Instalação....27
3.3.2 Modelos Atómicos....28
3.3.3 Modelos de Rede....29
3.3.4 Integração do CHOPIN no Adevs....30
3.3.5 Funcionalidades....31
....31
3.4 Sumário....35
4 Experimentação com o Simulador Adevs....36
4.1 Descrição dos testes realizados....36
4.2 Teste de conectividade da rede....37
4.3 Análise do protocolo CHOPIN....38
4.4 Detecção de Vizinhos comuns....40
4.5 Nós de último nível....42
4.6 Envio de múltiplos caminhos....46
4.7 Estrutura de dados – HigherLinkSet....49
4.8 Sumário....50
5 Simulador NS-3....52
5.1 Instalação....52
5.2 Módulos....52
5.3 Funcionalidades....52
5.4 Sumário....53

6 Experimentação com o Simulador NS-3.....54
6.1 Descrição dos testes realizados.....54
6.2 Colisões.....55
6.3 Protocolo CHOPIN V1.....57
6.4 Protocolo CHOPIN V2.....59
6.5 Protocolo CHOPIN V3.....60
6.6 Protocolo CHOPIN V4.....63
6.7 Protocolo CHOPIN V5.....64
6.8 Sumário.....66
7 Tempo Médio de Convergência vs Sinalização.....67
8 Conclusão.....68

Índice de Figuras

Figura 2.1: OLSR – Flood normal....9
Figura 2.2: OLSR – Flood com MPR....9
Figura 2.3: OLSR – Mensagem Hello....10
Figura 2.4: OLSR – Ligação bidireccional....11
Figura 2.5: OLSR – Mensagem MID....11
Figura 2.6: OLSR – Mensagem TC....12
Figura 2.7: Cenário de exemplo da propagação de um RREQ....13
Figura 2.8: DSR - Propagação de um RREQ....15
Figura 2.9: TREBOL - Temporizador Backoff....18
Figura 2.10: CHOPIN – Base Station e Node beacon....20
Figura 2.11: CHOPIN - Percurso de um Base Station Beacon....20
Figura 2.12: CHOPIN – Percurso de um Node Beacon....21
Figura 2.13: CHOPIN – Validação de um Beacon....23
Figura 3.1: Adevs – Estrutura de directorias....27
Figura 3.2: Adevs – ciclo de um modelo atómico....28
Figura 3.3: Adevs – Interacção entre modelos atómicos....29
Figura 3.4: Adevs – Modelo de rede....29
Figura 3.5: Simulador – Exemplo de um log....33
Figura 3.6: Cenário 2 – 16 nós....34
Figura 3.7: Cenário 4 – 250 nós....34
Figura 4.1: Teste à conectividade dos nós numa área de 400x300 m....37
Figura 4.2: Chopin V1 – Tempo de Convergência médio, cenário 2....38
Figura 4.3: Chopin V1 – Sinalização, cenário 2....38
Figura 4.4: Chopin V1 – Tempo de Convergência médio, cenário 1....39
Figura 4.5: Chopin V1 – Sinalização, cenário 1....39
Figura 4.6: Exemplo com 4 nós....41
Figura 4.7: Chopin V2 – Tempo de Convergência médio, cenário 1....42
Figura 4.8: Chopin V2 – Sinalização, cenário 1....42
Figura 4.9: Exemplo de um cenário em árvore....43
Figura 4.10: Chopin V3 – Tempo de Convergência médio, cenário 2....44
Figura 4.11: Chopin V3 – Sinalização, cenário 2....44
Figura 4.12: Chopin V3 – Tempo de Convergência médio, cenário 1....45
Figura 4.13: Chopin V3 – Sinalização, cenário 1....45
Figura 4.14: Comparação nos níveis de sinalização, com as versões 1 e 3....46
Figura 4.15: Exemplo de um cenário com uma bifurcação....46
Figura 4.16: Chopin V4 – Tempo de Convergência médio, cenário 2....47
Figura 4.17: CHOPIN V4 – Sinalização, cenário 1....47
Figura 4.18: Chopin v4 – Tempo de Convergência médio, cenário 1....48
Figura 4.19: Chopin V4 – Sinalização, cenário 1....48
Figura 4.20: CHOPIN V5 – Tempo de Convergência médio, cenário 1....50
Figura 4.21: CHOPIN V5 – Sinalização, cenário 1....50
Figura 4.22: Comparação nos níveis de sinalização, com as versões 1, 3 e 5....50
Figura 6.1: Hidden Terminal Problem....55
Figura 6.2: Teste de colisões, CHOPIN V1....56
Figura 6.3: CHOPIN V1 - Tempo de Convergência Médio, cenário 2....58
Figura 6.4: CHOPIN V1 - Sinalização, cenário 2....58

Figura 6.5: CHOPIN V1 - Tempo de Convergência Médio, cenário 1.....59
Figura 6.6: CHOPIN V1 - Sinalização, cenário 1.....59
Figura 6.7: CHOPIN V2 - Tempo de Convergência Médio, cenário 1.....60
Figura 6.8: CHOPIN V2 - Sinalização, cenário 1.....60
Figura 6.9: CHOPIN V3 - Tempo de Convergência Médio, cenário 2.....62
Figura 6.10: CHOPIN V3 - Sinalização, cenário 2.....62
Figura 6.11: CHOPIN V3 - Tempo de Convergência Médio, cenário 1.....62
Figura 6.12: CHOPIN V3 - Sinalização, cenário 1.....62
Figura 6.13: CHOPIN V4 – Tempo Médio de Convergência, cenário 1.....64
Figura 6.14: CHOPIN V4 – Sinalização, cenário 1.....64
Figura 6.15: CHOPIN V5 – Tempo Médio de Convergência, cenário 2.....65
Figura 6.16: CHOPIN V5 – Sinalização, cenário 2.....65
Figura 6.17: CHOPIN V5 – Tempo Médio de Convergência, cenário 1.....65
Figura 6.18: Sinalização, cenário 1.....65
Figura 7.1: Tempo médio de convergência vs sinalização.....67

Índice de Tabelas

Tabela 2.1: OLSR - Link SET.....9
Tabela 2.2: OLSR - Neighbor Set.....9
Tabela 2.3: OLSR - 2-hop Neighbor Set.....10
Tabela 2.4: OLSR - MPR Selector Set.....10
Tabela 2.5: AODV – Tabela de encaminhamento.....14
Tabela 2.6: DYMO – Tabela de encaminhamento.....14
Tabela 2.7: TREBOL - Mensagem CM.....18
Tabela 2.8: CHOPIN - Duplicate Set.....22
Tabela 2.9: CHOPIN - Link Set.....22
Tabela 2.10: CHOPIN - Routing Table.....23
Tabela 2.11: CHOPIN - Temporizadores.....23
Tabela 3.1: Simulador – Estado inicial.....30
Tabela 3.2: Simulador – Parâmetros de cálculo, do tempo de ocorrência, de um evento de recepção 31
Tabela 3.3: Simulador – Parâmetros de simulação.....32
Tabela 3.4: Simulador – Níveis de logs.....32
Tabela 3.5: Simulador – Tipos de logs.....33
Tabela 3.6: Simulador – Cenários de simulação.....34
Tabela 3.7: Simulador – Valores de retorno.....35
Tabela 4.1: CHOPIN – Higher Link Set.....49
Tabela 6.1: NS-3, CHOPIN V1 - parâmetros de configuração.....58
Tabela 6.2: NS-3, CHOPIN V2 - parâmetros de configuração.....60
Tabela 6.3: NS-3, CHOPIN V3 - parâmetros de configuração.....61
Tabela 6.4: NS-3, CHOPIN V4 - parâmetros de configuração.....63
Tabela 6.5: NS-3, CHOPIN V5 - parâmetros de configuração.....64

Glossário

Beacon	É um tipo de mensagem que tem como função anunciar a presença, de quem o envia.
Broadcast	O pacote a enviar, não tem um nó, em particular, como destino. É destinado a qualquer nó, que se encontre no raio de transmissão.
Flooding	Algoritmo de encaminhamento, onde os pacotes são sucessivamente, encaminhados por broadcast.
Hop	Número de saltos que separam dois nós.
Overhead	Processamento ou armazenamento considerado excessivo.
Sinalização	Diz respeito às mensagens propagadas pela rede.
Throughput	Em redes de computadores, é a quantidade de dados, de uma transferência, num espaço de tempo.
Unicast	O pacote a enviar, é endereçado a um único destino.

1 Introdução

1.1 Contexto

Em certos ambientes, não é possível, considerar a criação de uma rede sem fios convencional. Numa rede deste género, a comunicação entre diferentes dispositivos sem fios, é assegurada por routers, que encaminham pacotes de outros nós, de acordo, com as rotas armazenadas na sua tabela de encaminhamento. Os nós, não têm a capacidade para comunicar entre si.

Posto isto, é necessário garantir, que os routers, cubram toda a área pretendida, para que não haja um único dispositivo, impossibilitado de se conectar à rede. Os routers devem ser colocados estrategicamente, de modo, a cobrirem toda a área geográfica pretendida. Quanto maior a dimensão do terreno e maior o número de obstáculos físicos, mais difícil é, assegurar a conectividade total da rede.

Num ambiente, como o de uma catástrofe natural, é preciso agir de forma célere. O tempo necessário, de análise e montagem da rede, pode ser inviável. Como alternativa, a redes sem fios centralizadas, surgem as redes móveis Ad Hoc. Cada nó pode se movimentar e funciona como um router. No fundo, os routers tradicionais, são substituídos por routers móveis.

É neste contexto, que surge o projecto CHOPIN [1], que explora a cooperação entre equipas humanas e robôs móveis, em missões de busca e salvamento, em cenários urbanos de pequena escala.

Uma MANET, Mobile ad hoc Wireless Network, é uma rede constituída por nós móveis, conectados por ligações sem fio. É um tipo de rede que não necessita de infraestruturas para uma administração centralizada, são os nós móveis, que, entre si, trocam a informação necessária para a construção de um mapa da rede.

A mobilidade dos nós é responsável, por sucessivas alterações na topologia da rede. Isto provoca uma grande dificuldade, na manutenção de rotas válidas. A solução para o problema, passa por, inundar a rede, com mensagens de actualização. Esta técnica, dá aos nós, a capacidade para detectar a entrada e saída, de outros nós, no seu raio de transmissão. Permitindo, em tempo útil, a eliminação de rotas inválidas ou a sua substituição por rotas válidas.

As actualizações, podem ser periódicas e assim, manter um mapa de rede actualizado, com rotas válidas a qualquer momento, ou podem ser reactivas. Numa abordagem reactiva, a construção de uma rota é iniciada, após a sua requisição e os nós têm que esperar pela descoberta de um caminho para o destino pretendido. Estas características representam, respectivamente, protocolos pró-activos e reactivos.

Contudo, a inundação de uma rede, com mensagens de actualização, gera problemas ao nível da largura de banda, que é bastante mais limitada, do que em redes cabladas. É necessário que os protocolos utilizem a largura de banda disponível, de forma eficiente, procurando soluções alternativas, que acima de tudo, consigam minimizar a quantidade de informação propagada pela rede e assim, restringir ao máximo, problemas de congestão.

Embora, seja de extrema importância reduzir a quantidade das mensagens de sinalização, é necessário assegurar que os tempos de convergência da rede, se mantém em níveis aceitáveis . Os nós móveis, têm também, de ter em conta, as limitações no tempo de vida das baterias e do processamento computacional.

No protocolo CHOPIN são considerados dois tipos de agentes: os nós móveis, aos quais pertencem humanos e robôs e que têm como função, a exploração do cenário; um nó fixo, designado por Base Station que recebe toda a informação proveniente dos nós móveis.

O objectivo passa por permitir uma resposta mais rápida e com menores riscos para as equipas de resgate humanas e para as próprias vítimas. Num cenário destes, é necessário implementar mecanismos de comunicação sem fios entre os agentes humanos e os robôs, em cenários onde as redes de comunicação podem estar danificadas ou simplesmente nem existirem. Até a instalação de uma estrutura de comunicação sem fios, não está isenta de problemas. Num cenário de catástrofe, é bastante provável que existam diversos obstáculos físicos, que bloqueiem os sinais de rádio, trocados entre os nós móveis.

É por estas razões, que o protocolo CHOPIN, segue uma estrutura em forma de árvore. A maior parte do tráfego, pelos nós móveis, é gerado no sentido da Base Station, evitando a propagação de informação desnecessária, no sentido inverso, que apenas iria contribuir para a degradação do desempenho da rede. A manutenção de uma rede, com uma estrutura do género, retira o uso abusivo do broadcasting, característico em redes adhoc.

Contudo, a gestão de uma rede de nós em árvore, não se apresenta fácil. As tabelas de encaminhamento, têm que estar preparadas para encaminhar um pacote a qualquer momento, para um destino específico. No caso em questão, os nós móveis precisam de conseguir encaminhar um pacote para a Base Station e para isso, precisam de conhecer qual o próximo salto a dar e que esteja ao seu alcance. Por isso, o CHOPIN foi inspirado no protocolo OLSR, um protocolo pró-activo. Este tipo de protocolos, com o envio de mensagens de sinalização periódicas, conseguem manter as tabelas de encaminhamento actualizadas.

O CHOPIN, procura então, uma estrutura em árvore, que minimize a quantidade de mensagens transmitidas, criando caminhos fixos de menor custo. Mas, para tal, será sempre necessário, a difusão de mensagens por broadcast, que permitam a actualização de rotas em tempo real.

1.2 Motivação

O protocolo CHOPIN, apresenta uma solução alternativa, para a gestão de MANETs, em cenários de emergência. O protocolo foi idealizado, com o intuito de ultrapassar algumas das desvantagens, dos protocolos Ad Hoc existentes, apresentadas na secção acima.

A versão inicial do CHOPIN[2][3], foi testada e comparada contra um protocolo mais estabilizado, o OLSR, descrito neste documento, no capítulo sobre o estado de arte, na secção 2.3.1. Os testes realizados mostraram bons indícios ao nível da performance. Os testes realizados, focaram-se na análise do tempo de convergência, na quantidade de sinalização gerada e *throughput*.

Os tempos de convergência do CHOPIN, superaram os do OLSR. Adicionando restrições a caminhos e obrigando a rede a redesenhar a sua topologia, o CHOPIN continuou a apresentar melhores resultados.

No entanto, quando se comparou, a performance dos dois protocolos, ao nível da quantidade de mensagens de sinalização, propagadas pela rede, o CHOPIN, apresentou alguns problemas, gerando mais mensagens que o seu oponente.

Os níveis de *throughput* apresentados, foram equivalentes, constatando, que o débito

diminuiu com o aumento do número de nós.

Os testes realizados, mostraram, que, embora apresente resultados positivos, o protocolo CHOPIN, não está completo. Tem ainda, espaço para aperfeiçoamentos, nomeadamente, na redução das mensagens de controlo.

O presente estudo, pretende encontrar uma solução ideal, que consiga apresentar bons resultados em duas variáveis, tempo de convergência e sinalização gerada. Sem descurar na variável, tempo de convergência, é atribuída maior importância à quantidade de mensagens que são enviadas pela rede. No entanto, é imperativo encontrar um compromisso, que consiga equilibrar todos os factores envolvidos. Descoberta uma solução, que apresente um nível baixo de produção de mensagens, só pode ser considerada óptima, caso o tempo de convergência não atinja valores irrealistas ou considerados excessivos.

1.3 Contribuições

O presente documento, foca-se no estudo do protocolo CHOPIN. Este estudo, consistiu na realização de simulações, de forma a analisar a performance do protocolo. A cada simulação, foram essencialmente, considerados factores como o tempo médio de convergência e quantidade de sinalização gerada.

Foram analisados vários protocolos, CHOPIN original e versões modificadas que visam melhorar o seu desempenho, através da introdução de novas técnicas. Estas técnicas visam, essencialmente, reduzir o tráfego gerado.

A fase de simulação foi divida em duas fases. A primeira fase consistiu em simular o protocolo, com a ferramenta Adevs, tendo, numa segunda fase, sido utilizada a popular ferramenta, para simulação de redes, o NS-3. Estes dois simuladores, permitiram testar e avaliar o desempenho dos diversos protocolos criados.

Foi definida uma linha de compromisso experimental, entre duas variáveis de avaliação: tempo médio de convergência e sinalização. Se foram geradas muitas mensagens, o tempo de convergência será reduzido e vice-versa.

1.4 Estrutura do documento

O capítulo 2, descreve o estado da arte. São explicados conceitos como, MANET, Redes de sensores sem fios e protocolos de encaminhamento. Vários protocolos ad hoc, são referenciados, incluindo o protocolo CHOPIN.

O capítulo 3, apresenta o conceito de simulação de eventos discretos e define algumas ferramentas para o efeito. É ainda, apresentado, um simulador construído, para simular o protocolo CHOPIN. O simulador foi desenvolvido com a ferramenta Adevs.

O capítulo 4, faz a análise dos resultados obtidos, das diversas experiências de simulação do protocolo CHOPIN. São apresentadas conclusões, para cada experiência e definidas melhorias, a implementar no protocolo e resultados obtidos. Neste capítulo, o simulador utilizado foi o Adevs.

O capítulo 5, começa por apresentar a popular ferramenta, de simulação de redes, o NS-3. É ainda, apresentado o simulador desenvolvido com esta ferramenta. O simulador tem

funcionalidades similares com o simulador Adevs, descrito no capítulo 3.

O capítulo 6 aborda o mesmo tipo de análise, que foi feita no capítulo 4. Desta vez, foi utilizado o simulador integrado na ferramenta NS-3. São também abordados as variantes que este simulador oferece e que permitem testar o protocolo CHOPIN, num ambiente com características reais. É destinada uma secção, para analisar a influência e o prejuízo, derivado da colisão de pacotes.

No capítulo 7, é apresentado o compromisso definido entre sinalização e tempo médio de convergência.

No capítulo 8, é apresentada a conclusão desta dissertação. É feito um breve resumo, de todo o conteúdo deste trabalho, bem como, as análises, que foram realizadas, em cada fase da dissertação. O capítulo termina, com algumas referências, para trabalho futuro.

2 Estado de Arte

Neste capítulo, serão abordados temas relacionados a redes Ad Hoc.

É apresentada uma breve introdução sobre redes de sensores sem fios, que possuem algumas semelhanças, com as MANET.

De seguida, são descritos alguns protocolos de encaminhamento, relevantes para o desenvolvimento do trabalho.

Por último, são apresentados, vários protocolos Ad Hoc, entre os quais o protocolo CHOPIN, ao qual foi dado um maior ênfase, pois é o tema do presente trabalho. Protocolos como TREBOL e OLSR, são também descritos com mais detalhe, pelas similaridades encontradas com o CHOPIN. O primeiro, apesar de se apresentar, como um protocolo para VANET, constrói a sua topologia de rede em árvore, tal como o CHOPIN. Já o OLSR, foi utilizado como inspiração para o desenvolvimento do protocolo CHOPIN.

2.1 Redes de Sensores sem Fios

Uma WSN, Wireless Sensor Network, é uma rede de ligação sem fios, com os nós distribuídos pela rede, a utilizarem sensores, com o objectivo de monitorizar as condições físicas e ambientais, como por exemplo temperatura ou controlo de tráfego.

Tal como as MANET, são redes auto-configuráveis, que têm de conseguir adaptar-se a constantes mudanças, ao nível da topologia da rede. Os nós da rede têm que ter mecanismos auto-sobrevivência porque a intervenção humana é limitada. Têm que ser capazes de sobreviver a péssimas condições ambientais, de conseguir lidar com falhas de outros nós e realizar a recolha de energia e armazenamento de energia adicional. A duração de vida das baterias é, aliás, um dos principais problemas a ter em conta.

Este tipo de redes tem limitações ao nível do tempo de vida, capacidades de largura de banda e de memória e de velocidade de processamento, sendo mais limitadas que as MANET. Os seus nós possuem menor alcance na transmissão de dados e possuem uma maior taxa de erros. As WSN são maiores que as MANET, no que a quantidade de nós diz respeito.

Os nós organizam-se em clusters e um nó pode agir como o seu gestor. Após a detecção de eventos, por parte do sensor, o nó processa a informação e faz um broadcast dos dados pelo cluster. As comunicações , tanto podem seguir o protocolo IEEE 802.11 ou por Bluetooth.

2.2 Protocolos de Encaminhamento

Protocolos de encaminhamento distribuem informação de encaminhamento por toda a rede. Um nó da rede, tem apenas de saber interpretar e com essa informação, actualizar a sua tabela de encaminhamento.

Uma tabela de encaminhamento, contém rotas para outros nós. Sempre que um nó necessite de enviar um pacote, consulta a sua tabela de encaminhamento e se encontrar uma rota para esse destino, pode encaminhar a mensagem.

Uma rota não é composta, por todos os nós, por onde a mensagem tem que passar. Apenas basta, inserir na tabela, o endereço da interface do próximo nó, conhecido por *Gateway*. Uma rota contém outros campos, nomeadamente: endereço e máscara de rede, endereço da interface de saída e a distância que separa a fonte e o destino, cujo o valor, depende da métrica que é utilizada.

O tipo de informação, propagada pela rede, depende do tipo de protocolo, que se encontra a gerir a rede. Uma mensagem, pode conter toda a tabela de encaminhamento de um nó ou, simplesmente, descrever estado de ligações de um nó, para com os seus vizinhos. Pode ser difundida por toda a rede ou apenas entre nós vizinhos.

Estas questões são aprofundadas, ao longo, das subsecções seguintes, enquanto são apresentados, alguns protocolos de encaminhamento..

2.2.1 Distance-vector vs Link-state

Um router, é um encaminhador de pacotes. Ele deve conhecer a topologia da rede, de forma a conseguir enviar o pacote, para o destino correcto, ou seja, saber se pode enviar pacotes para o destino, directamente, se estiver ao seu alcance ou, caso contrário, encaminhar os pacotes por outros routers, para que estes, consigam alcançar o destino.

Desta forma, é essencial para um router, armazenar caminhos, para diferentes destinos. Assim, mesmo que o nó final, não esteja no raio de transmissão, o router, saberá, para que nó intermédio, o pacote deva ser encaminhado. O nó intermédio, passará a ser o responsável por fazer com que a informação chegue ao destino final.

O router armazena as diferentes rotas, numa estrutura interna, chamada, tabela de encaminhamento. Uma tabela de encaminhamento contém, para um dado destino, qual o próximo router a enviar o pacote e qual o custo para chegar ao destino. O custo entre dois nós, pode ser calculada por diferentes métricas, por exemplo, desde o número routers que o pacote tem de passar, entre o nó que o está a processar e o destino final (hops ou saltos), até dados estatísticos sobre atrasos nos nós e largura de banda disponível.

A topologia de uma rede, sofre alterações, ao longo do tempo. As alterações surgem devido a diversos factores, como por exemplo, falha ou remoção de nós, o que origina, quebras nas ligações existentes para os outros nós.

Os routers, têm, de forma dinâmica, adaptar-se às alterações que vão acontecendo, pois a quebra de uma ligação, pode deixar um nó fora do alcance. É, por esta situação, que são trocadas mensagens de actualização, entre os diferentes routers, para que todos estejam actualizados.

Um protocolo do tipo Distance-vector, obriga, um router, a enviar aos seus vizinhos, periodicamente, alterações na topologia de rede. Um nó, através da informação recebida, decide se deve, ou não, actualizar a sua tabela de encaminhamento. Apenas rotas com um custo menor serão consideradas.

Um router, que segue um protocolo do tipo Link-state, armazena internamente, um mapa de toda a da rede. O mapa consiste na construção de um grafo, que representa todas as conexões existentes, entre os diversos routers da rede.

Cada nó, utiliza a sua representação do grafo de conectividades, para calcular os melhores caminhos para todos os destinos da rede. Estes caminhos irão formar a tabela de encaminhamento de um router.

Para a construção de um grafo de conexões, em cada nó, é necessária a troca de

mensagens de actualização entre eles.

Um nó envia mensagens periódicas, aos seus vizinhos. Sempre que detectar alguma mudança nas ligações, gera uma mensagem em que inclui todos os seus vizinhos. Esta informação é então, propagada por toda a rede.

Durante a construção de um grafo, uma ligação, apenas é considerada, se for bidireccional, ou seja, se ambos os nós que a formam, a tiverem incluído nas suas mensagens de actualização.

Depois do grafo formado, são calculadas as melhores rotas para cada destino, através de algoritmos de cálculo do caminho mais curto, como por exemplo, Dijkstra.

As mensagens de actualização dos protocolos Link-state, são bastante pequenas, descrevem apenas o estado das ligações, mas são propagadas por toda a rede. Por outro lado, protocolos Distance-vector necessitam de enviar porções da sua tabela de encaminhamento, o que gera uma maior maior de quantidade de dados, mas apenas comunica alterações aos seus vizinhos.

Protocolos Link-state, têm menos tendência a encontrar ciclos de encaminhamento, do que os Distance-vector, pois têm um tempo de convergência mais curto e conseguem uma maior escalabilidade, estando aptos a operarem em redes de maior dimensão. Noutro sentido, são mais difíceis de implementar e de sustentar e necessitam de maior capacidade computacional, processamento e memória.

2.2.2 Pró-activos

Neste tipo de protocolos cada nó mantém uma tabela de encaminhamento que lhes permite o armazenamento de rotas, conhecendo, a qualquer momento e de forma imediata, como encaminhar um pacote para qualquer destino na rede. Os dados na tabela de encaminhamento são actualizados de forma periódica através da difusão de mensagens pela rede ou aquando de uma alteração na topologia da rede, como por exemplo, uma falha de conectividade entre nós.

O armazenamento de rotas evita atrasos na transmissão de pacotes já que um nó apenas necessita de consultar a sua tabela de encaminhamento para saber o próximo salto a realizar e permitir que o pacote chegue ao destino pretendido. Se por algum motivo, a rota para alcançar um certo nó for desconhecida, os pacotes recebidos para esse nó terão que ser colocados numa fila de espera, enquanto o nó actual tenta descobrir uma nova rota, através da difusão de mensagens pela rede.

A difusão de mensagens de actualização das tabelas de encaminhamento, de forma periódica, é um revés dos protocolos pró-activos pois geram um grande overhead ao nível de consumo de cpu e de largura de banda, já que como o processo de actualização decorre através do flooding da mensagem, um nó poderá receber o mesmo pacote de diversos vizinhos sobrecarregando a rede com informação redundante. Sendo assim, quanto maior for o número de nós pior será a performance de um protocolo pró-activo pois poderá gerar um número excessivo de pacotes a circular pela rede e que poderiam provocar o seu congestionamento. O maior consumo de largura de banda e de processamento tornam o consumo de energia, por parte do dispositivo móvel, maior.

Uma outra desvantagem a considerar é a possibilidade de existirem rotas guardadas nas tabelas de encaminhamento que poderão nunca ser utilizadas.

2.2.3 Reactivos

Um protocolo reactivo não exige a um nó que mantenha na sua tabela de encaminhamento rotas que não estão a ser precisas no imediato. Sempre que seja necessário encaminhar um pacote para um destino desconhecido é iniciado um processo de descoberta através do flooding de mensagens pela rede. A descoberta de uma rota para o destino pretendido pode terminar com sucesso, caso o destino seja encontrado e consequentemente uma rota seja devolvida, ou com insucesso, no caso de nenhuma rota seja devolvida.

O processo de descoberta de rotas que se inicia quando é necessário encaminhar um pacote para um nó de destino cuja a rota é desconhecida, gera um atraso inicial bastante significativo já que é necessário realizar o flooding das mensagens e esperar que surja uma resposta, o que obriga a que a mensagem fique em trânsito até que um caminho seja definido. Por outro lado, o facto de as rotas não serem permanentes acaba por gerar um menor consumo de largura de banda, em comparação com protocolos pró-activos, porque apenas é necessário realizar o flooding quando se pretende descobrir uma rota.

A técnica de flooding, consiste, em de forma sucessiva, que um nó que deseje enviar um pacote, faça um broadcast para a rede com a mensagem. Todos os nós ao alcance da transmissão wireless irão receber o pacote e processá-lo de acordo com o protocolo. Se esses nós, precisarem de reenviar o pacote, o processo repete-se, realizando, cada um deles, um novo broadcast. É esperado que com esta técnica, no limite a mensagem chegue a todos os nós da rede e assim encontrar o caminho desejado.

2.2.4 Híbridos

Os protocolos híbridos tentam conciliar o melhor que os protocolos pró-activos e reactivos têm para oferecer e assim absorver as vantagens que ambos garantem e ao mesmo tempo conseguirem ultrapassar as suas limitações.

Um protocolo híbrido divide a rede por zonas. Os nós constituintes de uma zona comunicam entre si utilizando um protocolo pró-activo, trocando as suas tabelas de encaminhamento de forma periódica, sendo que, um nó de uma zona apenas tem conhecimento da topologia da sua zona. Quando é necessário o encaminhamento de pacotes para fora de uma zona o nó inicia um processo de descoberta de rotas usando um protocolo reactivo.

2.3 Protocolos Ad Hoc

2.3.1 Optimized Link State Routing

O OLSR, Optimized Link State Routing, é um protocolo pró-activo, o que significa que todos os nós deverão conhecer a topologia da rede trocando entre eles actualizações periódicas do seu conhecimento da rede. Tem portanto, um grande overhead originado pelo constante tráfego de actualizações, mas por outro lado, não há atrasos na transmissão de pacotes.

O facto de seguir uma abordagem Link State permite reduzir o tamanho das mensagens de actualização mas a principal vantagem deste protocolo é a redução da sobrecarga na rede com a utilização de uma técnica que procura eliminar a redundância de pacotes derivada das mensagens periódicas de actualização. Essa técnica restringe a certos nós na rede, os MPR (MultiPoint Realying), a possibilidade de retransmitirem mensagens a outros nós. Consegue, desta forma, reduzir o congestionamento provocado pelo flooding, como é possível verificar pela análise das figuras 2.1 e 2.2. Na figura 2.2, os MPR estão identificados a preto.

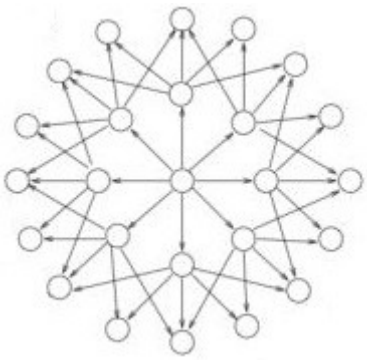

Figura 2.1: OLSR – Flood normal

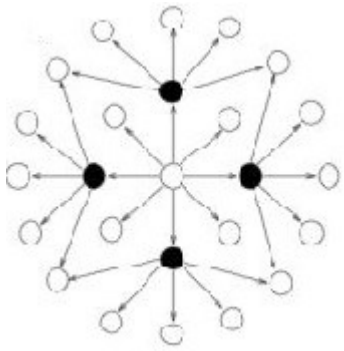

Figura 2.2: OLSR – Flood com MPR

O protocolo suporta vários tipos de estruturas de dados com diferentes funções: Link Set, Neighbor Set, 2-hop Neighbor Set, MPR Set e MPR Selector Set.

L_local_iface_addr	L_neighbor_iface_addr	L_SYM_time	L_ASYM_time	L_time

Tabela 2.1: OLSR - Link SET

A estutura Link Set, armazena informação sobre as conexões do nó com os seus vizinhos e é definida pelo conjunto de tuplos definidos na tabela 2.1. Os dois primeiros campos definem os extremos da conexão, o primeiro campo representa o endereço da interface do nó local, enquanto que, o segundo campo é o endereço do nó vizinho. Os restantes campos são temporizadores. L_SYM_TIME, define o tempo até que a conexão é considerada simétrica, ou seja, bidireccional, após o qual, a conexão é perdida. L_ASYM_time é o tempo, após o qual, a conexão, assimétrica ou unidireccional, é terminada. O último campo especifica o tempo, no qual, o registo deste tuplo, é eliminado.

N_neighbor_main_addr	N_status	N_willingness

Tabela 2.2: OLSR - Neighbor Set

A estrutura Neighbor Set tem como função, descrever os vizinhos do nó local e é

constituída pelos campos definidos na tabela 2.2. O primeiro campo é o endereço do vizinho. O segundo campo refere se o nó é simétrico ou assimétrico. O último campo especifica, se o nó local, é responsável por encaminhar o tráfego de N_neighbor_main_addr, ou seja, se é MPR desse nó.

N_neighbor_main_addr	N_2hop_addr	N_time

Tabela 2.3: OLSR - 2-hop Neighbor Set

A estrutura 2-hop Neighbor Set descreve as conexões simétricas entre o nó local e os nós a uma distância de dois hop (nós que estão a uma distância de um hop dos vizinhos e a dois hop do nó corrente) e é constituída pelos campos definidos pela figura 2.3. O primeiro campo indica o endereço do nó vizinho. O segundo campo indica o endereço de um nó a uma distância de dois hop e que é vizinho do nó com endereço N_neighbor_main_addr. O último campo é um temporizador e após o seu término, é necessário eliminar esta entrada da estrutura.

MS_main_addr	MS_time

Tabela 2.4: OLSR - MPR Selector Set

A estrutura MPR Selector Set, tem como função, registar quais os vizinhos que seleccionaram o nó, como MPR e é consituída pelos campos definidos na figura 2.4. O primeiro campo indica o endereço de um vizinho que seleccionou este nó como MPR. O segundo campo é um temporizador e após expirar, esta entrada é removida.

A estrutura MPR Set é responsável por manter uma lista de vizinhos que foram seleccionados como MPR.

Neste protocolo estão definidos quatro tipos de mensagens: Hello, TC (Topology Control) e MID (Multiple Interface Declaration).

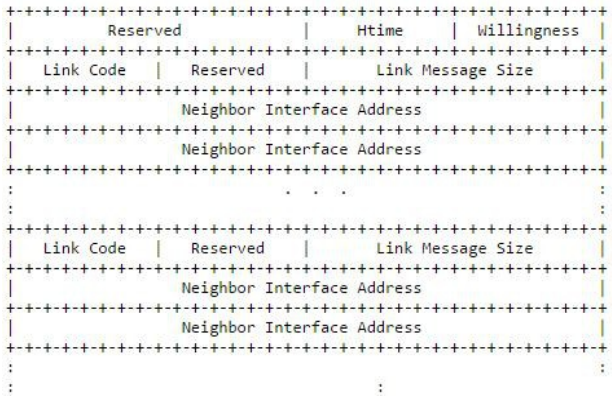

Figura 2.3: OLSR – Mensagem Hello

As mensagens Hello, ilustradas na figura 2.3, são propagadas pela rede, por broadcast e de forma periódica, sendo a sua periodicidade definida pelo campo Htime. A propagação deste tipo de mensagens, tem como objectivo, a detecção de vizinhos, estabelecimento de ligações com os nós vizinhos e a selecção de MPRs, sendo que têm um tempo de vida de um hop. O campo Willingness informa, os nós vizinhos, qual a disponibilidade de este nó em ser MPR.

O campo Link Code identifica o tipo de ligação e o tipo de vizinhos. Após cada campo, Link Code, segue-se a respectiva lista com endereços de nós vizinhos.

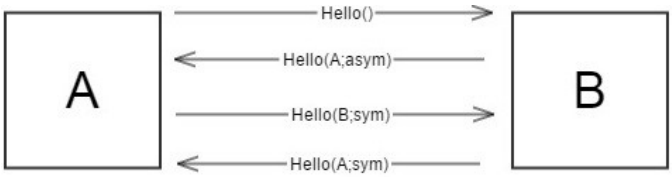

Figura 2.4: OLSR – Ligação bidireccional

A figura 2.4 mostra o processo de inicialização de conexões simétricas entre dois nós vizinhos, através da troca de mensagens Hello.

O nó A inicia a comunicação com uma mensagem Hello, na qual o nó B não está inserido. O nó B, ao verificar que não se encontra na lista enviada por A, define uma conexão com A, assimétrica, e responde com outra mensagem Hello, onde vai incluída toda a informação respeitante aos vizinhos de B. No exemplo, em questão, a informação é a de que B e A têm uma ligação assimétrica.

De seguida o nó A, fica a saber que a sua mensagem alcançou o nó B, estabelecendo deste modo, uma ligação simétrica com B, informando-o, posteriormente, do sucedido. Como etapa final, o nó B estará em condições de estabelecer uma conexão simétrica com A.

Nesta fase já está descrito o processo de detecção de vizinhos e o consequente estabelecimento de ligações. Falta portanto, apurar como é realizada a selecção de MPRs.

Após o estabelecimento de conexões simétricas com os vizinhos, um nó pode estabelecer um deles como seu MPR. Essa escolha dependerá de sempre da disponibilidade de um vizinho em ser seu MPR e, também de factores que têm em conta a escolha dos melhores MPRs, que permitem optimizar a topologia da rede. Contudo, este último factor de optimização, não é uma obrigação no funcionamento do protocolo.

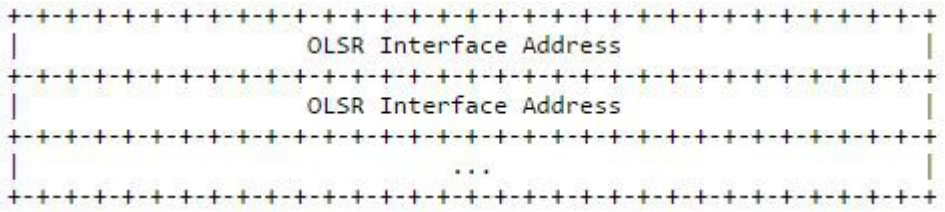

Figura 2.5: OLSR – Mensagem MID

A figura 2.5 mostra o formato de uma mensagem MID do protocolo OLSR.

Um pacote MID é difundido pela rede de forma periódica e transporta os endereços das interfaces do nó que estão a correr o protocolo OLSR. Estas mensagens têm um tempo de vida de 255 hops e são propagadas por toda a rede pelos MPRs. Um nó apenas gera uma mensagem MID se tiver pelo menos duas interfaces de rede a correr o serviço OLSR.

Pode acontecer que a quantidade de endereços a adicionar à mensagem, ultrapasse o limite máximo imposto pelo protocolo. Nesse caso, o nó terá que criar o número de mensagens MID suficientes para abranger todas as interfaces que estão a correr o protocolo OLSR.

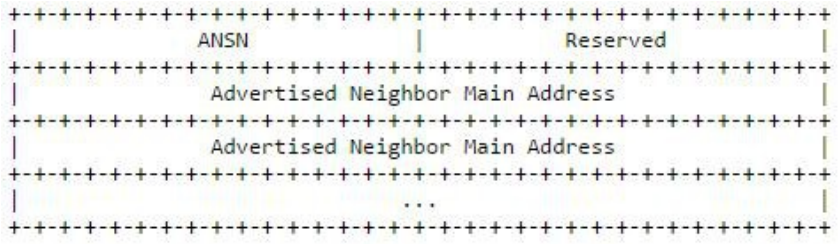

Figura 2.6: OLSR – Mensagem TC

A figura 2.6 mosta o formato de uma mensagem TC.

Uma mensagem deste tipo é difundida periodicamente por MPRs e tem incluído, os vizinhos do MPR em questão, de forma a que todos os nós na rede possam proceder ao cálculo de rotas e respectiva população das tabelas de encaminhamento.

O campo ANSN, é um número de sequência que indica a versão da tabela de vizinhos do nó. Este campo permite que quando um nó recebe um pacote TC, possa saber se a informação nele contido, é a mais actual e em caso contrário, a mensagem é ignorada.

De realçar que uma mensagem TC tem um tempo de vida de 255 hops.

2.3.2 Ad-Hoc On-Demand Vector

O AODV, Ad-Hoc On-Demand Distance Vector, é um protocolo reactivo, o que significa que um nó só conhecerá uma rota para outro nó, apenas quando precisar de comunicar com esse nó. Origina, deste modo, atraso no início de uma comunicação entre nós mas, por outro lado, consegue evitar o congestionamento da rede com actualizações periódicas de topologia da rede.

O processo de descoberta de uma rota será inicializado através de uma mensagem RREQ, Route Request, que será difundida através da rede pela técnica de flooding. O flooding de um RREQ é limitado, pois um nó apenas transmite um RREQ, para uma dada rota, uma vez, ignorando posteriores pedidos, que são derivados pela difusão da mensagem por parte dos vários nós que compõe a rede. Uma rota é encontrada com sucesso quando o nó de origem recebe um RREP, Route Reply.

Quando a mensagem RREQ alcança o destino pretendido ou quando alcança um nó intermédio que possui uma rota para o nó de destino, é então criada uma RREP para enviar ao nó que iniciou o processo de descoberta.

Cada nó que recebe um RREQ válido, irá criar uma entrada na sua tabela de encaminhamento com os parâmetros a seguir explicados. O destino, que representa o nó que

iniciou o processo de descoberta. O próximo hop, representado pelo nó que difundiu a RREQ. O número de hops para alcançar o nó de destino.

Até ao momento, apenas é possível encaminhar pacotes no sentido do nó final até ao nó inicial, o responsável pela descoberta da rota. O nó inicial ficará a conhecer esse caminho durante o envio de RREP.

Ao receber uma RREP, cada nó irá criar uma entrada na sua tabela de encaminhamento. O destino será o nó final e o próximo hop representa o transmissor da RREP.

Uma entrada na tabela de encaminhamento é removida após um certo período de tempo de inactividade.

Através da troca de mensagens Hello e da monitorização da recepção de Acknowledgements é possível verificar se ocorreu alguma falha na ligação. Após a verificação da existência dessa falha, a entrada na tabela de encaminhamento correspondente a essa rota é eliminada. Depois é enviado ao nó fonte uma mensagem RERR, Route Error, que indica que ocorreu uma quebra de ligação. O nó fonte após receber uma RERR reinicia o processo de descoberta de rotas para o destino, ao qual, acabou de se desconectar.

2.3.3 Dynamic MANET On-demand

O protocolo DYMO, Dynamic MANET On-demand, foi desenhado para ser o sucessor do protocolo reactivo AODV. Este protocolo partilha muitas características com o seu predecessor. No entanto e sendo uma segunda versão, o DYMO foi desenhado com a intenção de melhorar a performance do AODV.

No DYMO, cada RREQ ou RREP, mantém uma lista ordenada de todos os nós por onde já passou. Esta solução permite a um nó, ao analizar a mensagem, armazenar rotas para todos os nós inseridos na lista. Esta solução reduz a necessidade de realizar uma operação de procura de caminhos, em relação ao AODV, pois no final de cada processo de descoberta de rotas, cada nó envolvido (desde nó fonte ao nó de destino, contando com todos os nós intermédios), terá um maior número de rotas conhecidas para diferentes nós. Uma consequência disso, é a redução do tráfego gerado na rede, porque há uma menor necessidade de se recorrer ao flooding de uma mensagem na procura de um novo caminho, já que há uma menor probabilidade de um nó não conhecer uma rota para outro nó e ter, assim, que iniciar um novo processo de descoberta de rotas.

Cada nó responsável pela retransmissão de um RREQ ou de um RREP, ficará a conhecer uma rota para cada nó por onde a mensagem já passou. Cada uma dessas rotas será armazenada numa tabela de encaminhamento, onde o próximo hop será sempre o último nó que retransmitiu a mensagem.

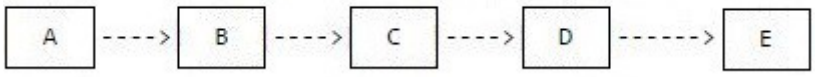

Figura 2.7: Cenário de exemplo da propagação de um RREQ

A figura representa um possível cenário da propagação de RREQ numa rede ad hoc e servirá como base a construção das tabelas de encaminhamento dos seus nós. As tabelas de encaminhamento terão como base no AODV e no DYMO e procuram demonstrar as melhorias alcançadas pelo DYMO, como evolução do protocolo AODV.

Nó	Destino	Próximo Hop	Hops
A	E	B	4
B	A	A	1
	E	C	3
C	A	B	2
	E	D	2
D	A	C	3
	E	E	1
E	A	D	4

Tabela 2.5: AODV – Tabela de encaminhamento

Nó	Destino	Próximo Hop	Hops
A	B	B	1
	C	B	2
	D	B	3
	E	B	4
B	A	A	1
	C	C	1
	D	C	2
	E	C	3
C	A	B	2
	B	B	1
	D	D	1
	E	D	2
D	A	C	3
	B	C	2
	C	C	1
	E	E	1
E	A	D	4
	B	D	3
	C	D	2
	D	D	1

Tabela 2.6: DYMO – Tabela de encaminhamento

Pela visualização das tabelas 2.5 e 2.6 é possível perceber que, para o mesmo percurso, o protocolo DYMO, consegue um maior número de rotas. Isto permite, que caso, por exemplo, o nó D, deseje contactar o nó A, já não precisará de iniciar um novo processo de descoberta de rotas, o que, como foi dito atrás, diminui consideravelmente a sobrecarga de mensgens na rede, derivada ao flooding. Por outro lado, a adição do caminho a uma mensagem, pode gerar um overhead excessivo.

2.3.4 Dynamic Source Routing

O DSR, Dynamic Source Routing, é um protocolo reactivo e que ao contrário dos protocolos comuns, não realiza consultas a uma tabela de encaminhamento. Os nós mantém, numa cache de rotas, os caminhos completos. Um caminho é adicionado ao header da mensagem e todos os nós pertencentes, à rota inserida no header pelo nó fonte, poderão consultá-la e encaminhar o pacote para o próximo hop, aquele que lhe segue imediatamente.

Quando um nó não possui na sua cache uma rota para o destino com que se deseja comunicar, inicia um mecanismo de descoberta de rotas. Tal como nos protocolos reactivos descritos anteriormente, é iniciado um processo de flooding, através do broadcast de um RREQ. Na mensagem RREQ, é incluído o endereço do nó fonte, que será o primeiro nó do caminho a construir. Os nós ao alcance, ao receberem o pacote, consultam qual a rota, que a mensagem tomou e qual o destino dela. Assim, o nó pode determinar qual o sentido a dar ao pacote.

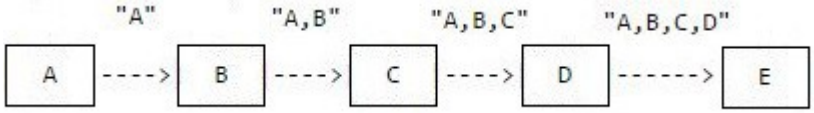

Figura 2.8: DSR - Propagação de um RREQ

A figura 2.8 procura exemplificar o processo de descoberta de rotas, que acima foi descrito, considerando a fonte como sendo o nó A e o destino o nó E. É pretendido demonstrar como a cada passo intermédio da mensagem, ela vai acumulando os endereços por onde passou e assim, originar uma nova rota.

Se a mensagem alcançou o nó de destino, ou um nó intermédio que sabe o caminho para o nó de destino, é gerado um RREP, com o destino sendo o nó que originou o processo e contendo o caminho completo do nó fonte para o nó destino. Se, por outro lado, o nó que receber um RREQ, não é o nó de destino, ele adiciona o seu endereço à rota já construída e reinicia o flooding da mensagem.

Antes de um nó enviar um RREP, terá que consultar na sua cache, se já conhece uma rota para o nó fonte. Em caso afirmativo, essa rota será adicionada à mensagem RREP, que irá seguir esse caminho. Se a rota não existe, poderão ser tomadas duas opções distintas.

A primeira opção, consiste em inverter a rota obtida da mensagem RREQ e adicioná-la à RREP. Por exemplo, de acordo com a figura acima, o nó “E” apenas teria que inverter a rota obtida, “A, B, C, D”. Ficando com a rota “D, C, B, A”.

A segunda opção, consiste em iniciar um novo processo de descoberta de rotas, neste caso, do nó “E” ao nó “A” e adicionar a rota encontrada à mensagem RREP.

A primeira solução é bem mais eficiente, já que é poupada bastante largura de banda em toda a rede. No entanto, para funcionar, terá que estar activa, a troca bidirecional de frames do protocolo IEEE 802.11 e assim evitar um segundo processo de descoberta de rotas.

Na segunda solução, quando a mensagem RREQ gerada por “E”, chega a “A”, o nó reconhece que é o nó de destino e gera uma RREP para informar a fonte, o nó “E”. Como nó “A”, não conhece o caminho para “E”, irá repetir o processo de descoberta de rotas, gerando um ciclo. De forma a contornar este problema, o nó “E”, adiciona à RREQ, uma mensagem RREP. Assim

sendo, “A” reconhece que a RREQ de “E” é uma resposta à sua RREQ inicial. Isto faz com que o nó “A” não reinicie um processo de descoberta de rota para o nó “E”.

2.3.5 Zone Routing Protocol

O ZRP, Zoune Routing Protocol, é um protocolo híbrido que usufrui das capacidades de ambos os protocolos, pró-activos e reactivos. Consegue assim aproveitar vantagens que ambos oferecem e também ultrapassar algumas das limitações que apresentam.

Este protocolo divide a rede por zonas. Uma zona tem um raio definido que representa o número de hops do nó central aos nós nas extremidades da zona. As zonas não têm de ser de igual tamanho.

Dentro de cada zona, a construção do mapa da rede é feito de forma pró-activa, o que faz com que um número mais limitado de nós esteja, de forma periódica, a trocar mensagens de actualização diminuindo os problemas em relação ao excesso de consumo de largura de banda. O protocolo de encaminhamento pró-activo interno chama-se IARP, Intra-zone Routing Protocol.

A comunicação externa à zona segue uma abordagem reactiva. O protocolo de encaminhamento reactivo chama-se Inter-zone Routing Protocol.

2.3.6 Babel

O Babel é um protocolo do tipo Distance-vector, especializado em evitar ciclos, desenhado, para redes com e sem fios. É baseado em diferentes protocolos, como por exemplo, DSDV (Dsetination-Sequenced Distance Vector Routing), AODV (descrito na secção 2.3.2) e EIGRP (Cisco's Enhanced Interior Gateway Routing Protocol). É capaz de trabalhar em redes com endereços IPv4 e IPv6.

Este protocolo, é apropriado para redes instáveis pela sua capacidade em evitar ciclos e os chamados buracos negros, locais da rede, onde a informação não avança e a origem dos pacotes não é notificada.

Está configurado, para calcular o custo de rotas, através de diferentes métricas, como, contagem de hops ou latência. Para o cálculo dos caminhos mais curto é utilizado o algoritmo Bellman-Ford, que não o mais eficiente, é capaz de lidar com pesos negativos.

Um nó, calcula a sua vizinhança, através do broadcast periódico, de mensagens Hello. Após a recpção de uma mensagem Hello, é necessário uma resposta, através de mensagens IHU (I Heard You). O custo dos caminhos, entre vizinhos, é calculado, com base em informação, extraída das mensagens Hello e IHU.

As mensagens IHU são enviadas com menor frequência que as mensages Hello, para a redução da quantidade de tráfego, gerada na rede.

2.3.7 TREBOL

O TREBOL, Tree-Based Routing and Address Autoconfiguration for Vehicle-to-Internet

Communications, é um protocolo que pretende implementar uma VANET, Vehicular ad hoc network, de forma a permitir o acesso à Internet, por parte do veículo. Numa rede do género, é necessário atender a certas características, configuração automática de endereços IP (no caso do TREBOL, os endereçõs têm a característica de serem IPv6) por parte do veículo, capacidade de encaminhamento de pacotes de dentro para fora da rede e vice-versa e conseguir lidar com a alta mobilidade que um veículo automóvel apresenta.

A alta mobilidade apresentada pelos nós constituintes numa VANET torna protocolos geográficos mais apropriados, procurando aumentar a sua performance. Neste tipo de protocolos é necessário que um nó conheça a informação geográfica de nós vizinhos. A partilha das diferentes localizações entre os vários nós é alcançada através da difusão de mensagens de sinalização periódicas pela rede. Sendo assim, cada nó terá conhecimento da posição da sua vizinhança e do destino final. O envio de um pacote seguirá uma abordagem gulosa, ou seja, o pacote será sempre encaminhado para o vizinho que esteja geograficamente mais perto. No entanto, existem protocolos geográficos que em vez de seguirem a solução gulosa, procuram determinar o próximo nó através de um maior conhecimento dos caminhos a percorrer, bem como, percepção das diferentes vias de uma estrada, o tráfego existente pelos diferentes caminhos, limites de velocidade. Apesar de o aumento de informação permitir escolher um caminho melhor, há um consequente aumento de overhead.

O encaminhamento entre a VANET e a Internet é assegurado através de vários pontos fixos localizados ao longo das estradas. Estes nós da rede são designados por RSG, Roadside Gateways, e procuram definir à sua volta uma área geográfica limitada, conhecida como, área TREBOL e que contém todos os nós que estão geograficamente posicionados dentro da sua área de alcance. Cada RSG, é responsável pelo envio de CM, Configuration Messages, mensagens periódicas que contém informação de configuração e que permitem a construção da topologia da rede.

O protocolo define ainda, uma topologia em árvore tendo em conta a posição geográfica dos nós. São construídas duas árvores diferentes, a upstream que procura encaminhar pacotes de um veículo interno para a Internet e a árvore downstream que procura realizar o encaminhamento na direcção oposta, encaminhar informação da Internet para os veículos internos.

O processo de criação da árvore upstream é iniciado pelos RSG, que é responsável pela difusão das CM dentro da área TREBOL. Neste ponto é importante definir que cada CM é identificada por um número de sequência único e que vai sendo incrementado consoante a criação de novas CM. Quando um nó móvel recebe um CM, verifica em primeira instância qual o número de sequência da mensagem, se o identificador for mais recente do que aquele recebido anteriormente, o nó actual passa a ter como pai (próximo hop), o nó que enviou a mensagem. O próximo passo consiste em decidir se a CM será reenviada de forma a prosseguir a construção da árvore. A decisão é determinada através da implementação de um temporizador, backoff, definindo que a mensagem só é reencaminhada se o temporizador expirar e assim tornar-se num nó pai. Se um nó receber uma CM com o mesmo número de sequência e com o temporizador ainda por expirar o reenvio da mensagem é cancelado. Isto significa que um outro nó, com o temporizador mais curto, é um melhor candidato a nó pai.

A árvore downstream vai sendo gerada de acordo com a necessidade de comunicação de um nó com o exterior. Após a criação da árvore upstream, cada nó passou a ter um nó pai a representar o próximo hop e cada nó não tem ideia se ficou como referência para outros nós e quais sãos esses nós. Sendo assim, quando um nó necessita de enviar dados, para um qualquer destino, os pacotes a enviar terão como destino intermédio o seu pai, ficando cada nó pai a

conhecer os seus filhos e podendo assim enviar-lhes tráfego originado na Internet.

A escolha dos nós pais assume uma enorme importância para a construção de uma topologia mais estável, já que passa por eles todo o encaminhamento de pacotes pela rede. Sendo assim, o valor do temporizador backoff é de extrema importância. O valor do temporizador é determinado através de certos parâmetros que estão embutidos numa CM, para além do número de sequência. Esses parâmetros estão descritos na tabela 2.7.

areaBoundary	Informação geográfica que descreve a área TREBOL. Nós que não estejam dentro dos limites definidos ignoram a CM.
sendPos	Localização geográfica do transmissor da CM. Este parâmetro vai sendo alterado para a localização do nó que retransmite a mensagem.
prefR	Distância preferida entre pais consecutivos.
R	Distância máxima permitida entre um nó emissor e o nó receptor. Se a distância entre dois nós for superior, a mensagem é descartada pelo nó receptor.
prefS	Parâmetro definido pelo RSG, que representa a velocidade, à qual, um nó pai se deve aproximar.
maxSpeedDiff	Nós cuja velocidade varia mais em relação a este parâmetro do que a prefS, velocidade preferida, ficam impossibilitados de se tornarem nós pais.
D_{pos} e D_{speed}	Parâmetros que determinam o valor máximo do temporizador backoff.

Tabela 2.7: TREBOL - Mensagem CM

A figura 2.9, mostra o cálculo do temporizador backoff, considerando que as variáveis pos e speed representam, respectivamente, localização e velocidade do nó actual.

$$T_{backoff} = \frac{\| \ \|pos - sendPos\| \ - prefR \ \|}{R} \times D_{pos} + \frac{\|speed - prefS\|}{maxSpeedDiff} \times D_{speed}$$

Figura 2.9: TREBOL - Temporizador Backoff

2.3.8 CHOPIN

O CHOPIN[2][3], **C**ooperation between **H**umam and r**O**botic teams in catastro**P**hic **In**cidents, tem como objectivo a cooperação entre humanos e robôs para que, em conjunto, consigam funcionar, de forma segura e eficiente, em situações de emergência dentro de cenários urbanos. O protocolo defende a complementação de humanos e robôs e não a substituição completa dos humanos por outras entidades com inteligência artificial. Tal cenário não é considerado como uma

solução realista.

Este protocolo determina três entidades: humanos e robôs, que constituem os nós móveis; CCO, Centro de Comando Operacional ou, também conhecido como Base Station, que é um posto de comando fixo que coordena as operações dos nós móveis.

Os humanos seguem os protocolos da protecção civil existentes e são aconselhados pelo CCO sobre quais as acções a efectuar segundo a informação recolhida pelos agentes robóticos

O CCO é o ponto de convergência da rede, já que é o destino, para onde a maior parte da informação gerada na rede é enviada. A informação recolhida pelo CCO será responsável pela construção de um mapa global da rede, que será apenas do seu conhecimento. A actuação dos agentes móveis irá depender substancialmente dos dados vindos do CCO.

O CHOPIN é um protocolo pró-activo, inspirado no OLSR, que faz uso de mensagens periódicas, para a recolha e propagação de informação sobre o estado da rede. Conciliando a técnica de flooding com a manutenção da rede numa estrutura em árvore, o protocolo CHOPIN consegue limitar o número de mensagens propagadas na rede e reduzir a redundância derivada do próprio flooding.

O protocolo tem definido dois tipos de mensagens: Base Station Beacon e Node Beacon. Mensagens do tipo Base Station Beacon, são originadas na Base Station e contém os dados necessários para que os nós móveis que constituem a rede, saibam como contactá-la. Uma mensagem Node Beacon é uma mensagem originada pelos nós móveis e durante o caminho até à Base Station, vão lhe sendo adicionados, os endereços de todos os nós que encaminharam a mensagem.

Base Station Beacons, são transmitidos por broadcast. A sua difusão é iniciada periodicamente pela Base Station e são retransmitidos pelos nós móveis. O objectivo, consiste em que todos os nós móveis, mantenham uma rota actualizada para a Base Station e desta forma, consigam, a qualquer momento, comunicar com a Base Station. Este tipo de beacons, é então, responsável, pela definição da topologia da rede. Só após, a sua recepção, os nós móveis, terão conhecimento, de uma rota, para iniciar a difusão de Node Beacons.

A rede começa a definir uma topologia em árvore, durante a difusão de Base Station Beacons. A Base Station passa a ser a raiz da árvore e a comunicação, é realizada na direcção dos nós mais distantes. A técnica de broadcast, dificulta a propagação de Base Station Beacons, apenas no sentido oposto à Base Station. No entanto, com recurso a uma estrutura de dados, que armazena mensagens duplicadas, é possível descartar os beacons, que tentem prosseguir na direcção da raiz. A estrutura de dados que armazenas os duplicados, é parte do protocolo CHOPIN, chama-se DuplicateSet e é apresentada com maior detalhe, ao longo desta secção.

Ao receber um Base Station Beacon, um nó móvel actualiza a sua tabela de encaminhamento, com uma rota para alcançar a Base Station. A rota é constituída pelo destino (Base Station), próximo salto (nó ao alcance de transmissão que redireccionou o beacon) e o número de hops (número de nós que receberam a mensagem, ou seja, a distância do nó para a Base Station). Cada rota, é, no conceito topológico de árvore, uma aresta que conecta dois vértices (nós). Essa ligação é unidireccional, no sentido inverso, ao sentido de propagação de Base Station Beacons.

Node Beacons, são transmitidos em unicast. Cada nó móvel, inicia a difusão do seu Node Beacon e encaminha os Node Beacons, dos seus pais, para o seu filho, que na tabela de encaminhamento, está identificado como o próximo hop, na rota que tem como destino final, a Base Station.

Desta forma, é possível garantir, que os Node Beacons, se propaguem no sentido da Base Station. Consegue-se reduzir a quantidade mensagens propagadas pela rede, eliminando informação inútil, como beacons que viajariam no sentido oposto à Base Station.

A Base Station, ao receber um Node Beacon, pode adicionar uma rota, à sua tabela de encaminhamento, tendo como destino, o nó que iniciou a propagação do beacon.

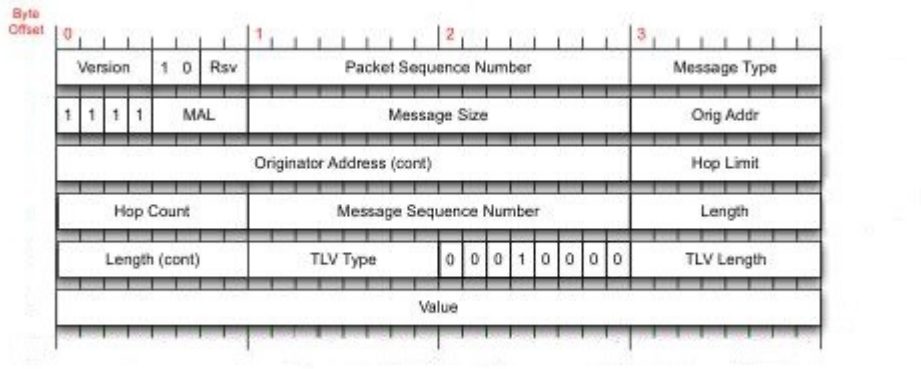

Figura 2.10: CHOPIN – Base Station e Node beacon

A figura 2.10 representa uma mensagem do tipo Base Station Beacon, que é gerada periodicamente pelo CCO. Cada Base Station Beacon tem um tamanho de 24 bytes. Aqui, há a salientar alguns dos campos constituintes da mensagem e que vão sendo influenciados ao longo do percurso da mensagem desempenhando, portanto, um papel da maior relevo que outros. Esses campos são: Orig Addr, Hop Count, Hop Limit e TLV.

Orig Addr, representa o endereço que originou a mensagem e tem uma característica importante que é a sua imutabilidade. O que significa que este campo representa, em qualquer altura, o endereço da Base Station.

No campo TLV, estará sempre identificado o último nó que reenviou a mensagem. No primeiro envio, o campo estará representado pelo endereço da Base Station.

O campo Hop Count identifica o número de nós que a mensagem já passou, ou seja, a distância a que a mensagem se encontra do CCO. Sempre que um nó vai reenviar uma mensagem, o valor do campo é incrementado.

Hop Limit representa o número máximo de saltos possíveis. Sempre que este número for ultrapassado a messagem é descartada. O campo é decrementado a cada salto.

A figura 2.11 procura demonstrar as funcionalidades dos campos atrás explicados, bem como, a sua evolução ao longo do percurso da mensagem.

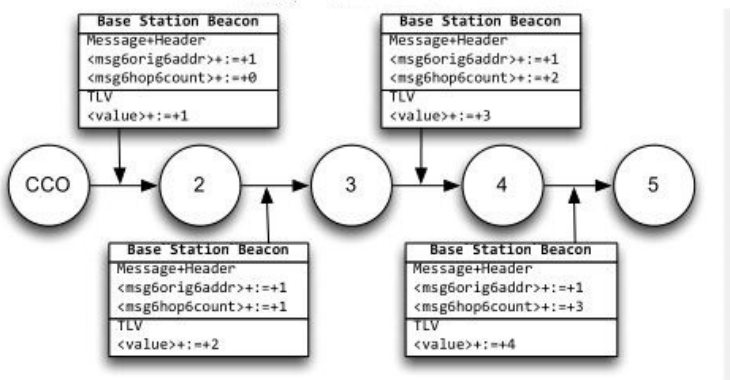

Figura 2.11: CHOPIN - Percurso de um Base Station Beacon

A figura 2.10 representa uma mensagem do tipo Node Beacon, que é gerada periodicamente pelos nós móveis na rede. Uma mensagem Node Beacon não tem um tamanho fixo, estando dependente da variação do tamanho do bloco TLV. Sendo assim, o tamanho de uma destas mensagens será de 17 bytes mais o espaço ocupado pelo bloco TLV .

Dos campos apresentados na figura 2.10 é de extrema importância destacar os campos Orig Addr, Hop Count, Hop Limit e TLV. Estes foram já explicados anteriormente, aquando da explicação do formato de uma Base Station Beacon. Em ambos os tipos de mensagens, estes campos apresentam muito em comum. Apenas o bloco TLV apresenta algumas diferenças. Este campo, numa mensagem do tipo Node Beacon, mantém uma lista de todos os nós intermédios por onde a mensagem passou. Esta lista de nós representa uma rota que será armazenada pelo CCO. A variação no tamanho da mensagem deriva, da variedade do número de nós que uma mensagem pode atravessar até ao CCO.

A figura 2.12 demonstra como uma Node Beacon se vai modificando ao longo de um percurso.

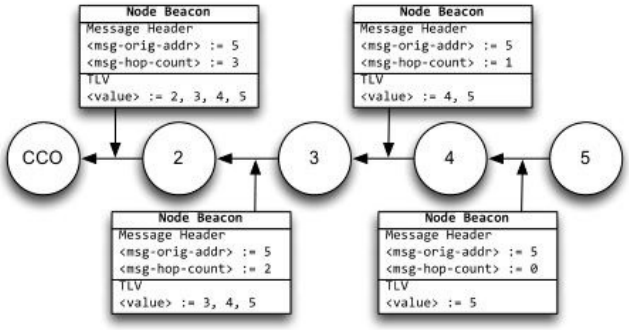

Figura 2.12: CHOPIN – Percurso de um Node Beacon

As mensagens de controlo, os beacons, são enviados sob a camada UDP. O CHOPIN está à escuta no porto 269. O ambiente da aplicação é multi-threaded, onde se destacam duas threads, uma que é responsável pela leitura e processamento das mensagens e a outra trata de enviar as mensagens para a rede.

Durante a fase de envio de mensagens, a thread responsável, utiliza uma estrutura auxiliar , onde coloca as mensagens de saída. Essa estrutura chama-se OUT-BUFFER, e é utilizada como como uma fila FIFO para guardar as mensagens que estão em espera até que estas possam ser escritas para socket.

No CHOPIN estão ainda definidas três estruturas de dados principais, responsáveis pela manutenção de informação essenciais para o correcto funcionamento do protocolo, Duplicate Set, Link Set e Routing Table.

As mensagens recebidas pelo sistema, vão sendo armazenadas na estrutura Duplicate Set com o intuito de prevenir que a mesma mensagem seja processada mais do que uma vez. As mensagens têm um tempo de expiração definido e que após o seu término a mensagem é eliminada do sistema.

Esta estrutura é representada por uma hashtable. A chave de acesso é calculada utilizando uma função de hashing dos seguintes campos: tipo de mensagem, se é Base Station Beacon ou Node Beacon; endereço de origem, representa o endereço do nó que criou a mensagem, sendo que este campo é imutável; número de sequência da mensagem, que diz respeito a cada nó.

Não é necessário armazenar todo o conteúdo da mensagem, bastando armazenar os campos que constituem a chave mais o tempo que resta à mensagem antes de ser removida, como é demonstrado pela tabela 2.8.

Originator Address	Message Type	Message Sequence Number	Expiration Time

Tabela 2.8: CHOPIN - Duplicate Set

A estrutura Link Set descreve a associação entre interfaces locais e endereços remotos dos vizinhos e CCO. A informação contida na estrutura Link Set será posteriormente utilizada para popular a estrutura Routing Table.

Esta estrutura é representada por uma hashtable. A chave de acesso é calculada utilizando uma função de hashing do endereço do vizinho.

Na estrutura são armazenados os seguintes campos, que estão identificados na tabela 2.9: o endereço da interface local; o endereço CCO ou dos vizinhos correspondentes; tempo de validade, que após o seu término, a entrada na tabela é removida.

Local Interface Address	Neighbor Address	Expiration Time

Tabela 2.9: CHOPIN - Link Set

A estrutura Routing Table é responsável pela gestão da informação de encaminhamento. É através desta informação que será possível tomar decisões sobre quais caminhos um pacote deve e pode tomar.

Esta estrutura é representada por uma hashtable. A chave de acesso é calculada utilizando uma função de hashing do endereço do destino, para o qual se pretende enviar uma mensagem.

Na estrutura são armazenados os seguintes campos, que no fundo traduzem a informação necessária em qualquer tabela de encaminhamento gerida por outros protocolos e que estão identificados na tabela 2.10: destinatário da mensagem; nó intermediário, que sabe qual o caminho para o nó de destino e para o qual corresponde o próximo destino do pacote; número de

nós que separam o nó actual do nó final.

É de ressalvar que esta estrutura, bem como toda a informação gerida por ela, está localizada no espaço de utilizador. No espaço do kernel há também acções respeitantes à função de encaminhamento. Os dados mantidos na estrutura Routing Table, são utilizados para povoar a tabela de encaminhamento do kernel, através da biblioteca NETLINK.

Destination Address	Next Hop	Hop Count

Tabela 2.10: CHOPIN - Routing Table

Como foi referido, há certos eventos periódicos que vão ocorrendo, durante o funcionamento do protocolo CHOPIN. Esses eventos estão explícitos na tabela 2.11, sendo descrito, os vários temporizadores encontrados.

REFRESH_INTERVAL	Representa o intervalo de tempo, entre a geração de novos beacons, de ambos os tipos. O temporizador apenas diz respeito a mensagens geradas pelos nós, beacons vindos de outros nós são tratados e preparados para reenvio no imediato.	2 segundos
DUP_HOLD_TIME	Representa o intervalo de tempo, onde a estrutura Duplicate Set é verificada para a remoção de entradas cuja a validade expirou.	30 segundos
NEIGHB_HOLD_TIME	Representa o intervalo de tempo, onde a estrutura Link Set é verificada para a remoção de entradas cuja a validade expirou.	6 segundos $(3 \times REFRESH\_INTERVAL)$

Tabela 2.11: CHOPIN - Temporizadores

A figura abaixo apresenta o pseudo-código, do processamento de beacons, por parte de um nó. A messagem, passa, em primeiro, por uma filtragem dos seus campos (etapas 1-6), de onde sai a decisão se a mensagem é descartada ou processada.

```
0:  Receive Message
1:  if Message Type is not valid then discard
2:  else if Hop Limit == zero then discard
3:  else if Hop count == 255 then discard
4:  else if Message Origin Address == Host Address then discard
5:  else if Duplicate Set contains Message then discard
6:  else if TLV Block is not valid then discard
7:  else Process Message
```

Figura 2.13: CHOPIN – Validação de um Beacon

1. Verificação se o campo Message Type é válido, ou seja, se representa um Base Station Beacon ou um Node Beacon.

2. Nesta etapa é feita a consulta do campo Hop Limit, de forma a verificar se o beacon já expirou o seu tempo de vida, ou seja, atingir o limite mínimo de zero.

3. Aqui é analisado se o campo Hop Count atingiu o limite máximo de 255 saltos.

4. Neste passo verificado se a mensagem a processar teve origem no nó actual. Em caso afirmativo a mensagem não é processada.

5. Aqui é feita a consulta à estrutura de dados Duplicate Set para verificar se a mensagem já foi processada e em caso afirmativo, ignorá-la. É nesta situação que entra o campo Message Sequence Number, que, ao ser incrementado a cada envio de um beacon, torna uma mensagem única.

6. Esta é a última etapa do processo de avaliação de uma mensagem, a qual, consiste em analisar o campo TLV e analisar se existe alguma anomalia e no caso de existir, descartar a mensagem. Um campo TLV, é inválido, se não conter endereços IPv4 bem formados. No caso de um Base Station Beacon o bloco TLV apenas poderá conter o endereço da Base Station, para que o beacon seja considerado válido.

7. Se esta etapa for alcançada, significa que a mensagem foi validada com sucesso. Assim sendo, o primeiro passo passa por popular as estruturas de dados, com a informação recolhida. Caso já exista uma ligação, entre os nós fonte e destino, na estrutura Link Set, o tempo de expiração é reiniciado. Se já existir uma entrada na Routing Table, entre fonte e destino, ela poderá ser substituída, caso, a distância guardada, entre os nós, for pior que a distância recolhida do beacon que está a ser analisado. O último passo passa pelo reencaminhamento da mensagem. Apenas um nó móvel pode reencaminhar beacons, sendo que, só pode reencaminhar um Node Beacon, se recebeu, previamente, um Base Sation Beacon. O beacon a encaminhar sofrerá algumas alterações: Campo Hop Count será incrementado em 1 salto; Campo Hop Limit será decrementado em 1 salto; caso a mensagem seja um Node Beacon, o endereço do nó corrente, é adicionado, no início da lista de endereços do bloco TLV; caso a mensagem seja um Base Station Beacon, o bloco TLV passará a ser constituído pelo endereço do nó corrente.

3 Simulador Adevs

Este capítulo, descreve uma aplicação construída, para a simulação do protocolo CHOPIN. A simulação de eventos discretos, é a técnica de simulação abordada e implementada no simulador.

É descrito o ambiente de simulação de eventos discretos, bem como algumas das ferramentas, do género, disponíveis.

No final é apresentado o simulador criado, como foi feita a sua integração, com a ferramenta de simulação de eventos discretos e todas as suas funcionalidades.

3.1 Simulador de Eventos Discretos

Um simulador de eventos discretos (DES, Discret Event Simulator), é definido, como um modelo de simulação em que as variáveis do sistema, alternam de estado em pontos específicos do tempo, ao contrário de um simulador contínuo, onde as variáveis alternam de estado continuamente ao longo do tempo.

O estado num sistema de simulação discreta, representa todas as variáveis que conseguem identificar as suas propriedades. A alternância entre estados, atrás descrita, significa que as suas características, as suas variáveis internas, foram alteradas. A alternância de estados acontece com o decorrer de eventos.

Um sistema de simulação discreta, é baseado numa sequência de eventos. Cada evento ocorre num particular instante do tempo, o qual, é passível de ser determinado com exactidão. Ou seja, um sistema do género, poderá manter uma lista de eventos, ordenada de forma crescente, pelo tempo da sua ocorrência. Cada evento representa um acontecimento num instante concreto no tempo e a diferença temporal entre dois eventos consecutivos representa o tempo desprezado por este tipo de modelos de simulação.

O salto temporal entre eventos, poderá ser homogéneo ou heterogéneo, consoante o tipo de modelos que se quer desenhar. Dois eventos que estão programados para o mesmo instante de tempo, terão que respeitar sempre a ordem, pela qual, foram inseridos na lista de eventos, sendo que, o salto temporal será nulo.

O programa responsável pela simulação terá sempre que ter presente o tempo actual. Para isso, o sistema necessita de manter um relógio interno, responsável pela actualização do tempo de simulação aquando da ocorrência de eventos.

O relógio interno, que identifica o tempo de simulação corrente, pode ser utilizado para definir uma condição de paragem. Por exemplo, se for pretendido que a simulação, decorra durante dez segundos, assim que o relógio ultrapasse o instante de tempo igual a dez segundos, a simulação é interrompida.

É também possível, definir condições de paragem, exclusivas ao tempo de simulação. Por exemplo, na simulação de uma fila de atendimento, pode-se definir que a simulação termina após um determinado número de clientes serem atendidos.

A simulação contínua é baseada em actividades, contrastando com a simulação de eventos discretos, onde o seu funcionamento é orientado a eventos. Em modelos contínuos, a simulação segue o desenvolvimento do sistema ao longo do tempo. O tempo é dividido em pequenos intervalos e é nestes períodos que se define a ocorrência de actividades que são responsáveis por

actualizar as variáveis internas ao sistema.

O facto de a simulação de eventos discretos, não ter que considerar todas os intervalos de tempo, consegue, tipicamente, correr um mesmo modelo, a velocidades superiores, que a simulação contínua.

3.2 Análise de Ferramentas

Esta secção pretender descrever o processo de escolha da ferramenta, a utilizar, para a criação de um simulador, para averiguar as capacidades do protocolo CHOPIN.

De forma a conhecer, as soluções de mercado existentes, com maior detalhe, foram analisadas as seguintes ferramentas de simulação de eventos discretos: Adevs, SimPy e DESMO-J. Essencialmente, foram escolhidas estas três, pelo facto, de continuarem a sofrer actualizações recorrentes, o que dá, logo à partida, uma maior confiança, na qualidade do produto.

A escolha, entre as três, acabou por cair na ferramenta Adevs. Questões preponderantes para a tomada de decisão, prendem-se, essencialmente, pela melhor documentação, a começar por um manual bem explicado, usando um exemplo em concreto e de fácil orientação. O manual pode ser encontrado na sua página web[4], mas, também está à disposição, aquando do download da biblioteca.

Para além, do manual, também vêm incluídos, diversos exemplos de simuladores, para diferentes áreas, com código fonte à disposição e prontos a serem compilados com recurso a um Makefile já criado. A aprendizagem torna-se mais flexível, já que é possível olhar como os exemplos foram construídos e no caso de dúvidas, proceder a alterações no seu código.

Outra vantagem desta ferramenta é que, a forma como ela está desenhada, permite uma fácil integração, para a simulação de modelos de rede, como é o caso do projecto deste trabalho.

Como a biblioteca foi implementada em C++, é esperado uma maior performance que os simuladores SimPy (Python) e DESMO-J (Java).

SimPy é uma ferramenta de simulação de eventos discretos baseada em processos e implementada em Python. Cada processo pode interagir entre si e com o ambiente via eventos. Um Processo é descrito como uma função geradora do Python. Processos executam à vez, podendo no entanto, ser interrompidos por outro processo que queira assumir, ou seja, que já não esteja à espera de eventos.

Esta ferramenta permite ainda, a partilha de recursos entre diferentes processos, por exemplo, durante a simulação de abastecimento de carros eléctricos. Considerando processos para identificar veículos e a estação, que mantém uma fila FIFO, com locais para carregamento de baterias. Um processo automóvel que deseje proceder ao carregamento da sua bateria terá que perguntar a um processo estação, se tem zonas de abastecimento disponíveis. Em caso afirmativo, o veículo inicia o abastecimento, caso contrário, o processo que identifica o automóvel entre em estado de espera, até a fila ter postos de abastecimento livres.

Esta ferramenta pode ainda ser útil na simulação de eventos de rede, bem como, em ambientes multi-agente. Mais informação sobre a ferramenta pode ser consultada na sua página web[5].

DESMO-J[6],Discrete-Event Simulation Modeling in Java, é outra ferramenta para simular modelos de eventos discretos e está implementada em Java. Suporta simulação híbrida, capaz de processar eventos e processos.

O DESMO-J produz, de forma automática, logs e ficheiros erros e debug, com os resultados da simulação que poderão ser visualizados numa GUI. A interface gráfica ajuda na observação das experiências, permitindo visualizar a evolução dos modelos, ao longo do tempo. Através da interface é possível controlar o fluxo de execução da simulação com as opções de iniciar, pausar, parar e reiniciar. Permite ainda, a alteração de parâmetros, o que possibilita, testar a simulação com diferentes características de forma fácil e rápida. No final, são disponibilizados todos os relatórios gerados pela ferramenta. São, ainda, oferecidas, mais duas opções de visualização. Uma animação a duas dimensões, baseada em ícones e símbolos e uma animação a três dimensões, baseada em Java 3D.

As aplicações DESMO-J, são essencialmente focadas para ambientes de manufacturação e logística. O simulador encontra-se integrado em algumas ferramentas, viradas para a gestão de processos de negócio. Mais informação sobre o simulador pode ser encontrada na sua página web[7].

A ferramenta Adevs, está descrita, com maior detalhe na secção 3.3.

3.3 A Discret Event Simulator

3.3.1 Instalação

O Adevs, A Discret EVent Simulator, é uma biblioteca desenhada para o desenvolvimento de simuladores de eventos discretos e está escrita na linguagem C++. A versão mais recente e estabilizada é a 2.8.1 e o download pode ser realizado no seu endereço web[8].

Após o download, é obtido um arquivo comprimido. A descompressão do arquivo resulta na seguinte estrutura de directorias, apresentada na figura abaixo.

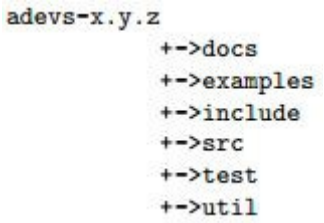

Figura 3.1: Adevs – Estrutura de directorias

Da estrutura apresentada é importante destacar três sub-pastas: docs, examples e include.

Na sub-directoria docs há documentação sobre a biblioteca.

É na sub-directoria include, que se encontram os ficheiros header necessários para a construção de qualquer simulador. Na implementação de um simulador pessoal, não é necessário incluir todos os headers, localizados no interior desta sub-directoria. Só é necessário fazer uma referência a um header, o ficheiro adevs.h.

A sub-pasta examples contém alguns exemplos de simuladores, onde é possível analisar o seu código fonte. É também possível, compilar os programas de simulação, bastando para tal, utilizar o respectivo makefile, que se encontra na mesma pasta do código fonte.

3.3.2 Modelos Atómicos

Modelos atómicos são as entidades básicas da ferramenta Adevs e o seu comportamento é descrito pelas suas funções de transição de estado, função de saída, função de avanço do tempo e a função que implementa o garbage collector. Estas funções representam métodos virtuais da classe Atomic e descrevem o ciclo de processamento de eventos de um modelo atómico, como está descrito na figura 3.2. Um modelo atómico é derivado da classe Atomic.

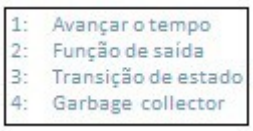

Figura 3.2: Adevs – ciclo de um modelo atómico

1. Função responsável por determinar o tempo que falta para a ocorrência do próximo evento. No início de cada ciclo, cada modelo invoca este método para saber qual o instante de tempo do evento seguinte. O modelo que apresentar o instante de tempo menor, inicia o processamento do evento, que é classificado por interno.

2. Este método é invocado imediatamente a seguir à função de avanço no tempo e permite a um modelo, enviar informação na forma de evento, a outro modelo, em resposta, ao acontecimento de um evento interno. Quando um modelo recebe um evento, com origem noutro modelo, é classificado como externo. A forma como a comunicação é realizada, envolve outro tipo de modelo, que está explicado na secção 3.3.3.

3. A transição de estado, descreve as mudanças internas, derivado do processamento de um evento. Esta etapa é composta por três funções:

    - Função de transição interna, invocada quando há eventos internos. Os eventos internos são mantidos numa lista escalonada por ordem crescente. Na primeira etapa, o modelo foi seleccionado, como tendo o evento, a decorrer no instante de tempo mais próximo, por conseguinte, o primeiro da evento da lista é escolhido para processamento.

    - Função de transição externa, invocada no instante de tempo, em que o modelo recebe, um evento (externo), por parte de outro modelo. A transmissão e recepção do evento decorrem no mesmo instante de tempo. Esta função encontra-se fora do ciclo descrito na figura 3.2, já que o modelo, apenas entra em acção, por força de factores externos.

- Função de transição confluente, invocada quando o modelo tem que tratar na mesma unidade de tempo, eventos internos e externos. Este método, substitui a função de transição interna, no ciclo atómico.

4. A última etapa consiste em limpar da memória, todos os eventos, que foram sendo trocados, pelos modelos envolvidos.

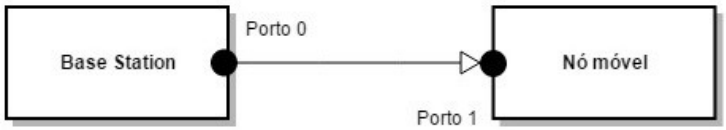

Figura 3.3: Adevs – Interacção entre modelos atómicos

A figura 3.3 representa a interacção entre dois modelos atómicos. O exemplo é composto por dois nós que seguem o protocolo CHOPIN, explicado na secção 2.3.8 e que no fundo, reflecte o foco deste trabalho.

Para se conseguir representar o protocolo CHOPIN, no simulador Adevs, é necessário, em primeiro lugar, de identificar os componentes atómicos. Esses componentes são os nós que constituem a rede.

3.3.3 Modelos de Rede

Modelos atómicos definem os comportamentos fundamentais, enquanto, modelos de rede permitem a conexão entre os vários modelos atómicos, definindo uma estrutura de comunicação. É então, possível de se afirmar, que modelos atómicos, estão contidos em modelos de rede.

Modelos de rede derivam da classe Network. Esta classe possui duas funções virtuais, uma que fornece ao sistema, o conjunto de componentes que constituem a rede e um método de encaminhamento, que implementa as conexões entre os componentes da rede. É este último método, o responsável, pelo encaminhamento de eventos, entre modelos atómicos, descritos na secção 3.3.2.

A ligação entre componentes, é realizada pelo acoplamento entre portos, que representam os pontos terminais da conexão, tal como se pode observar pela figura Figura 3.3. Cada componente pode ter um variado número de portos e associar diferentes componentes a cada um deles.

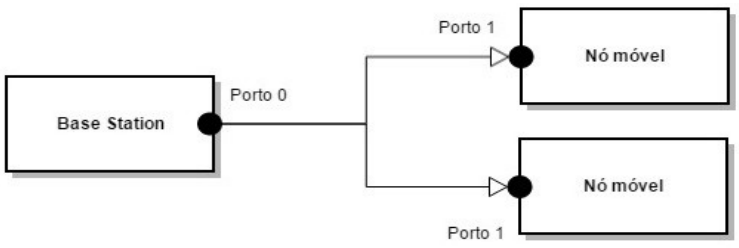

Figura 3.4: Adevs – Modelo de rede

Como está ilustrado na figura 3.4, a ligação entre a Base Station e os nós móveis, é feita pelo acoplamento do porto zero, de saída, da Base Station, com os portos número um, de entrada, dos nós móveis.

Assim sendo, o componente, para enviar eventos, precisa de definir qual o porto de saída. Isso será assumido na função de saída, do modelo atómico. Essa informação será redireccionada, para a função de encaminhamento, do modelo atómico e transportada para todos os componentes, com ligações para esse porto de saída.

Olhando para o ciclo de um modelo atómico, figura 3.2, o processo de encaminhamento será invocado no final da função de saída.

3.3.4 Integração do CHOPIN no Adevs

Em primeiro lugar, é necessário enquadrar o protocolo CHOPIN no simulador Adevs, definindo modelos atómicos e de rede. A abordagem seguida, consiste em definir os nós da rede, móveis e Base Station, como modelos atómicos e inserir esses componentes num modelo de rede que trata de simular ligações ponto a ponto. Esse cenário é exemplificado na figura 3.4.

Neste momento, é crucial definir quais as variáveis internas de um modelo atómico. As variáveis a considerar, para cada modelo atómico, são, o tempo actual de simulação e uma lista para albergar os eventos em espera, como foi descrito na secção 3.1. A unidade de medida do tempo é o segundo e a lista é ordenada, pelo tempo, do mais próximo para o mais distante.

Após este passo, é obrigatório definir que eventos devem ser criados. De acordo com a especificação do protocolo, definido na secção 2.3.8, é necessário definir eventos que estejam ligados à acção dos temporizadores do CHOPIN, envio de beacons e actualização das estruturas Duplicate Set e Link Set.

Estes eventos são classificados como internos e ocorrem de forma periódica, sendo reinseridos na lista. O tempo de ocorrência será o tempo actual mais o valor do temporizador, definido no protocolo.

Neste momento é possível definir o estado inicial de um modelo atómico.

Variáveis internas	Estado inicial
Tempo de simulação	0 s
Lista de Eventos	(Enviar beacon, T = 0) (Actualizar Link Set, T = 6 s) (Actualizar Duplicate Set, T = 30 s)

Tabela 3.1: Simulador – Estado inicial

A tabela 3.1 mostra o estado zero de um modelo atómico. O tempo de simulação começa no instante zero. Inicialmente cada componente terá uma lista com os três eventos internos. Os beacons começam a ser transmitidos, logo no instante de tempo inicial, já os eventos de actualização das estruturas, terão que esperar pelo término dos respectivos temporizadores.

Os eventos são enviados em duas situações, por acção do temporizador e por retransmissão. Um modelo receberá eventos externos, no mesmo instante de tempo, que

modelos, aos quais ele está conectado, transmitirem beacons, sob a forma de eventos. No intervalo, em que um evento é enviado por um modelo e é recebido por outro, há um processo de encaminhamento, inerente ao modelo de rede. O processo consiste em verificar, quais os nós que estão ao alcance do transmissor e só a esses, enviar o evento. Cada receptor receberá um evento, com a origem idêntica, mas com tempo de ocorrência diferente, já que os nós estarão a distâncias diferentes do transmissor.

Após receber um evento, um modelo, irá colocá-lo na sua lista de eventos, definindo um novo tipo de evento interno. O instante de tempo em que o evento será tratado foi calculado na função de encaminhamento, pela fórmula $T = T_A + \frac{\|A,B\|}{V_{luz}} + T_{atraso}$.

T_A	Tempo actual de simulação no modelo A, o transmissor.
$\frac{\|A,B\|}{V_{luz}}$	Tempo de propagação: divisão da distância entre o transmissor e o receptor, sobre a velocidade do sinal no meio de propagação (velocidade da luz).
T_{atraso}	Atraso considerado na propagação do sinal.

Tabela 3.2: Simulador – Parâmetros de cálculo, do tempo de ocorrência, de um evento de recepção

A tabela acima, explica os parâmetros utilizados na fórmula anterior, para calcular o tempo em que um evento de recepção, deve ser tratado.

O parâmetro T_A , procura simular atrasos na propagação da transmissão de um sinal, como por exemplo, colisões. O valor deste parâmetro foi determinado através do comando ping. O comando permite saber, quanto tempo demoram os pacotes a chegar a um destino. Foram realizados duzentos comandos ping, entre dois nós, no final foi calculada a média desse tempo. O atraso obtido foi de três milissegundos

3.3.5 Funcionalidades

Esta secção pretende descrever as capacidades do simulador. O simulador tanto pode correr, com parâmetros padrão, bem como, com parâmetros à escolha do utilizador. Os novos valores são passados como argumentos, para o programa que corre o simulador. Os argumentos são passados na seguinte forma: <pârametro=valor>. A primeira tarefa do programa, passa por realizar o parsing, que altera, se necessário, os valores padrão.

Parâmetros	Valor Padrão
address	0x0AFE0001
dim	400x300
dist	100
file	chopin_input
log	0
length	10
nodes	11
scenario	2

Tabela 3.3: Simulador – Parâmetros de simulação

A tabela 3.3 apresenta os vários pârametros que definem o modo de execução do simulador.

O parâmetro address, representa o endereço base, dos nós da rede. Sendo o primeiro endereço para a Base Station. Para a atribuição dos outros nós, basta ir incrementando o valor da variável e atribuir o endereço aos vários nós. O valor padrão do campo, está na forma hexadecimal, representado na forma IPv4 por 10.254.0.1.

O argumento dim, representa os tamanho dos lados do rectângulo, que define os limites físicos do cenário de simulação, o local, onde os nós podem ser adicionados. O valor padrão do campo 400x300, define um rectângulo com quatrocentos metros de comprimento e trezentos metros de largura.

O parâmetro file define o ficheiro, a partir do qual, o programa de simulação, é capaz construir um novo cenário. O formato do ficheiro é o seguinte: na primeira aparece o número de nós e nas linhas seguintes, uma linha para cada nó, estarão no seguinte formato, <x y>. As variáveis x e y, representam as coordenadas cartesianas dos nós, abcissa e ordenada respectivamente.

Os campos dist, length e nodes, representam respectivamente: alcance máximo da transmissão dos nós, em metros; tempo de simulação, em segundos, para o qual, quando alcançado, se determina que não há convergência; número de nós que constituem a rede.

O argumento log, tem como função, delinear o nível de informação, oferecido pelo simulador, que o utilizador deseja. Há três níveis de registo, desde a sua inexistência, passando por um nível de registo de informação intermédio, até ao nível máximo, com o simulador a disponibilizar todos os dados gerados. A tabela 3.4 descreve os níveis de registo (logs) existentes.

Nível	Descrição
0	Nível de registo máximo, onde são guardados todos os dados do cenário e toda a actividade realizada pelos nós.
1	Torna a experiência de simulação mais rápida, já que não há registos de actividade, por parte dos nós
2	O nível de registo é nulo.

Tabela 3.4: Simulador – Níveis de logs

O registo da actividade de um nó é de fulcral importância, pois permite uma análise mais cuidada das capacidades do protocolo e também a procura de erros, através do controlo do fluxo

de mensagens protocolares. Cada nó escreve para o seu ficheiro correspodente e cada evento gera uma nova linha, com o seguinte formato <Tempo,Tipo,Beacon,Erro>.

O Tempo diz, naturalmente, respeito ao tempo actual de simulação, enquanto, o campo Erro é opcional e surge quando um beacon, não passa no processo de validação.

O campo Beacon segue o seguinte formato <Tipo Número_Sequência Origem Hop_Count Hop_Limit TLV>. Estes campos já foram explicados na secção 2.3.8 e são partes integrantes de uma Mensagem protocolar do CHOPIN, definida na figura 2.10. Este campo é opcional e é necessário quando o evento a registar é oriundo de um beacon.

O campo Tipo, define o tipo de evento, que deu origem ao registo, seja, a chegada de uma mensagem, como a expiração de temporizadores e conseguinte alteração nas estruturas de dados. Há cinco tipos de eventos e estão definidos na tabela 3.5.

Tipo	Descrição
BT	Geração de novo beacon.
BR	Retransmissão de beacon.
BA	Recepção de beacon.
LS	Remoção de entradas inválidas, da estrutura Link Set.
DS	Remoção de entradas inválidas, da estrutura Duplicate Set.

Tabela 3.5: Simulador – Tipos de logs

Os eventos do tipo BT, LS e DS acontecem por expiração dos respectivos temporizadores.

A figura 3.5 apresenta um exemplo de ficheiro de registos da actividade de um nó.

```
1  0.000000000000000,BT,BSB 0 10.254.0.1 1 254 10.254.0.1
2  0.003000333564095,BA,NB 0 10.254.0.2 1 254 10.254.0.2
3  0.006000667128190,BA,BSB 0 10.254.0.1 2 253 10.254.0.2,"Mensagem descartada - Criada por este nó"
4  2.000000000000000,BT,BSB 1 10.254.0.1 1 254 10.254.0.1
5  2.003000333564095,BA,NB 1 10.254.0.2 1 254 10.254.0.2
6  2.006000667128190,BA,BSB 1 10.254.0.1 2 253 10.254.0.2,"Mensagem descartada - Criada por este nó"
7  2.006000667128190,BA,NB 1 10.254.0.3 2 253 10.254.0.2 10.254.0.3
8  2.009001000692285,BA,NB 1 10.254.0.4 3 252 10.254.0.2 10.254.0.3 10.254.0.4
```

Figura 3.5: Simulador – Exemplo de um log

Através da análise da primeira linha, é possível verificar, que o nó é a Base Station, pois indica a geração de um novo beacon (BT) e que o tipo da mensagem é um Base Station Beacon (BSB). É a primeira mensagem criada por este nó porque o campo com o número de sequência se encontra a zero e o endereço do nó é o 10.254.0.1.

Ao se analisar a última linha, pode-se afirmar que o nó recebeu um Node Beacon (NB), do endereço 10.254.0.4, sendo a segunda mensagem criada pelo nó transmissor. É também relevante analisar o bloco TLV, de forma a localizar todo o caminho percorrido pela mensagem: 10.254.0.4 (nó gerador), 10.254.0.3 e 10.254.0.2.

Por último, a Base Station, apenas tem ao seu alcance o nó, com endereço 10.254.0.2, já que todas as mensagens, de entrada, foram provenientes desse nó, sejam de forma directa ou por retransmissão de mensagens com origem em outros nós.

O último campo, scenario, define qual o cenário a simular. Os cenários existentes encontram-se explicados na tabela 3.6.

Cenário	Descrição
1	Os nós são posicionado aleatoriamente, dentro do rectângulo, com dimensões definidas pelo parâmetro dim.
2	Os nós são posicionados em linha recta, segundo o eixo das abcissas, fazendo uma perpendicular com o eixo das ordenadas. A distância de separação dos nós é definida pelo parâmetro dist. A Base Station, encontra-se na ponta esquerda.
3	O cenário é construído pela leitura de um ficheiro. O nome do ficheiro é definido pelo parâmetro file.
4	A Base Station é posicionada no centro de um rectângulo, estando os nós móveis distribuídos nos limites.

Tabela 3.6: Simulador – Cenários de simulação

As figuras abaixo demonstram a estrutura dos cenários dois e quatro, respectivamente. A vermelho, estão representados os nós móveis, enquanto a azul, está representada a Base Station, que se encontra, em ambos os exemplos, nas coordenadas cartesianas (0;0).

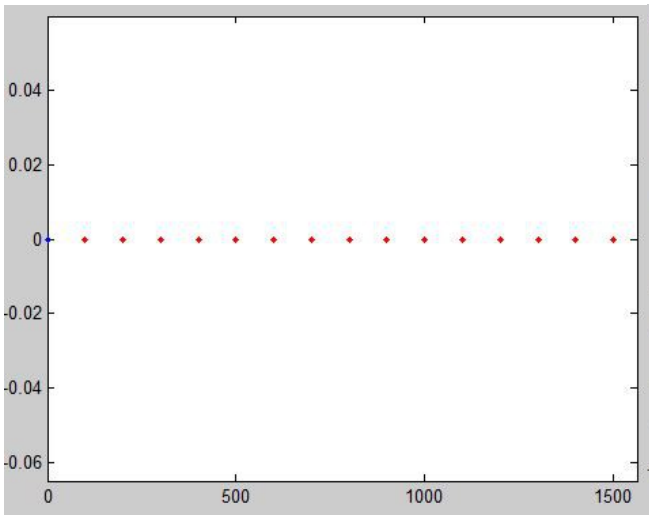

Figura 3.6: Cenário 2 – 16 nós

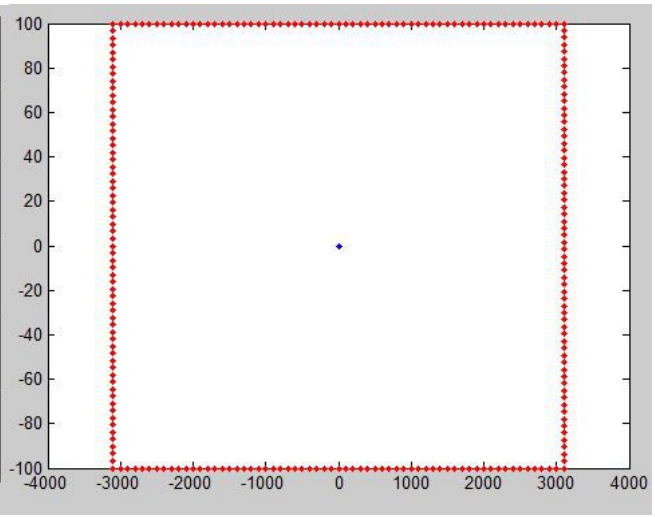

Figura 3.7: Cenário 4 – 250 nós

O simulador apresenta duas formas de condições de paragem, por convergência e por limite de tempo. Como foi dito atrás, há um limite que define o tempo máximo de simulação, cujo valor pode ser configurado. Se esse instante de tempo for alcançado, a simulação termina e portanto, é considerado que a rede não converge. A simulação termina por convergência da rede, se a Base Station, possuir caminhos para todos os nós móveis, constituintes da rede ad hoc. Essa informação é consultada na estrutura Routing Table.

0	A rede converge.
1	Há nós isolados.
2	A rede não converge.

Tabela 3.7: Simulador – Valores de retorno

Quando o simulador termina a sua execução, ele emite um valor de retorno. A correspondência entre os valores devolvidos e seu significado, está representado na tabela 3.7.

Quando há nos isolados, ou seja, há pelo menos, um nó, que, por questões de alcance de transmissão, não consegue comunicar com qualquer nó, não vale a pena iniciar a simulação já que o resultado é logo à partida, definido como não convergente.

Se, todos os nós conseguirem alcançar a Base Station, a simulação do protocolo é iniciada. No final, o programa mostra qual o tempo de simulação do cenário e a quantidade de mensagens de sinalização (beacons), que foram propagadas pela rede.

3.4 Sumário

Neste capítulo, foi feita uma introdução, ao conceito de simulação de eventos discretos. Foram apresentadas três ferramentas diferentes: Adevs, DESMO-J e SimPy.

As três ferramentas, foram avaliadas, de forma a perceber, qual delas seria a mais adequada, para desenvolvimento de uma aplicação de simulação. O simulador Adevs, foi o escolhido, para construir a tal aplicação, com o intuito de testar a performance do protocolo CHOPIN.

O capítulo prosseguiu com uma descrição da ferramenta Adevs, nomeadamente, com a explicação de modelos atómicos e de rede. Sucintamente, os modelos atómicos, representam os nós da rede e estão sujeitos a um ciclo de trabalho, com as seguintes funções para executar: avançar o tempo de simulação, função de saída, transição de estado e garbage collector. Um modelo de rede, tem a função de assegurar a comunicação, em forma de eventos, entre modelos atómicos.

O capítulo foi concluído, com uma explicação de como foi possível integrar o CHOPIN no simulador e quais as suas funcionalidades.

No próximo capítulo são apresentados e analisados, os vários testes executados com este simulador. Para além, do protocolo CHOPIN, foram ainda criadas e testadas várias versões alternativas, que foram também, submetidas a testes.

4 Experimentação com o Simulador Adevs

Este capítulo, apresenta os resultados das simulações realizadas ao protocolo CHOPIN. Ao longo do capítulo é testado o desempenho para diferentes versões do protocolo. Estas versões vão sendo apresentadas e analisadas como um aperfeiçoamento à versão anterior.

O objectivo, consiste em simular o protocolo CHOPIN em diferentes cenários e com a variação no número de nós. O simulador Adevs, descrito na secção 3.3, foi a ferramenta utilizada nesta fase. Os resultados obtidos com este simulador, não são precisos, nem próximos da realidade. Num cenário real, há todo um conjunto de factores que influenciam a comunicação entre os nós de uma rede adhoc. Presença de obstáculos físicos e colisões de pacotes são exemplos disso, com os quais este simulador não consegue lidar.

Contudo, o importante nesta fase, é adquirir uma visão geral sobre a performance do protocolo nas experiências realizadas para as diferentes versões testadas. Características como, tempo de convergência médio e o tráfego gerado, são os principais factores a considerar.

É imperativo, analisar como o estado da rede evolui com o aumento do número de nós, seja, ao nível do tempo de convergência, ou pela quantidade de beacons que se propagam pela rede.

Só após a extracção dos resultados e posterior análise, será possível definir, se o protocolo está a corresponder às expectativas, se há aspectos a melhorar e em caso afirmativo, determinar e implementar técnicas que procurem resolver os problemas encontrados.

Antes de se entrar no tema geral, como melhorar a performance do protocolo CHOPIN, este capítulo reserva uma secção, que procura mostrar, qual o menor número de nós, que garantem, conectividade total na rede, ou seja, todos os nós conseguem alcançar a Base Station.

O capítulo prosseguirá com a descrição das técnicas de aperfeiçoamento encontradas, auxiliada pela demonstração e análise dos resultados obtidos. Foram estudadas e documentadas um total de cinco versões do protocolo.

A versão 1 corresponde ao protocolo CHOPIN, descrito na secção 2.3.8. Os resultados analisados serão discutidos na secção 4.3.

Na secção 4.4, é abordado um método que procura evitar a retransmissão de pacotes, cuja a vizinhança do receptor está contida na vizinhança do transmissor. Esta técnica, tem como foco, reduzir a quantidade de Base Station Beacons, propagados pela rede.

Nas secções seguintes são descritas as restantes versões. Os métodos abordados funcionam à base do mesmo modelo, o qual, procura restringir o envio de Node Beacons, que vão no sentido da Base Station. São descritas mais três versões, pela ordem apresentada: 3, 4 e 5.

4.1 Descrição dos testes realizados

Foram realizados vários testes com uma quantidade diferente de nós e em diferentes cenários, descritos na secção 3.3.5, na tabela 3.6. Para a realização dos testes foram escolhidos os cenários um e dois. O cenário dois é específico, já que a estrutura mantém-se, sendo independente da quantidades de nós. Já no cenário um, não há uma estrutura a seguir, os nós são colocados aleatoriamente, numa área pré-definida. Aqui, há uma maior proximidade com o que se espera encontrar num ambiente real.

Para o cenário dois, foram realizadas simulações de 2-200 nós e foram obtidos os valores referentes ao tempo de convergência médio e à quantidade de sinalização gerada.

Para o cenário um, foram extraídos os mesmos tipos de dados e realizadas simulações no intervalo de 10-200 nós, com saltos de 10 em 10 nós. Como neste cenário, os nós são dispostos aleatoriamente, um teste para N nós, consiste na simulação de trinta redes adhoc, distintas. Os valores finais, são obtidos, pela média aritmética de todas as trintas simulações, $\sum_{i=1}^{30}(\mathrm{Sim}_i)/30$. Os testes foram todos realizados, com os nós enquadrados num rectângulo de dimensões 400x300 metros.

Importa referir como é realizado o cálculo do tempo de convergência médio. Consiste numa média aritmética, dos tempos de chegada de um Node Beacon, à Base Station, proveniente de um nó móvel. Para cada nó é criada uma entrada numa tabela, onde esses tempos são armazenados. Uma entrada na tabela, apenas é alterada, se o Node Beacon, oriundo do nó respectivo a essa entrada, provocar uma actualização da rota, entre o nó e a Base Station.

Como foi referido, ao simular o protocolo CHOPIN, pretende-se conhecer o tempo médio de convergência e a quantidade de sinalização gerada. Os resultados obtidos, correspondem, até ao instante de tempo, em que a rede convergiu, ou seja, até que a Base Station, conheça rotas, para todos os nós que constituem a rede.

Em cada simulação, é determinado um limite máximo de tempo, para o qual, se considera que a rede não converge. Nas simulações analisadas para o presente trabalho, esse limite de tempo, corresponde a sessenta segundos.

4.2 Teste de conectividade da rede

Nesta secção é feito um estudo sobre, qual o número mínimo de nós que garantem, num espaço físico com as dimensões de 400x300 metros, a conectividade total. Ou seja, que não existe um único nó que não consiga comunicar, directa ou indirectamente, com a Base Station. Os resultados são apresentados na figura 4.1.

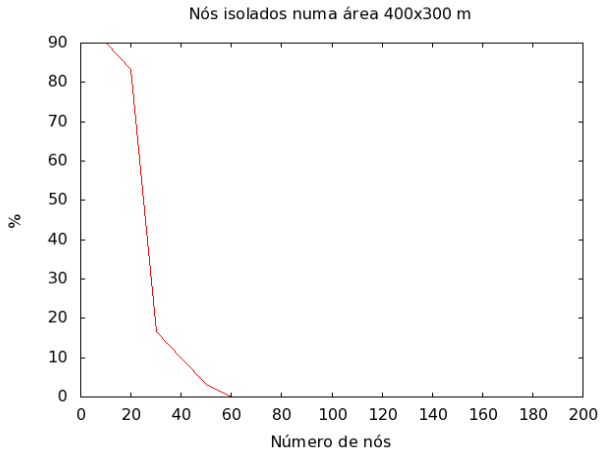

Figura 4.1: Teste à conectividade dos nós numa área de 400x300 m

Foram realizados testes no intervalo de 10-200 nós, de 10 em 10. Para cada teste com x nós, foram criadas trinta redes distintas. Foi então verificado, qual a percentagem das redes, que não garantiam conectividade total entre nós móveis e a Base Station.

Pela a análise do gráfico verifica-se que, considerando um terreno com área 400x300 metros, são necessários, no mínimo, sessenta nós, para garantir conectividade total.

4.3 Análise do protocolo CHOPIN

Nesta secção são apresentados os resultados da simulação da versõa original do protocolo CHOPIN, descrito na secção 2.3.8. Os valores, para o cenário dois, estão representados, nos gráficos das figuras 4.2 e 4.3.

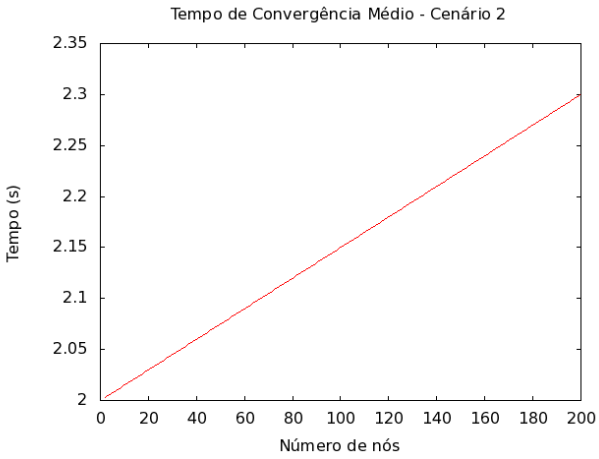

Figura 4.2: Chopin V1 – Tempo de Convergência médio, cenário 2

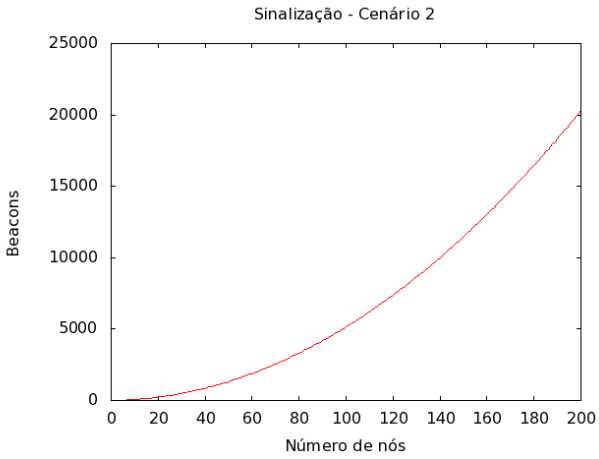

Figura 4.3: Chopin V1 – Sinalização, cenário 2

Pela a análise da figura 4.2, pode-se concluir, que o tempo de convergência médio, apresenta, com a adição de nós, um aumento linear, $O(N)$.

A figura 4.3, apresenta um aumento quadrático na propagação de beacons, com o aumento do número de nós. Analisando a propagação de Node Beacons e considerando o cenário em questão, com um caminho, até a Base Station, com N nós móveis, o nó mais distante, é responsável pela geração de, no mínimo, N beacons. Aquele que ele criou, mais os beacons retransmitidos pelos nós mais próximos à Base Station. O nó a seguir, o seu vizinho, é, no mínimo, responsável pela criação de $N-1$ beacons e o nó seguinte, por $N-2$ e por aí em diante.

O nó mais próximo, da Base Station, será responsável pela geração de apenas um beacon. Sendo assim, a geração de Node beacons. até à Base Station é representada pela seguinte fórmula $N+N-1+N-2+\ldots+1$. A análise de complexidade resulta, então, em $O(\frac{N(N+1)}{2})=O(N^2)$.

Considerando o cenário identificado, na figura 2.7 e designando o nó E, como sendo a Base

Station, fica-se com $N=4$. A quantidade de sinalização gerada, corresponde a $\frac{4*(4+1)}{2}=\frac{20}{2}=10$ Node beacons.

Analisando a propagação de Base Station Beacons, é fácil de concluir que a sinalização gerada, é de $N+1$ beacons, pois todos os nós apenas enviam uma vez, por período. Assim sendo e como seria de esperar, é a propagação de Node Beacons, que é responsável pela aparente evolução quadrática, na quantidade de sinalização gerada pelo protocolo CHOPIN.

De seguida são apresentados os testes realizados para o cenário um. Como foi referido, na secção 4.1, as simulações foram realizadas, numa área de 400x300 metros.

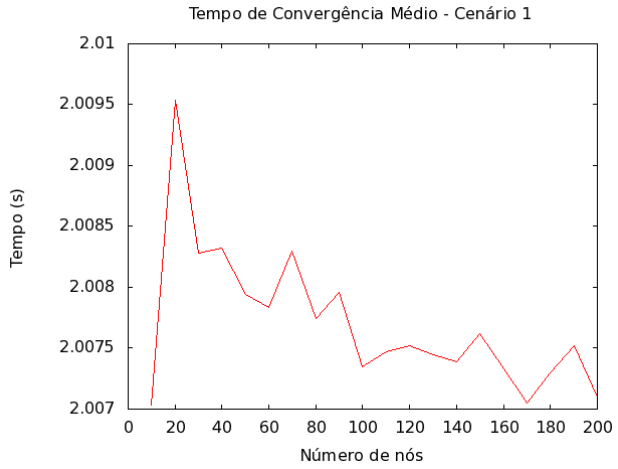

Figura 4.4: Chopin V1 – Tempo de Convergência médio, cenário 1

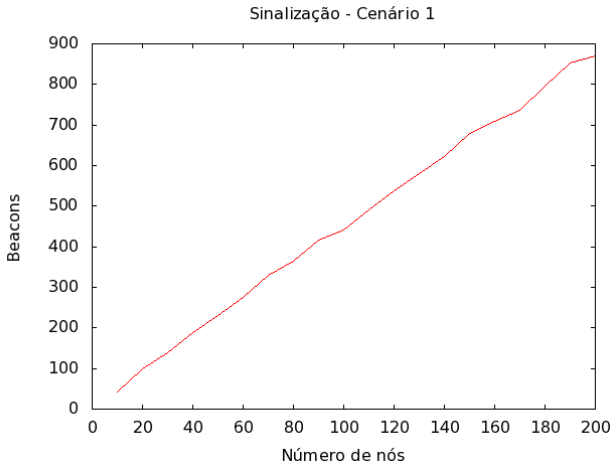

Figura 4.5: Chopin V1 – Sinalização, cenário 1

A figura 4.4, apresenta os tempos de convergência médio. É possível, verificar que o tempo de convergência, não aumenta, com o número de nós, ao contrário do que sucedeu para o cenário dois. É, desta maneira, passível de concluir, que o tempo de convergência está mais dependente de como os nós estão distribuídos geograficamente, do que propriamente, pela quantidade de nós. Nós mais dispersos e mais distantes da Base Station, demoram mais tempo a convergir.

Esta comparação torna-se mais perceptível, comparando estes resultados com os do gráfico 4.2. Nos testes anteriores, respeitantes ao cenário dois exemplificado na figura 3.6, os nós foram dispostos em linha recta, com os nós intermédios a terem dois vizinhos e os dois nós nas extremidades a terem apenas um vizinho. Num cenário destes, aparecerão nós bastante distantes da Base Station, só com um caminho disponível. Isto torna o processo de propagação de beacons mais lento.

Com os nós colocados de forma aleatória, é expectável que haja, não só, mais caminhos para a Base Station, como sejam de menor dimensão, semelhantes ao caminho apresentado no cenário dois. Assim sendo, é perfeitamente lógico que os tempos apresentados no gráfico 4.4, sejam menores do que aqueles apresentados no gráfico 4.2.

Por exemplo, voltando ao exemplo da figura 2.7 , atribuindo a Base Station ao nó E e considerando T , como o tempo de propagação de um beacon entre dois nós. O tempo de convergência será $4T$, pois é o tempo que demora, o Node Beacon do nó A, a alcançar o nó E.

Passando à análise do tempo de convergência médio, tem-se $\frac{4T+3T+2T+T}{4}$.

Repetindo a análise anterior, mas, alterando a Base Station para nó C, o centro do exemplo. Ao contrário do cenário anterior, agora há dois caminhos para a Base Station: A-B-C e E-D-C. O tempo de convergência passará a ser $2T$ e o tempo de convergência médio será $\frac{2*(2T+T)}{4}$. Comparando os valores em causa, verifica-se que, se a Base Station corresponder ao nó C, os tempos serão melhores, pois, os nós móveis estão mais próximos da Base Station.

Na análise da figura 4.5 é possível verificar que a evolução no número de beacons gerados, já não aparenta, uma forma quadrática. A mesma análise que foi feita para o tempo de convergência médio, aplica-se para a análise da quantidade de sinalização gerada. Para ambos os factores de análise, houve uma redução, do cenário dois para o cenário um.

Voltando ao exemplo da figura 2.7, se a Base Station corresponder ao nó E, são gerados 10 Node Beacons, como foi mostrado, na análise feita para o cenário dois. Se o nó C passar a representar a Base Station, o número de Node Beacons gerados irá diminuir, porque com dois caminhos distintos, em vez de se ter $N=4$, passar-se-á a ter $N=2$ para cada uma das duas rotas. Por conseguinte, neste exemplo, serão gerados $\frac{2*(2+1)}{2}=3$ Node Beacons por percurso, perfazendo um total de 6 beacons.

Após o estudo feito à versão original do protocolo CHOPIN, para dois cenários distintos, é possível apresentar algumas ideias, sobre quais os próximos passos a tomar. O foco principal, no desenvolvimento de versões futuras, está na redução dos níveis de sinalização. Como foi mostrado, o protocolo demonstra uma complexidade quadrática, na propagação de beacons, contribuindo, decisivamente, a forma como é tratada, a transmissão de Node Beacons.

Independentemente dos os resultados obtidos para o cenário um, serem significativamente melhores, não se pode ignorar a performance do protocolo, para exemplos com as características do cenário dois. Quanto menos caminhos, até à Base Station existirem e quanto maior forem esses percursos, pior será a performance do protocolo.

4.4 Detecção de Vizinhos comuns

Na secção anterior, realizaram-se as primeiras experiências, para verificar a competências do protocolo CHOPIN. Os resultados determinaram, uma rápida convergência e uma quantidade excessiva de mensagens de sinalização.

Esta secção descreve uma técnica que tem como objectivo, a redução das mensagens de controlo, Base Station Beacons. Este método, procura evitar retransmissões de beacons desnecessárias. Para tal, é necessário adicionar um novo campo a uma mensagem. Esse campo corresponderá a uma lista de vizinhos do transmissor.

Como foi descrito na secção 2.3.8, o protocolo CHOPIN, guarda na estrutura Link Set, as ligações de um um hop, que representam os vizinhos de um nó. Isto proporciona ao receptor, a comparação da sua vizinhança, com a do nó transmissor. No caso, de toda a vizinhança do nó que recebeu o beacon, estar contida na vizinhança do transmissor, torna desnecessária a retransmissão da mensagem. A vizinhança do nó corrente, estaria a receber a mensagem em duplicado.

A figura 4.6, mostra uma rede, com quatro nós, desenhada de modo a exemplificar o

conceito anterior.

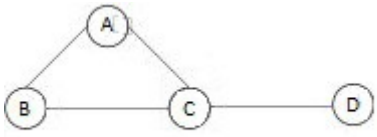

Figura 4.6: Exemplo com 4 nós

O nó B faz o broadcast de um beacon, que será recebido pelos os seus vizinhos A e C. Na mensagem vai incluída, informação, sobre a vizinhança de B, {A,C}. Neste momento, cada um dos nós A e C, terão que decidir, se devem ou não retransmitir a mensagem vinda de B. Para isso, cada um deles irá analisar os vizinhos em comum com B.

O nó C, possui a vizinhança, {A,B,D}. Dos seus vizinhos, apenas o nó A é comum a ambos, o que significa, que C deverá retransmitir a mensagem, com origem em B. Caso contrário, há a possibilidade de D nunca a vir a receber, a não ser, que haja outro caminho, B-D, que não contenha C.

O nó A, por sua vez, já não deverá retransmitir a mensagem. Ora, da intersecção das vizinhanças de A ({B,C}) e B, sai o nó C. Do ponto de vista de A, resta o nó B, que é o nó transmissor e portanto, seria um desperdício a retransmissão da mensagem.

Esta técnica, já é implementada, por outros protocolos, como por exemplo, o OLSR, com a utilização das mensagens Hello, tal como foi descrito na secção 2.3.1.

O primeiro passo, consistiu na alteração da estrutura de um beacon, exemplificado na figura 2.12.10. Como foi descrito anteriormente, é necessário a adição de um campo, que contenha, os vizinhos do nó transmissor.

Para os testes realizados, a única alteração efectuada, resulta na adição do novo campo, cuja função, já foi explicada. Em cada beacon gerado, é incluída a informação da vizinhança do nó responsável por enviar o pacote.

Não foi considerada a utilização de mensagens Hello nem a alteração ou adição de outros requisitos. As mensagens Hello, têm a função, de sinalizar os vizinhos, de cada nó. No caso do protocolo CHOPIN, isso já é conseguido, com a troca de Base Station Beacons. Como foi referido na análise à figura 2.13, que demonstra as etapas de validação de um beacon, o primeiro passo, do processamento de um beacon, após ser validado, consiste em extrair dados da mensagem, para actualizar as estruturas de dados do protocolo. Entre elas, encontra-se a estrutura Link Set, que mantém as conexões de um hop activas.

Sendo assim, em vez de aplicar as tradicionais mensagens Hello, existentes em alguns protocolos adhoc, são utilizados os próprios Base Station Beacons. As mensagens Hello, aplicadas ao protocolo CHOPIN, iriam gerar um overhead desnecessário, pois iriam aumentar a quantidade de mensagens de controlo e replicando parte do trabalho dos beacons.

Esta técnica não tem qualquer influência, ao nível da transmissão de Node Beacons, pois são propagados por unicast. É ao nível do broadcast de Base Station Beacons, que é pretendido optimizar o protocolo.

Para o cenário dois, o desempenho das duas versões é idêntico. Em virtude das características do cenário dois, esta técnica torna-se irrelevante. Neste cenário, nenhum nó está

em condições de descartar a retransmissão de pacotes. Para tal acontecer, terá sempre, de existir, um grupo de pelo menos três nós, que sejam todos, vizinhos uns dos outros. Como é fácil de observar, essa questão, não se verifica neste cenário.

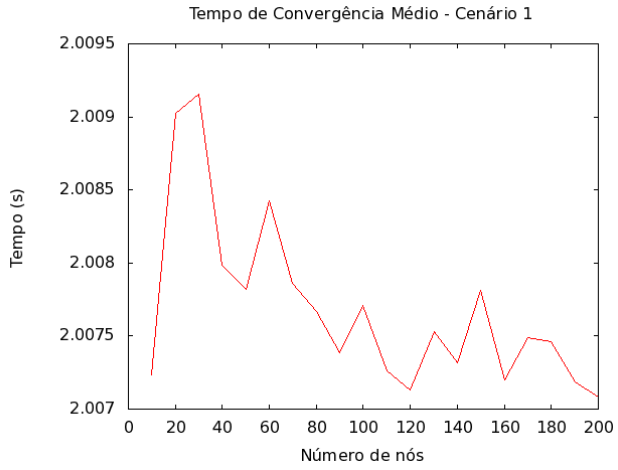

Figura 4.7: Chopin V2 – Tempo de Convergência médio, cenário 1

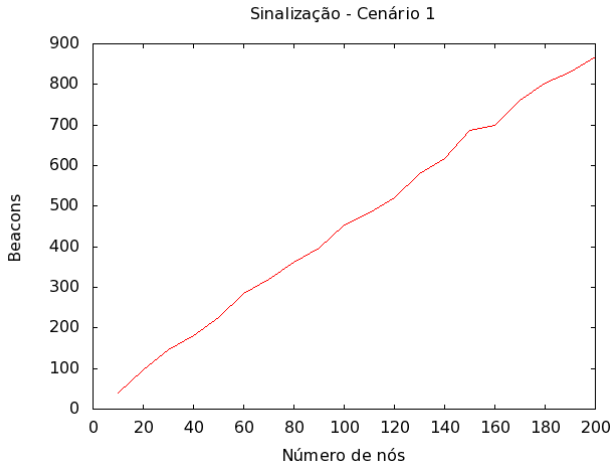

Figura 4.8: Chopin V2 – Sinalização, cenário 1

As figuras 4.7 e 4.8, representam os resultados dos testes efectuados à segunda versão do protocolo. Os testes foram realizados para o cenário um. Apresentam respectivamente, o tempo médio de convergência e a sinalização gerada. Como se verificou, nos testes anteriores e comparando com os gráficos das figuras 4.4 e 4.5, não se consegue observar melhorias significativas na performance do protocolo.

Há duas questões, que justificam estes resultados. Como foi concluído, na secção 4.3, a optimização prioritária, deverá se focar na redução de Node Beacons. O tráfego constituído por Base Station Beacons, é muito inferior.

A segunda razão, reporta-se, ao que foi referido na secção 4.1, sobre, os testes realizados terminarem após a convergência da rede. Como a figura 4.7, ajuda a demonstrar, os tempos de convergência médio, rondam os dois segundos. Ora, a optimização implementada nesta versão, só tem efeito, a partir dos dois segundos. Antes, as vizinhanças, ainda se estão a formar e portanto, não há qualquer redução na sinalização. Só após, esse instante de tempo, é iniciada a segunda difusão de Base Station Beacons.

Assim sendo, é expectável, que o protocolo CHOPIN, versão 2, apresente resultados superiores, aos do seu antecessor, se observado, durante um período de tempo superior ao evidenciado. Com uma observação a longo prazo, acontecerão mais difusões de Base Station Beacons e com as vizinhanças dos nós, já formadas. Há então, maiores probabilidades de reduzir a quantidade de sinalização propagada pela rede.

4.5 Nós de último nível

Nesta etapa foi abordado um método, que procura reduzir o tráfego de Node Beacons.

Como foi visto na secção 4.3, a grande parte, da sinalização gerada pelo protocolo CHOPIN, vai na direcção da Base Station. Esse tráfego é composto por Node Beacons. Ora, torna-se prioritário encontrar soluções, que consigam reduzir a informação gerada, pelos nós móveis, para a Base Station.

Esta nova abordagem, introduz o conceito de nível de um nó. O nível de um nó, reflecte o número de hops, a que o nó se encontra da Base Station. Com este novo conceito, surgem novas restrições: apenas nós de último nível podem iniciar a difusão de Node Beacons e os nós só podem retransmiti-los se tiverem origem em nós de níveis superiores. O nível de um nó consiste, na distância a que este, se encontra da Base Station, ou seja, o número de saltos necessários para a alcançar. Sendo assim, o nível de um nó, é definido, aquando da recepção de um Base Station Beacon, estando a tabela de encaminhamento, actualizada sempre com o caminho de menor custo.

Definir o nível de um nó, não é um novo conceito, no entanto, é crucial para encontrar e definir nós de último nível. Um nó de último nível, é aquele, que dentro de uma dada rota, se encontra mais distante da Base Station. Os nós de último nível, constituem as folhas da árvore de propagação, no sentido da Base Station, a raiz.

Para a implementação desta técnica, foram adicionados dois campos, à estrutura de um beacon: level, que representa o nível do nó, que enviou o beacon, com um tamanho de um byte; next_hop, o endereço do nó, que representa o próximo salto, até à Base Station. Para além disso, cada nó terá associado, um estado interno, que pode alterar entre, ser ou não ser, um nó de último nível.

É durante a propagação de Base Station Beacons, que um nó define o seu nível. Ao receber um Base Station Beacon, um nó, se necessário, poderá actualizar a sua tabela de encaminhamento e definir novas rotas até à Base Station. Para se saber o nível do nó, bastará consultar, qual a distância, que o separa da Base Station.

Quando o nó retransmite o Base Station Beacon, ele inclui o nível a que se encontra da Base Station, bem como, o endereço do nó, que representa o próximo salto, até à Base Station. Quando, este último nó, receber o beacon, irá comparar os níveis, entre transmissor e receptor. Se o nível do transmissor for superior, o receptor toma conhecimento, de que há um nó, ainda mais distante da Base Station. Posto isto, o receptor, já não se classifica, como nó de último nível.

Caso, o receptor do beacon, não seja aquele, identificado no campo next_hop, não há qualquer alteração ao estado interno respeitante, independentemente de se encontrar num nível inferior. Nesta situação, o transmissor e o receptor, encontram-se em caminhos diferentes, pois a propagação de Node Beacons, é realizada, em modo unicast.

Com o auxílio, do exemplo, da figura 4.9, serão exemplificados, com maior detalhe, alguns aspectos do funcionamento deste novo protocolo.

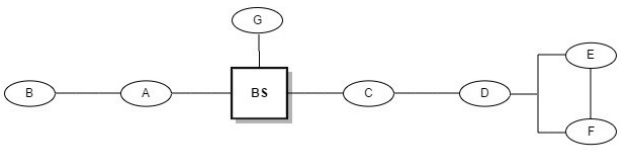

Figura 4.9: Exemplo de um cenário em árvore

É possível analisar um cenário onde se formam quatro caminhos diferentes, para a Base Station: B-A-BS, G-BS, E-D-C-BS, F-D-C-BS. É fácil de perceber que os nós B e G são nós de último nível, estando B no nível dois e G no nível um. Os nós E e F são também eles, de último nível, já que, apesar de pertencerem à mesma vizinhança, estão ambos, a três saltos da Base Station.

O nó B, após receber o Base Station Beacon, vindo do nó A, passa a saber que pertence ao nível dois, já que o beacon passou por dois nós desde a Base Station. Como não há nenhum nó com nível superior, B considera-se como o nó mais distante. Também o nó A se considera nó de ultimo nível, já que, ainda não tem conhecimento de um elemento de nível superior. Só após receber a retransmissão oriunda do nó B e verificar que este se encontra num nível acima, é que A, alterará o seu estado interno, definindo-se como um nó intermédio.

Como apenas os nós de último nível poderão iniciar a difusão das mensagens, a Base Station, ao receber um Node Beacon, terá que guardar, uma rota para todos os nós, que encaminharam o beacon. Nas versões anteriores, apenas era necessário manter uma rota para a origem do beacon, mas agora, como nem todos os nós, podem iniciar a propagação de um node beacon, isso, já não é exequível.

O estado interno, padrão, de um nó, é ser de último nível. O estado é alterado, depois de receber um Base Station Beacon, de um nó de nível superior. Para voltar ao estado inicial, pode acontecer uma de duas situações: perder a conectividade, com um nó filho, que lhe garantia, uma rota para a Base Station; não tem ligação, com qualquer nó pai; não recebeu qualquer Base Station Beacon, de um nó de nível superior, durante um certo período de tempo. Este período de tempo, é controlado, por um temporizador. Este temporizador tem a duração de $3 \times REFRESH\_INTERVAL = 6\,s$ e é reiniciado, sempre que chega um Base Station Beacon, proveniente, de um dos pais.

De seguida são demonstrados e analisados os testes realizados.

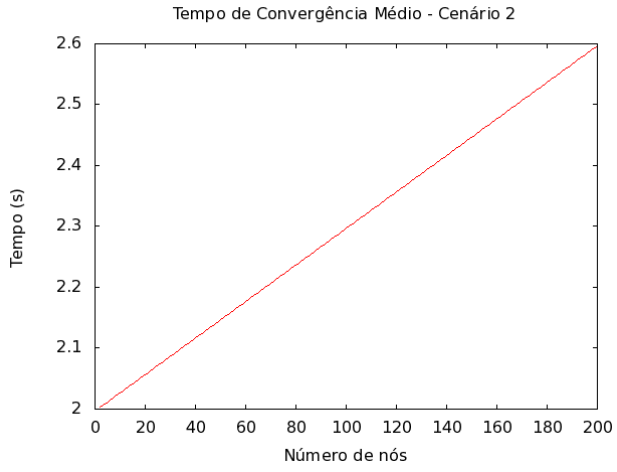

Figura 4.10: Chopin V3 – Tempo de Convergência médio, cenário 2

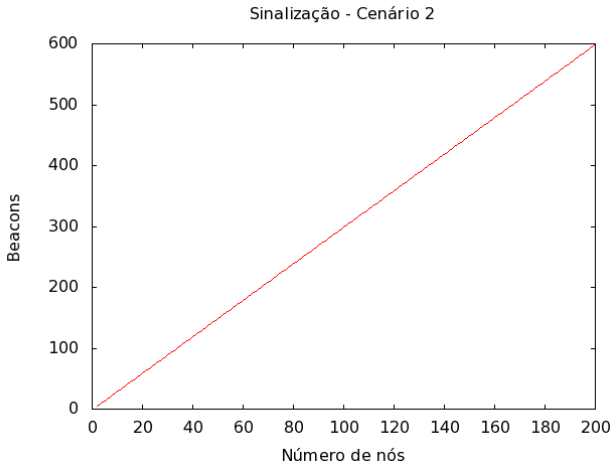

Figura 4.11: Chopin V3 – Sinalização, cenário 2

A figura 4.10, apresenta os tempos de convergência média, para o cenário dois, utilizando o protocolo CHOPIN V3. Os tempos obtidos, são ligeiramente superiores, se comparados, com os

obtidos nas versões anteriores. Isto acontece, porque os nós mais perto da Base Station, têm que esperar, pelos nós mais distantes, para enviarem beacons. Todo um caminho, é adicionado à Base Station, em simultâneo, estando os tempos de chegada, dos nós menos longínquos, dependentes dos nós superiores.

A figura 4.11, mostra a quantidade de beacons gerados. Esta versão, conseguiu, resultados bastante superiores, em comparação, com as versões anteriores. Para cada rota, um nó apenas gera um beacon: a folha do ramo da árvore gera o beacon e os nós móveis restantes, reencaminham-no até à Base Station.

Nas versões anteriores, cada nó móvel gerava o seu próprio beacon e para além disso, reencaminhavam os beacons de nós antecessores. Como foi visto, na secção 4.3, onde foi feita a análise à versão original, do protocolo CHOPIN, existia uma complexidade quadrática, $O(\frac{N(N+1)}{2})$.

Fazendo a mesma análise para a versão três e sendo N , o número de nós móveis de um caminho, são gerados N Node Beacons.

Os resultados permitem concluir, que o protocolo CHOPIN, melhorou substancialmente a sua performance, conseguindo, uma redução de sinalização, aproximadamente de 97%. Outro motivo de satisfação, é o facto, do cenário dois, já não apresentar ser, o calcanhar de aquiles do protocolo CHOPIN, no que ao nível da sinalização diz respeito.

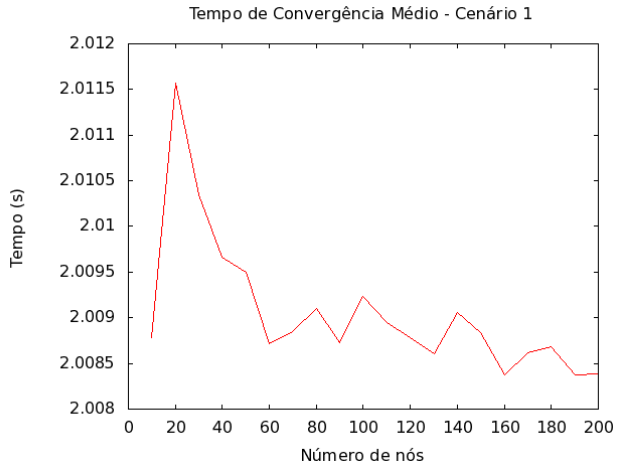

Figura 4.12: Chopin V3 – Tempo de Convergência médio, cenário 1

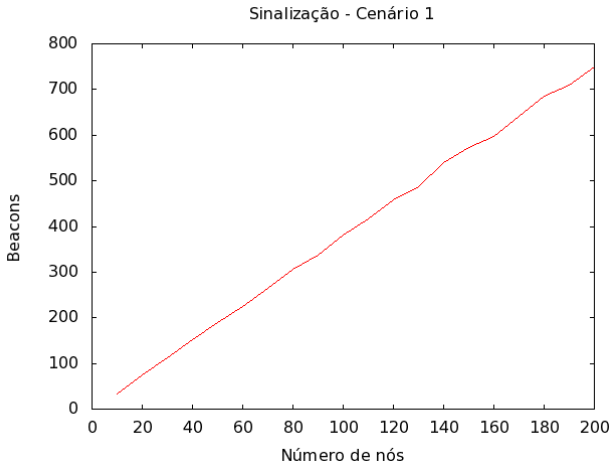

Figura 4.13: Chopin V3 – Sinalização, cenário 1

As figuras 4.12 e 4.13, apresentam os resultados dos testes efectuados à versão três, para o cenário um.

Em relação aos tempos de convergência média, os resultados são ligeiramente superiores, aos tempos das versões anteriores, sendo ligeiramente superiores, tal como foi verificado, para o cenário dois. É a consequência de aparecerem, mais caminhos e de menor dimensão.

Na quantidade de mensagens geradas, há uma ligeira redução, comparativamente às versões anteriores.

Ao contrário, do que se verificou, em experiências anteriores, são observados níveis de

sinalização inferiores, para o cenário dois. Isto acontece, porque no cenário um, podem existir nós, que pertençam a mais do que um caminho e portanto, retransmitem mais beacons. Neste cenário, a quantidade de beacons gerados, corresponde à soma, do comprimento de todos os caminhos existentes. É representada pela fórmula $\sum_{i}^{i \in S} C(i)$: S é o conjunto de nós de último nível; $C(i)$ é uma função, que permite obter o comprimento de um caminho, com origem no nó i .

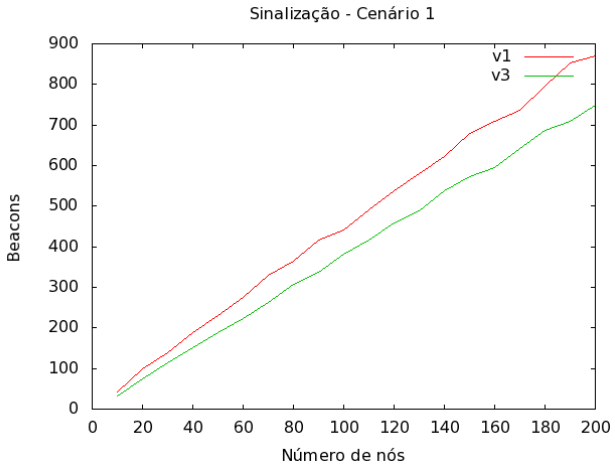

Figura 4.14: Comparação nos níveis de sinalização, com as versões 1 e 3

A figura 4.14, mostra os níveis de sinalização para as versões um e três, no cenário um. Observando o gráfico, verifica-se a melhoria, de desempenho, ganha com a versão três. Com o aumento do número de nós, as diferenças na performance dos protocolos, vai se tornando mais acentuada.

4.6 Envio de múltiplos caminhos

Nesta secção é descrita uma abordagem, que reflecte os mesmos princípios da versão três, descrita na secção 4.5, introduzindo uma nova ideia ao protocolo. A nova actualização permite aos nós móveis, o envio de vários caminhos, num único Node Beacon. A figura 4.15 exemplifica um cenário, onde este método pode ser útil.

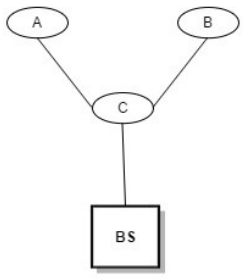

Figura 4.15: Exemplo de um cenário com uma bifurcação

O exemplo mostra uma rede com quatro nós: a Base Station; o nó intermédio C, que se situa no nível um; os nós de último nível A e B, ambos no nível dois.

Na versão 3, considerando este exemplo e a propagação de Node Beacons, são gerados quatro pacotes. A e B, são responsáveis pelo início da difusão de beacons, ou seja, são gerados dois beacons, um por cada nó. Por sua vez, o nó C tem que reencaminhar os pacotes de cada um dos nós superiores, logo, é responsável pelo envio de dois beacons.

Com o método implementado na versão 4 é possível reduzir a quantidade de Node Beacons transmitidos. A ideia consiste em um nó intermédio atrasar a retransmissão de beacons, de forma a conseguir, que num só beacon, vá mais do que um caminho.

Nesta versão, os nós intermédios, voltam a gerar os seus próprios beacons, capacidade perdida na versão anterior, onde apenas podiam realizar retransmissões. Desta vez, este tipo de nós, deixa de realizar retransmissões de Node Beacons. Cada nó gera o seu próprio beacon após a expiração do temporizador. Em cada beacon gerado, serão incorporados, todos os caminhos, que o nó transmissor foi recebendo.

Foi necessário adicionar uma nova estrutura de dados para armazenar os caminhos que vão chegando. Na altura do envio, a lista de caminhos é copiada para o beacon. Após o envio da mensagem, todo o conteúdo, da lista de caminhos, é eliminado.

Voltando ao exemplo da figura 4.15, com a nova versão, são precisos três beacons, contrastando com os quatro da versão 3. Desta vez, o nó C, receberá os beacons de A e B. Quando o seu temporizador expirar, enviará o beacon com os caminhos com origem em A e B.

Os resultados, das simulações realizadas, são apresentadas, de seguida.

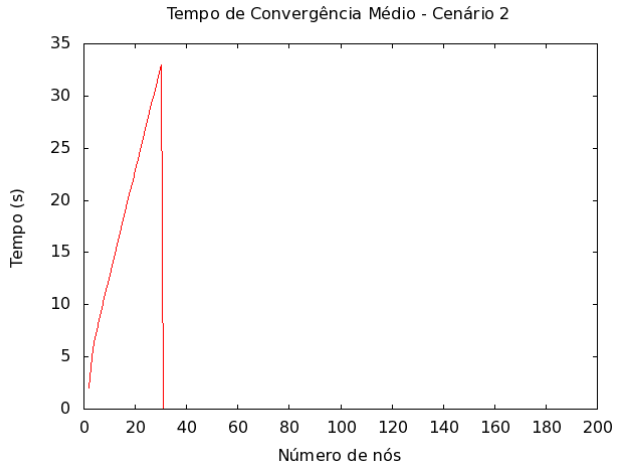

Figura 4.16: Chopin V4 – Tempo de Convergência médio, cenário 2

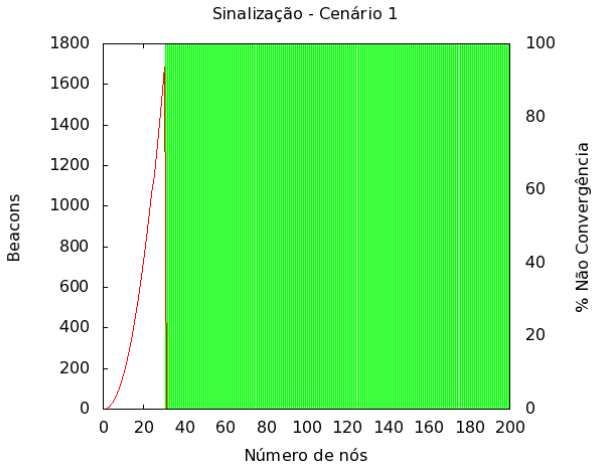

Figura 4.17: CHOPIN V4 – Sinalização, cenário 1

A figura 4.16, apresenta os tempos de convergência médio, para a versão quatro, cenário dois. Na maior parte dos testes, os tempos de convergência, são nulos. Significa, que, para as

respectivas simulações, não houve convergência, ou seja, foi ultrapassado o limite de sessenta segundos. A partir dos trinta nós, a rede deixou de convergir. Já nos casos em que a rede convergiu, os tempos verificados, são largamente superiores, àqueles observados nos protocolos anteriores.

A figura 4.17, mostra os níveis de sinalização. Aqui, a análise tem que ser separada em dois momentos. Quando há convergência e quando não há. Os testes de convergência, estão representados, por uma curva quadrática. A fase sem convergência, está representada, com valores nulos, tal e qual, como foi feito para o gráfico da figura 4.16.

Observa-se um retrocesso, pois na versão anterior, o tráfego gerado foi bastante inferior. No entanto, é necessário ter em conta, que os testes da versão quatro, foram realizados durante um maior período de tempo. Muitos dos testes, atingiram os sessenta segundos, enquanto, que nos protocolos anteriores, os testes rondam os dois segundos.

Cada nó intermédio de um caminho, armazena todos os caminhos que vai recebendo, enviando-os, no momento em que o seu temporizador indicar. É esta espera, que gera estes péssimos tempos de convergência e, consequentemente, provoca um enorme aumento na sinalização. Numa topologia de rede como a do cenário dois, o desempenho do protocolo revela-se pouco eficiente.

Quanto maior forem os caminhos numa rede, maior o tempo de espera, pois um Node Beaocn, terá que esperar pelo tempo de envio do nó que o recebeu. Esse tempo está sempre limitado inferiormente pela seguinte fórmula $(N-2) * T_{temporizador}\ (s)$, sendo N , o número de nós e $T_{temporizador}$ tempo de intervalo do temporizador responsável pela transmissão de pacotes. Ao número de nós tem que se retirar a Base Station e o nó de último nível que iniciou a difusão do beacon, depois basta multiplicar o valor da duração do temporizador por cada um dos restantes nós do caminho.

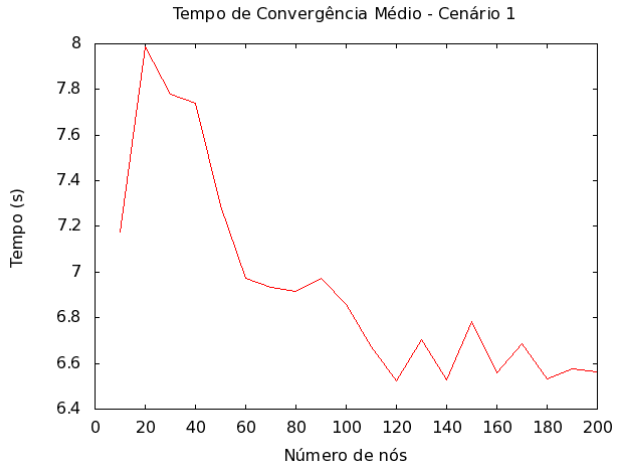

Figura 4.18: Chopin v4 – Tempo de Convergência médio, cenário 1

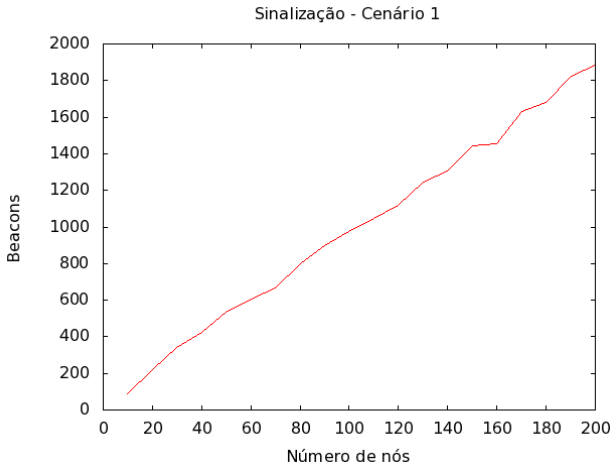

Figura 4.19: Chopin V4 – Sinalização, cenário 1

Na figura 4.18, estão os tempos de convergência médios para o cenário um. Os resultados seguem a tendência verificada, nos testes para o cenário dois. O cenário um, apresenta topologias com menos caminhos e mais curtos, resultando em convergências mais céleres. No entanto, os

resultados provam, que uma rede gerida por este protocolo, converge mais lentamente, do que em todos os protocolos anteriores.

Com a diminuição dos tempos de convergência, há uma redução na sinalização. Como mostra a figura 4.19. Contudo, os valores apresentados, continuam a ser piores, se comparados com as versões anteriores.

Em nenhuma das versões anteriores, foram apresentados estes números exorbitantes. Verifica-se que a versão 4 do protocolo CHOPIN é impraticável quando as redes apresentam caminhos bastante longos. O mesmo se aplica para cenários de rede menos densos.

4.7 Estrutura de dados – HigherLinkSet

Esta versão, procura ultrapassar os problemas apresentados na versão 4 do protocolo. Com solução criada, um nó intermédio, já não está dependente do seu temporizador, para enviar o Node Beacon com todos os caminhos pendentes. Agora, a partir do instante, em que um nó, recebe os Node Beacons de todos os seus pais, pode juntar todos os caminhos retidos num beacon e enviá-lo antes do temporizador expirar.

Para tal, um nó tem que saber quais os nós vizinhos, que estão num nível superior. Sabendo isso, mal tenha recebido todos os beacons desses nós, estará em condições de enviar o seu. Desta forma, escusa-se a esperar pelo fim do temporizador e pode de imediato reprogramá-lo para o próximo intervalo de tempo.

Para armazenar a informação necessária dos vizinhos que têm nível superior, foi criada uma nova estrutura de dados para o protocolo CHOPIN, à qual se chamou Higher Link Set. Esta estrutura é representada por uma hashtable. A chave de acesso é calculada utilizando uma função de hashing do endereço do vizinho. Na estrutura são armazenados os seguintes campos, que estão identificados na tabela 4.1: o endereço do vizinho; o nível a que se encontra o vizinho.

Neighbor Address	Neighbor Level

Tabela 4.1: CHOPIN – Higher Link Set

Se, após o temporizador expirar, um nó ainda não tiver recebido beacons de todos os nós que estão na estrutura Higher Link Set, ele gera o beacon com a informação recebida e envia o Node Beacon. Se um vizinho saiu do alcance, um nó não tem que ficar à espera indefinidamente.

De seguida são apresentados os testes realizados para o cenário um. Não é necessário testar a performance do protocolo para o cenário dois. Para esse tipo de topologias, o desempenho é parecido à versão três, tendo os resultados, sido apresentados, na secção 4.5.

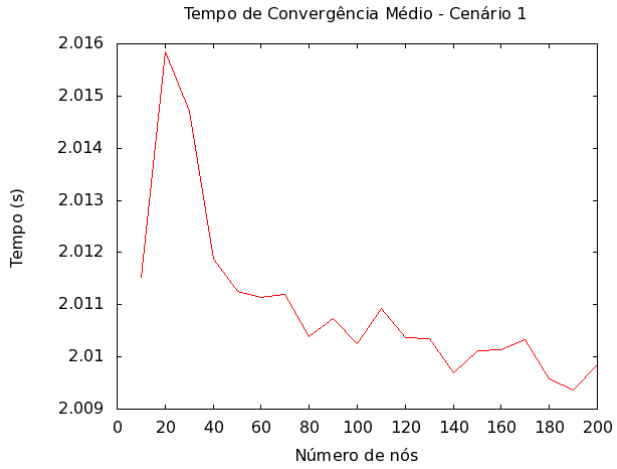

Figura 4.20: CHOPIN V5 – Tempo de Convergência médio, cenário 1

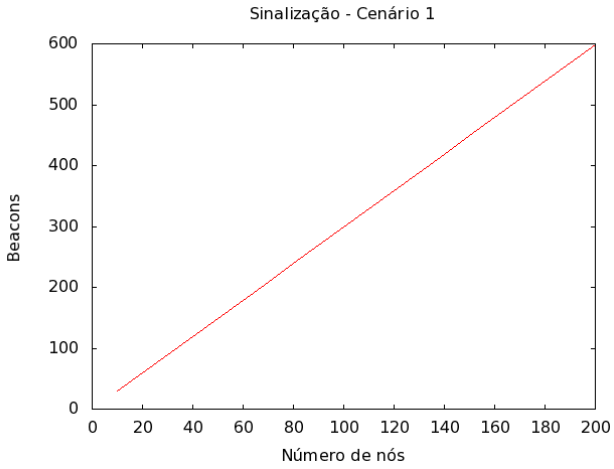

Figura 4.21: CHOPIN V5 – Sinalização, cenário 1

Os tempos de convergência no gráfico da figura 4.20, são melhores que os da versão anterior e adequam-se, aos valores verificados em outras versões.

É nos níveis de sinalização que esta versão demonstra a sua eficiência. Apresenta uma evolução de complexidade linear, com um tráfego bastante reduzido e inferior a todos protocolos já testados, como mostra a figura 4.22.

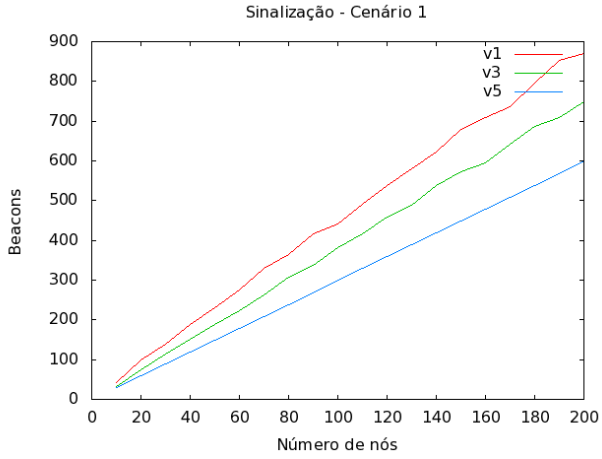

Figura 4.22: Comparação nos níveis de sinalização, com as versões 1, 3 e 5

4.8 Sumário

Ao longo deste capítulo, foram analisados diversos protocolos, o protocolo CHOPIN e mais quatro versões. Os novos protocolos, foram criados com a intenção de reduzir o tráfego gerado pela versão um. Os protocolos foram testados com o simulador apresentado no capítulo 3. Foram definidos dois cenários de simulação: cenário um, representado por redes com uma distribuição

aleatório, dentro do espaço definido e o cenário dois, no qual os nós estão separados em linha recta, a uma distância, que equivale ao alcance máximo de transmissão.

A primeira análise realizada, consistiu em verificar, a partir de que quantidade de nós, para um cenário com dimensões, 400x300 m, há garantia de conectividade total. A partir dos sessenta nós, em todas as redes geradas, não apareceu qualquer nó isolado.

O passo seguinte, consistiu em testar o desempenho do protocolo CHOPIN. Foi avaliado com base, em duas variáveis: tempo de convergência médio e quantidade de beacons propagados pela rede. É dado maior ênfase a este último.

A performance do protocolo, foi repartida em duas fases: cenário um e dois. Para o cenário dois, foi gerado uma quantidade de sinalização, bastante superior, à verificada no cenário um. Foi concluído, que desempenho do protocolo piora, em redes com caminhos grandes e, que a maior parte do tráfego, é constituído por Node Beacons.

De seguida, foi analisado o CHOPIN V2. Esta versão, visa a redução de Base Station Beacons, caso a vizinhança do receptor não esteja contida na vizinhnça do trasmissor. Os resultados obtidos, não foram muito diferentes, dos apresentados para a versão um. Por um lado, o tráfego de Base Station Beacons, é muito inferior, ao de Node Beacons e por outro, a rede converge, antes da formação de vizinhanças.

As três versões seguintes, centram-se no mesmo conceito, nós de último nível. Este tipo de nós, não têm qualquer outro nó móvel, mais distante da Base Station, do que eles e são responsáveis por iniciarem a difusão de Node Beacons.

No CHOPIN V3, são os únicos que podem gerar Node Beacons, todos os outros, apenas podem retransmitir. Neste caso, os resultados foram melhores no cenário dois, porque é formado, por um único caminho e nó apenas tem de fazer uma retransmissão por cada difusão.

O CHOPIN V4, procura resolver esse problema, permitindo que, nós intermédios, retenham caminhos que foram chegando, para posteriormente os enviarem todos de uma vez. Este protocolo não apresentou bons resultados. O envio de um caminho, é retardado, a cada nível que passa. Para redes, com caminhos muito grandes, como o do cenário dois, este protocolo é impraticável.

O CHOPIN V5, vem para resolver o problema identificado na versão anterior. Um nó, passa a ter conhecimento de quais os seus vizinhos, lhe vão enviar Node Beacons. Quando todos os beacons chegarem, o nó pode enviar um Node Beacon, com todos os caminhos acumulados. Esta versão foi aquela que apresentou os melhores resultados, em ambos os cenários estudados.

No próximo capítulo, é apresentada, a ferramenta de simulação de redes, o NS-3.

5 Simulador NS-3

O NS-3[9], Network Simulator 3, é uma ferramenta de simulação de eventos discretos, para redes informáticas. Oferece um ambiente de simulação para estudo e desenvolvimento de projectos de redes.

No NS-3, é necessário configurar vários componentes de rede, para que a simulação possa correr sem problemas:

- Configurar a camada de acesso ao meio físico;
- Criar um canal de comunicação e atribuir-lhe um modelo de propagação de erros e outro, de perdas;
- Atribuir um canal de comunicação, às interfaces de rede;
- Definição do protocolo,sem fios, a utilizar e definir suas propriedades;
- Instalar tudo o que foi dito atrás nos nós.
- Estabelecer qual o o protocolo de encaminhamento a utilizar;
- Atribuir endereços Ipv4;

5.1 Instalação

O NS-3 é distribuído, como código fonte, desta forma, tem que se assegurar, que o sistema de hospedagem, está configurado e possui todas as dependências, para a correcta instalação do simulador.

Como foi referido, o NS-3, tem uma enorme lista de dependências. Essa lista está disponível on-line, bem como, o manual de instalação[10]. Seguindo todos os passos, aí descritos, a instalação, torna-nse bastante mais facilitada.

O código fonte[11], está disponível, num repositório, para download, onde só é necessário, escolher a versão do simulador pretendida.

5.2 Módulos

Um módulo, agrupa um conjunto de classes, exemplos e testes. É onde se coloca todo o código necessário para correr a simulação.

Para criar um novo módulo, é necessário ir à dirctoria *src* e correr o comando “create-module.py <nome_do_módulo>”.

5.3 Funcionalidades

O simulador NS-3, possui, paraticamente, as mesmas funcionalidades, que o simulador Adevs, capítulo 3.3.5. A única diferença, é que neste simulador foi implementado um detector de perdas.

O processo de detecção funciona da seguinte forma:

- Cada nó tem um lista de pacotes;
- Quando um nó, envia um pacote, esse pacote é colocado na lista de todos os nós, ao alcance do transmissor;
- Quando um nó, recebe um pacote, vai à sua lista, e remove o pacote correspondente;
- É definido um período, para o qual, todas as listas dos nós, são consultadas. Os pacotes que ainda não foram retirados, significa, que eles podem ter se perdido. Um pacote é considerado perdido, se a diferença, entre o tempo actual e o tempo de quando o pacote foi colocado na lista, for superior a um determinado valor. Em caso afirmativo é contabilizada, como perda e o pacote é removido da lista.

5.4 Sumário

Este capítulo permitiu fazer uma breve referência, ao simulador NS-3. Foram abordados temas sobre: instalação do NS-3 e de módulos do NS-3.

Foi ainda discutido, como trabalha o gestor de perdas do simulador.

O próximo capítulo prosseguirá com o NS-3, com o estudo dos diferentes protocolos.

6 Experimentação com o Simulador NS-3

Neste capítulo, é pretendido testar os protocolos, estudados no capítulo anterior, num simulador diferente. Pretende-se observar, o desempenho dos protocolos, num ambiente, mais próximo da realidade. O simulador escolhido para esta fase de testes, foi o NS-3.

O NS-3, ao contrário do Adevs, foi desenhado especificamente, para simular sistemas de redes informáticas. Antes de se iniciar o processo de simulação, é necessário configurar os vários componentes de rede, para as diversas camadas protocolares: definir características da camada de acesso ao meio físico; criar um canal de comunicação e definir, por exemplo, modelos de propagação de atraso e perdas; criação e atribuição de interfaces de rede e associação com um canal de comunicação; definição de protocolos de encaminhamento e transporte; criação de uma pilha de endereços Ipv4 ou Ipv6. É um simulador extremamente poderoso, pois disponibiliza ao programador, as ferramentas necessárias, para testar o desempenho de uma rede. Desta forma, é bastante útil, para o desenvolvimento e teste de protocolos, para as várias camadas de funcionamento de uma rede informática.

Depois de tudo isto, há ainda, a destacar, a capacidade desta ferramenta, em detectar colisão de pacotes. Os resultados apresentados pelo simulador Adevs, não contemplam esta questão. Com o NS-3, foi possível simular as várias versões do protocolo CHOPIN, num ambiente mais próximo da realidade e, com isso, obter resultados mais rigorosos.

O capítulo começa por identificar o problema da colisão pacotes e quais os efeitos no desempenho do protocolo. É também explicado, qual a solução encontrada, para ultrapassar o problema.

De seguida, são testados , todos os protocolos já criados.

6.1 Descrição dos testes realizados

Foram realizados semelhantes, aos realizados no capítulo 4, referente à simulação do protocolo com o simulador Adevs. Os cenários de simulação um e dois, foram, novamente escolhidos. As mesmas características foram obtidas, tempo de convergência médio, sinalização gerada e percentagem de de testes que não convergiram. Os testes foram todos realizados, com os nós enquadrados num rectângulo de dimensões 400x300 metros.

Para o cenário dois, foram realizadas simulações de 10-200 nós, com saltos de 10 em 10 nós. Para cada teste com N nós, foram realizadas dez simulações. Apesar de este cenário apresentar sempre o mesmo formato, neste capítulo foi inserido um atraso variável, no envio dos beacons. Este atraso, tem como objectivo reduzir o número de colisões. Será abordado com maior pormenor ao longo do capítulo.

Para o cenário um, foram realizadas simulações no intervalo de 10-200 nós, com salts de 10 em 10 nós. Como neste cenário, os nós são dispostos aleatoriamente, para cada teste com N nós,são realizadas trinta simulações.

Como foi referido, ao simular o protocolo CHOPIN, pretende-se conhecer o tempo médio de convergência e a quantidade de sinalização gerada. Os resultados obtidos, correspondem, até ao instante de tempo, em que a rede convergiu. Como podem surgir redes, que não convirjam, seja por defeito dos protocolos, ou por serem muito densas, é apresentado, num gráfico de barras,

a probabilidade de não convergência. Dos valores apresentados, apenas são contabilizados, quando há convergência.

Dependendo da versão e cenário de teste, foi necessário ajustar o temporizador para geração de beacons e configurar diferentes atrasos no envio de pacotes. Como é explicado nas secções seguintes, cada protocolo precisará de diferentes configurações, para conseguirem o desempenho mais optimizado possível.

Em cada simulação, é determinado um limite máximo de tempo, para o qual, se considera que a rede não converge. Nas simulações analisadas para o presente trabalho, esse limite de tempo, corresponde a sessenta segundos.

6.2 Colisões

O NS-3 permite testar o comportamento do protocolo CHOPIN, num ambiente próximo da realidade. Nos testes realizados no capítulo 4, o simulador em questão, não oferecia essa garantia. Os testes foram executados num cenário ideal, onde não existe colisão de pacotes.

Num ambiente real, não é possível ignorar essa possibilidade, pois qualquer rede está exposta à ocorrência de colisões. Quando um nó recebe dois pacotes em simultâneo, ocorre uma colisão de pacotes. Esta situação, resulta na interferência mútua dos sinais de rádio que transportam a informação e corrompendo esses dados.

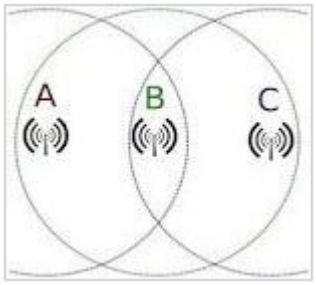

Figura 6.1: Hidden Terminal Problem

De acordo com a figura 6.1, os nós A e C não estão ao alcance, um do outro e ambos, conseguem alcançar B. Se A e C começarem a transmitir informação em simultâneo, haverá colisão dos pacotes, ao chegarem a B. Ao contrário de B, os restantes nós, não são capazes de detectar a colisão. Como A e C, só conseguem comunicar, entre si, se o nó B proceder ao encaminhamento dos seus pacotes, não se conseguem aperceber da presença do outro nó. Este contratempo é conhecido por, Hidden Terminal Problem.

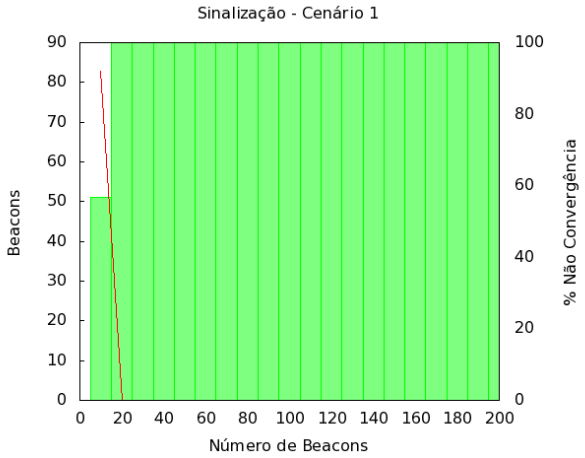

Figura 6.2: Teste de colisões, CHOPIN V1

A figura 6.2, mostra o desempenho da versão original, do protocolo CHOPIN, no simulador NS-3. Como é possível verificar, a partir, dos vinte nós, há uma probabilidade de 100%, de a rede não convergir, enquanto que, para, apenas, dez nós, se observa uma percentagem, próxima dos 55%.

Isto resulta, da propagação de um imenso número de mensagens, num curto espaço de tempo, sem qualquer sincronização, por parte dos nós, no envio de pacotes. Esta questão, gera um excessivo número de colisões e impede que os nós mais distantes, consigam comunicar com a Base Station. Os nós mais próximos à Base Station, servem de encaminhadores para os nós mais distantes. Ora, se um dos nós inferiores, está constantemente, a descartar pacotes corrompidos, devido a colisões e não conseguirá encaminhar beacons, de nós superiores. No fundo, é como se existisse uma muralha, bloqueadora de qualquer sinal, que tente atravessar.

Para corrigir este problema, foi necessário encontrar uma solução, que acima de tudo, não originasse um aumento no número de mensagens, mesmo que, como consequência, os tempos de convergência piorassem.

A solução encontrada, consiste em inserir um atraso, no momento do envio das mensagens. O valor deste atraso, varia a cada envio e é calculado aleatoriamente sem nunca exceder um limite máximo imposto. Esta técnica, é especialmente útil, para zonas na rede, muito densas, em que todos os nós conseguem comunicar entre si. Numa situação destas, um nó terá que esperar, antes de poder retransmitir um beacon e evitar conflitos, com os nós da vizinhança. Desta forma, a ocorrência de colisões será menos provável de acontecer, pois é mais improvável, que os tempos de transmissão, de nós vizinhos coincidam.

Esta solução alternativa, requer a adição de duas variáveis ao protocolo CHOPIN: uma lista, onde são armazenadas todas as mensagens, que estão à espera para serem enviadas; um temporizador que controlará o tempo de acesso à lista, para proceder ao envio das mensagens em espera. A lista, é organizada de forma crescente, pelo tempo de envio. O processo, despoletado pelo temporizador, apenas terá que ir buscar a mensagem ao início da lista.

Há, a ressalvar. que o facto de se introduzir um atraso no envio de mensagens, em cada nó, irá gerar um aumento considerável, no tempo de propagação dos beacons e consequentemente, aumentar o tempo de convergência da rede. Por exemplo, se considerarmos um atraso $1.5s$ e

um caminho com comprimento $6 nós$, até à Base Station. O atraso na propagação, está limitado superiormente, por $1.5 \times 6 = 9 s$, pois no máximo há um atraso de um segundo e meio, em cada nó. No entanto, não expectável que todos os nós apresentem valores próximos ao limite máximo. Pode-se considerar, que em média, o atraso, por nó, é de $0.75 s$, resultando num atraso médio total, de $0.75 \times 6 = 4.5 s$.

Numa rede, com nós distribuídos de forma semelhante ao verificado no cenário dois, o atraso na transmissão de pacotes, tem que ser configurado com um valor mais curto. A razão disto, prende-se em dois factores: num cenário destes, está espera-se que os caminhos sejam bem mais compridos e que os nós estejam mais isolados, ou seja, menos vizinhos para interferir nas comunicações com outros nós. Como já foi visto, com o tempo de propagação de beacons, num dado caminho, é influenciado, pelo valor do atraso. A juntar a isto, como a rede é menos densa, não há necessidade de atribuir um valor demasiado elevado, pois há menos probabilidades de ocorrerem colisões.

Posto isto, para o cenário dois, não é necessário configurar o atraso, com um valor demasiado alto, por duas razões que, como se viu, acabam por se conjugar: caminhos grandes e redes pouco densas. Pelo contrário, no cenário um, verifica-se o oposto. A forma como os nós estão distribuídos, obriga a que seja atribuído um valor superior para tempo de atraso no envio de pacotes. Como é fácil de entender, as razões são, exactamente opostas, ao cenário anterior. As redes são mais densas, havendo vizinhanças bastante grandes, com um valor no atraso suficientemente grande, é possível criar mais espaços temporais, para que cada nó possa enviar o seu beacon sem interferir com o envio do vizinho. O facto de os caminhos serem mais curtos, não cria qualquer impedimento, em que o atraso seja configurado com um valor superior.

Para uma rede móvel Ad Hoc, é crucial, que se consiga adaptar às diferentes alterações na topologia da rede. Se num momento, a rede pode apresentar um aspecto semelhante ao cenários dois, mais tarde, já poderá estar mais próxima do aspecto apresentado no cenário um. Se no primeiro momento, o atraso terá que ser menor e ao evoluir, para uma rede mais densa, ir aumentando esse valor, para diminuir as perdas de pacotes que possam surgir.

Nas próximas secções, serão analisados, os testes realizados pelo NS-3, ao protocolo CHOPIN, original e restantes versões, introduzidas e analisadas durante o capítulo 4. De forma a obter os melhores resultados possíveis, os vários protocolos foram testados com diferentes configurações. Cada teste foi alternando o tempo máximo no atraso de envio e no intervalo de tempo entre cada geração. De forma a escolher a melhor configuração possível, foram focadas as seguintes características: baixo nível de sinalização, baixa quantidade de colisões e frequência com que houve convergência.

O tempo de convergência poderá ser prejudicado, no entanto, a prioridade está em atingir valores de sinalização o mais reduzidos possível. Assim, haverá, uma maior aproximação, na procura do compromisso do tempo de convergência médio e sinalização gerada, como aliás é o objectivo do trabalho.

6.3 Protocolo CHOPIN V1

Esta secção continua o estudo da versão original do protocolo CHOPIN. Procura complementar o estudo feito na secção 4.3. Desta vez, o protocolo, foi submetido a testes, pelo simulador NS-3. As configurações de teste estão apresentadas, na tabela 6.1.

	Cenário 1	Cenário 2
Atraso máximo	1.5 s	0.1 s
REFRESH_INTERVAL	2 s	2 s

Tabela 6.1: NS-3, CHOPIN V1 - parâmetros de configuração

O intervalo de tempo, entre a geração de beacons, mantém-se nos dois segundos, em ambas os cenários. Apenas foi adicionado um tempo de atraso para o envio de beacons, como foi explicado na secção 6.2. Tal como foi explicado, no cenário dois, o tempo de atraso é inferior, ao tempo encontrado no cenário um.

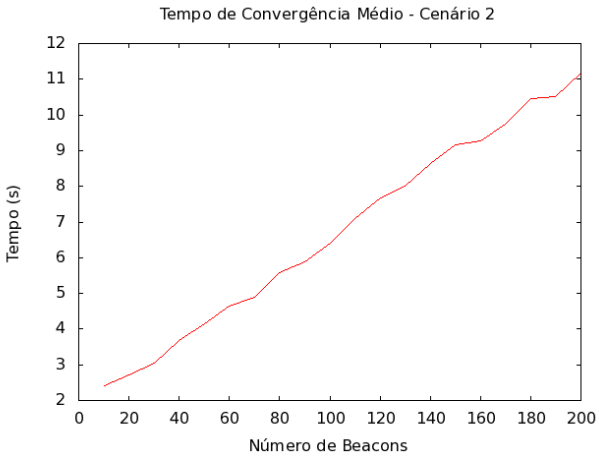

Figura 6.3: CHOPIN V1 - Tempo de Convergência Médio, cenário 2

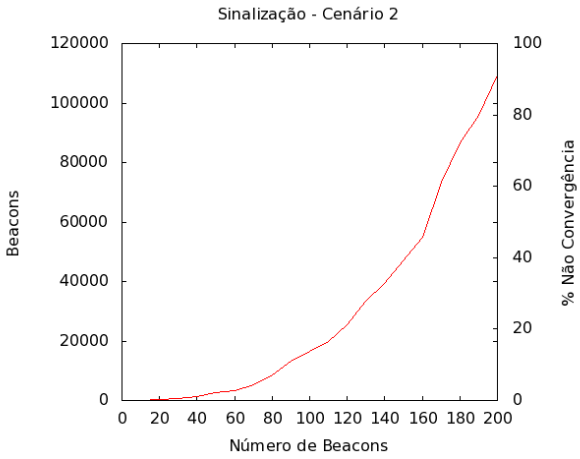

Figura 6.4: CHOPIN V1 - Sinalização, cenário 2

A figura 6.3, apresenta os tempos de convergência médios, para o cenário dois. Como era de esperar, os tempos de convergência, são elevados, pois os últimos nós do percurso, têm um longo caminho, até alcançarem a Base Station. Por exemplo, numa rede com duzentos nós, o beacon gerado, tem um tempo médio de propagação equivalente a $199 \times 0.05 = 9.95 s$. Apesar, de serem valores temporais altos, é o preço necessário, para garantir a convergência da rede.

O facto de a rede demorar demasiado tempo a convergir, influencia negativamente, a quantidade de beacons que são propagados pela rede, como se pode observar pela figura 6.4. Por outro lado, não há nenhum teste, em que a rede não tenha convergido.

A versão um do protocolo CHOPIN, apresenta resultados bastante negativos, no cenário dois, com níveis elevadíssimos de sinalização. É de salientar, que a curva sinalização, aparenta ter uma evolução quadrática, como aliás, já se tinha verificado na secção 4.3.

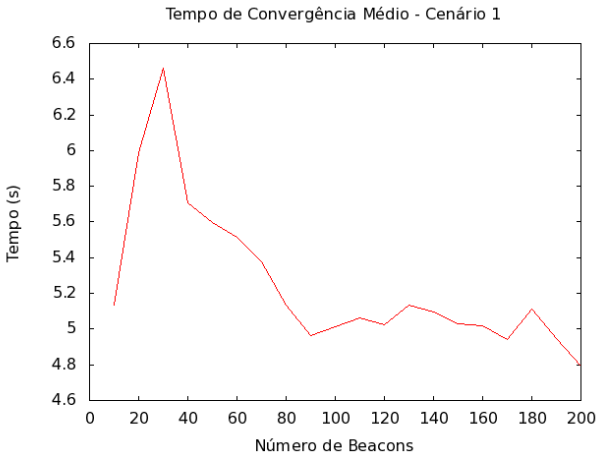

Figura 6.5: CHOPIN V1 - Tempo de Convergência Médio, cenário 1

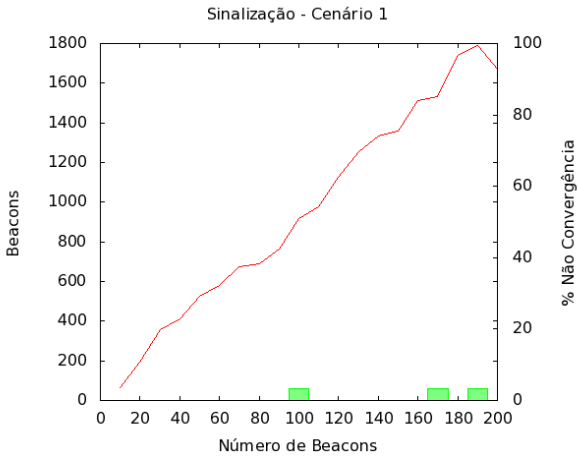

Figura 6.6: CHOPIN V1 - Sinalização, cenário 1

Como se pode observar, pela figura 6.5, os tempos de convergência diminuíram. Esta questão, era expectável, pois no cenário um, para o mesmo número de nós, há vários caminhos e consequentemente, mais curtos. Mesmo considerando, que o atraso máximo, em cada nó, é maior, para o cenário um, essa diferença é suplantada, pelo menor comprimento dos vários percursos existentes, contrastando com o único caminho no cenário dois.

Também na sinalização, os resultados no cenário um, foram superiores. Pela observação da figura 6.6, o nível de sinalização sofreu uma redução drástica. O que é natural, visto que há mais caminhos, com pouca ou nenhuma intersecção, e encaminhar Node Beacons até à Base Station.

O factor, de atraso no envio de pacotes, teve um efeito crucial no desempenho do protocolo. Ao contrário do que se verifica na figura 6.2, onde, para os testes efectuados, não foi inserido qualquer atraso na comunicação, praticamente todos testes resultaram em redes convergentes.

Pode-se concluir que em certa medida, o problema gerado por colisão de pacotes, foi ultrapassado. É crucial, que os próximos protocolos, consigam reduzir a quantidade de mensagens difundidas pela rede. Ao reduzir a sinalização gerada, é esperado que o número de colisões diminua e assim, garantir convergência total, para as redes testadas.

6.4 Protocolo CHOPIN V2

Nesta secção prosseguirá o estudo da versão dois do protocolo CHOPIN. Esta versão, foi inicialmente descrita, na secção 4.4, e visa reduzir a difusão de Base Sation Beacons que considerar desnecessários. Um nó, só retransmite um Base Station Beacon, se, pelo menos, um dos seus vizinhos, não se encontra na vizinhança, do nó que lhe enviou o beacon.

As configurações de teste são apresentadas na tabela 6.2. Como esta versão, tem o mesmo desempenho que a versão um, apenas serão demonstrados, os resultados dos testes para o cenário um.

	Cenário 1
Atraso máximo	1.5 s
REFRESH_INTERVAL	2 s

Tabela 6.2: NS-3, CHOPIN V2 - parâmetros de configuração

A configuração é a igual, àquela, utilizada para os testes da versão um.

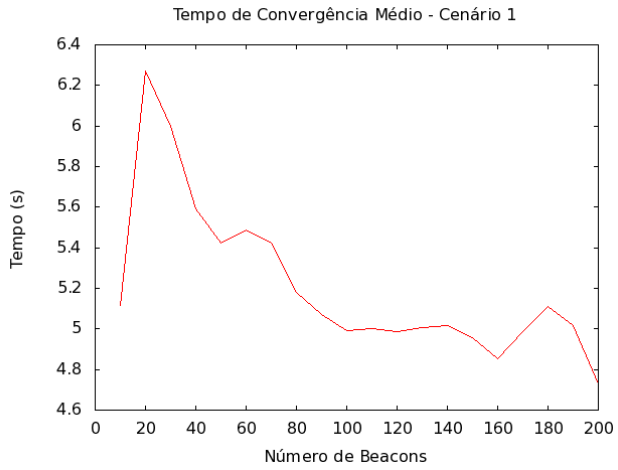

Figura 6.7: CHOPIN V2 - Tempo de Convergência Médio, cenário 1

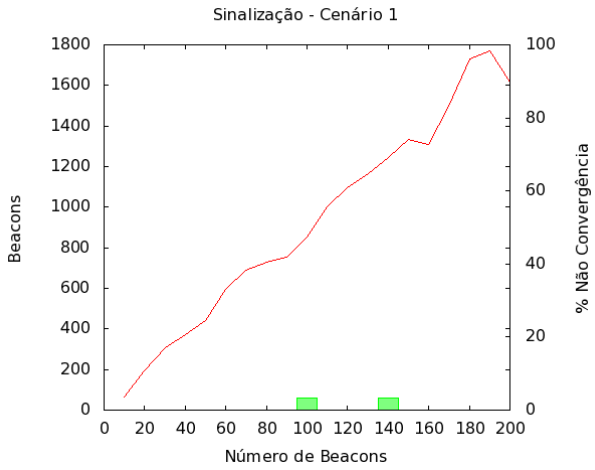

Figura 6.8: CHOPIN V2 - Sinalização, cenário 1

As figuras 6.7 e 6.8, apresentam os resultados dos testes efectuados para a versão dois, respectivamente, tempo de convergência médio e sinalização.

O desempenho da versão dois, não apresenta melhorias significativas, em relação à versão um. Tal, já tinha sucedido, na análise a esta versão, na secção 4.4. Esta técnica foca-se em evitar a difusão de Base Station Beacons desnecessários. Por sua vez, os Base Station Beacons, representam, uma pequena porção das mensagens geradas, sendo maior parte da informação gerada, constituída por Node beacons.

Como tal, os protocolos seguintes, que focam, a redução do tráfego, no sentido da Base Station, tenderão a apresentar melhores resultados. Este protocolo, poderá, todavia, ser um complemento, a qualquer protocolo. O protocolo resultante, conseguiria reduzir os dois tipos de sinalização.

6.5 Protocolo CHOPIN V3

Na versão três do protocolo, foi introduzido, um novo tema: nós de último nível. Estes nós, são aqueles, que se encontram, o mais distante da Base Station e os únicos que podem gerar Node Beacons. Como se viu na secção 4.5, com esta versão, foi possível reduzir o tráfego constituído por Node Beacons.

No entanto, foi necessário introduzir uma melhoria ao protocolo apresentado na secção

4.5. Nessa versão, apenas as mensagens do tipo Base Station Beacon, são consideradas para efeitos de verificação e alternância do estado de um nó móvel. Esta necessidade é fruto, da nova variável, colisão de pacotes, introduzida pelo simulador NS-3,

Essa verificação só acontece, quando o beacon volta para trás, ou seja, quando o nó recebe o Base Station Beacon, que tinha enviado, de nós superiores. Mas, caso a resposta se perca, o nó não saberá que há, pelo menos, um nó pai, acima dele e portanto, considera-se, nó de último nível, podendo gerar beacons periodicamente. Desta forma, o protocolo aproxima-se, do modo de funcionamento da versão um, onde todos os nós, dentro do sue período de tempo, difundiam o seu Node Beacon.

Nestas condições, os resultados não apresentavam melhorias, portanto, para contornar este problema, foi decidido realizar uma pequena alteração à versão três do protocolo.

Mensagens do tipo Node Beacon, passam também, a ser consultadas, pelos nós móveis, para verificarem, qual a sua posição, dentro do caminho, em que se encontram.

Assim, se o um nó, não tiver recebido resposta ao Base Station Beacon e receber posteriormente, um Node Beacon, que esteja de acordo com as regras do protocolo, ele ficará a saber que não é o último nó do seu caminho e portanto, apenas poderá retransmitir Node Beacons, de nós de nível superior e, que o tenham, como próximo salto, para a Base Station.

Os protocolos quatro e cinco, analisados nas secções seguintes, também foram alterados, pois esta é uma questão comum às três versões. Foi ainda, alterado, um pormenor nestes protocolos. Nas três versões, os nós móveis, apenas podem iniciar a difusão de Node Beacons, após a recepção de um Base Station Beacon. Depois de difundirem o seu Node Beacon, os nós só poderão voltar a enviar outro, após a recepção do próximo Base Station Beacon. No caso da versão três, apenas nós de último nível, continuam a poder iniciar a difusão de Node Beacons.

	Cenário 1	Cenário 2
Atraso máximo	1 s	0.1 s
REFRESH_INTERVAL	8 s	8 s

Tabela 6.3: NS-3, CHOPIN V3 - parâmetros de configuração

A tabela 6.3, apresenta os parâmetros de configuração, para os diferentes cenários. De salientar, o intervalo de tempo de oito segundos. Dos vários pares, <Atraso máximo;REFRESH_INTERVAL>, testados, foi o que obteve os melhores resultados, tendo sempre em mente, a redução dos níveis de sinalização e a convergência da rede.

Com a introdução de nós de último nível, foi necessário aumentar o intervalo entre o envio de beacons. Este valor, tem que ser suficientemente grande, para permitir que o beacon alcance a Base Station, antes que se inicie a próxima difusão de beacons.

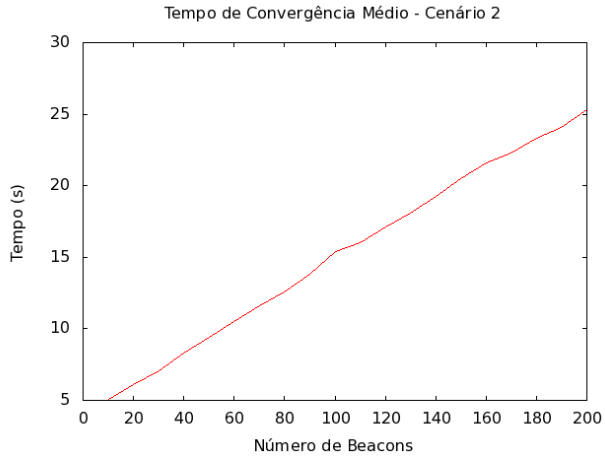

Figura 6.9: CHOPIN V3 - Tempo de Convergência Médio, cenário 2

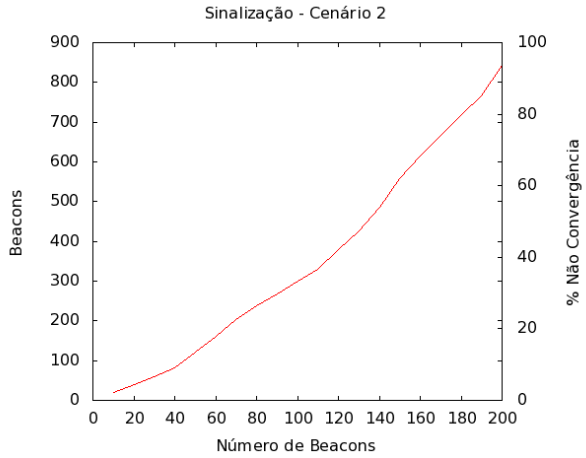

Figura 6.10: CHOPIN V3 - Sinalização, cenário 2

Observando os gráficos acima, observa-se uma redução na quantidade de sinalização gerada e total convergência, às custas, de um aumento nos tempos de convergência médios. Começa-se a observar, a existência de um compromisso, entre estas características. Para melhorar a sinalização, poderá ter que haver um retrocesso, nos tempos de convergência.

Ao comparar estes resultados, com aqueles obtidos, para a versão um e cenário dois, observa-se, que de facto, esta versão foi capaz de reduzir a propagação de Node Beacons mas, por outro lado, demorou mais tempo a convergir.

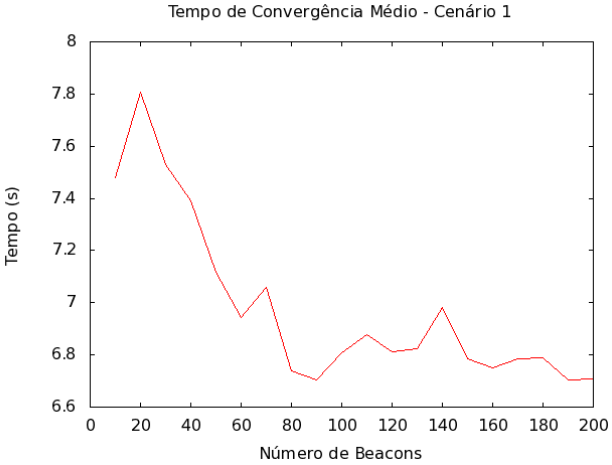

Figura 6.11: CHOPIN V3 - Tempo de Convergência Médio, cenário 1

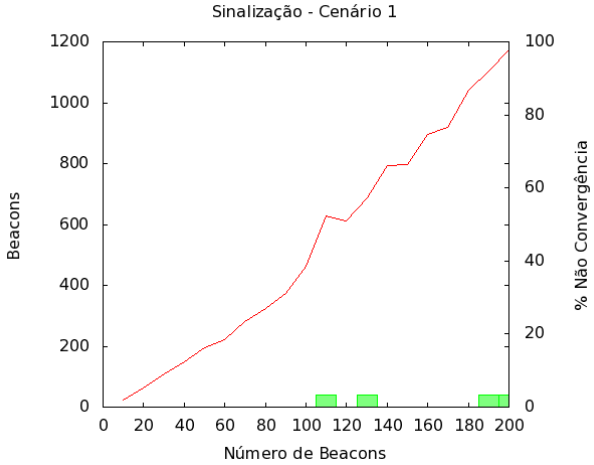

Figura 6.12: CHOPIN V3 - Sinalização, cenário 1

As figuras 6.11 e 6.12, apresentam respectivamente, os tempos de convergência médio e sinalização, para o cenário um.

Mais uma vez, os resultados apresentados, mostram um aumento no tempo de convergência e redução na quantidade de sinalização gerada. Por outro lado, em quatro testes, a rede não convergiu, ao contrário, do que se verificou para o cenário dois.

Como foi analisado na secção 6.2, no cenário um, o protocolo é testado, em redes bastante densas. Considerando também, o elevado número de nós, há a tendência para a ocorrência, de um maior número de colisões. Com a perda de um único beacon, pode ser necessário, esperar pela próxima difusão. Com intervalos de tempo maiores, como os oito segundos atribuídos a este teste, há menos hipóteses de, em caso de perda, a rede convergir no tempo limite. Quanto maior os intervalos de tempo, menor o número de difusões de beacons, dentro do limite de tempo definido.

6.6 Protocolo CHOPIN V4

Nesta secção será abordada, a técnica introduzida na secção 4.6. Nesta versão, um nó, tem a capacidade, de enviar múltiplos caminhos. À medida que vão chegando, vão sendo guardados numa tabela. Quando chega o momento, de gerar o próximo beacon, todos os caminhos, que até então, foram guardados, são colocados na mensagem a enviar. Consegue-se desta forma, poupar no número de retransmissões, em comparação, com a versão anterior.

	Cenário 1
Atraso máximo	1 s
REFRESH_INTERVAL	4 s

Tabela 6.4: NS-3, CHOPIN V4 - parâmetros de configuração

A tabela 6.4, apresenta os parâmetros de configuração escolhidos para testar a versão quatro. O valor do intervalo, tem que ser curto, por causa do retardamento de caminhos, implemento nesta versão.

Apenas são apresentados resultados para o cenário um. Como se viu na secção 4.6, este protocolo não é aplicável para cenários, com caminhos grandes.

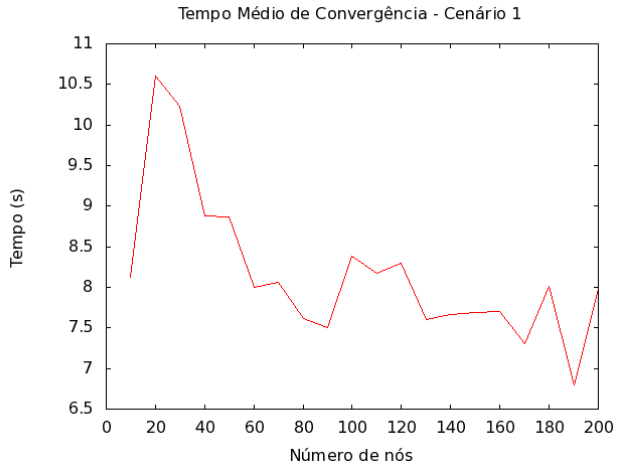

Figura 6.13: CHOPIN V4 – Tempo Médio de Convergência, cenário 1

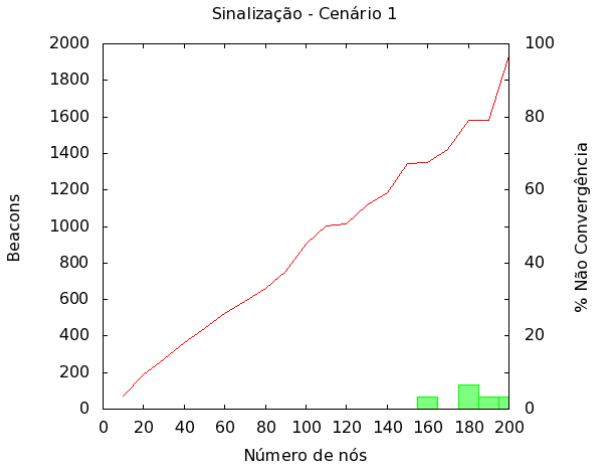

Figura 6.14: CHOPIN V4 – Sinalização, cenário 1

Como se pode observar pelos gráficos das figuras 6.13 e 6.14, o protocolo, mostra ser um retrocesso. Tanto os tempos de convergência, como a quantidade de sinalização gerada, aumentaram. Nomeadamente, os níveis de sinalização, até conseguiram ultrapassar, os níveis da versão um do protocolo.

Comum tempo de convergência médio, menor, este protocolo, gerou mais sinalização, que o seu antecessor.

6.7 Protocolo CHOPIN V5

A versão cinco do CHOPIN, é uma optimização da versão quatro. Os nós intermédios, deixam de ter de esperar pelo temporizador, de geração de beacons, para enviarem os caminhos dos nós pais, que foi armazenando.

Um nó intermédio, pode enviar todos os caminhos armazenados, a partir do momento, em que todos os nós, para os quais encaminha Node Beacons e que sejam de nível superior, enviarem os seus Node Beacons.

	Cenário 1	Cenário 2
Atraso máximo	1 s	0.05 s
REFRESH_INTERVAL	20 s	20 s

Tabela 6.5: NS-3, CHOPIN V5 - parâmetros de configuração

A tabela 6.5, apresenta os parâmetros de configuração escolhidos para testar a esta versão. O valor do intervalo, tem que ser curto, por causa do retardamento de caminhos, implemento nesta versão.

Para ambos os cenários, a Base Station, gera o seu beacon de vinte em vinte segundos. Este tempo, reserva algum tempo, para o caso se existirem caminhos maiores do que o esperado.

Para o cenário dois foi escolhido um tempo bastante curto, para o atraso máximo. No entanto, tendo em conta a topologia da rede, há menos hipóteses de acontecerem colisões e ao mesmo tempo, conseguir suportar caminhos, tão grandes, como o cenário apresenta. De lembrar, que o o teste maior, é constituído por uma fila de duzentos nós.

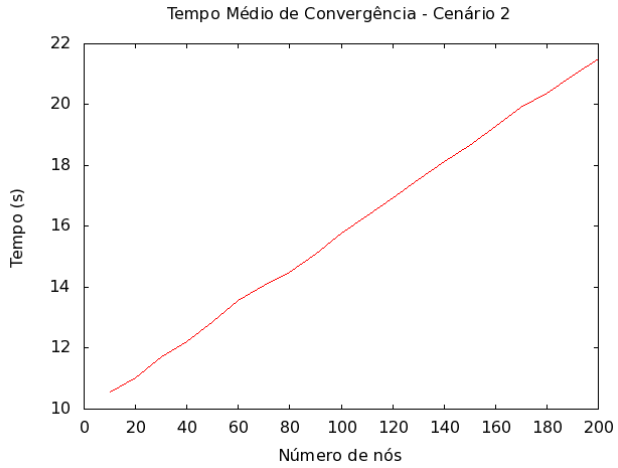

Figura 6.15: CHOPIN V5 – Tempo Médio de Convergência, cenário 2

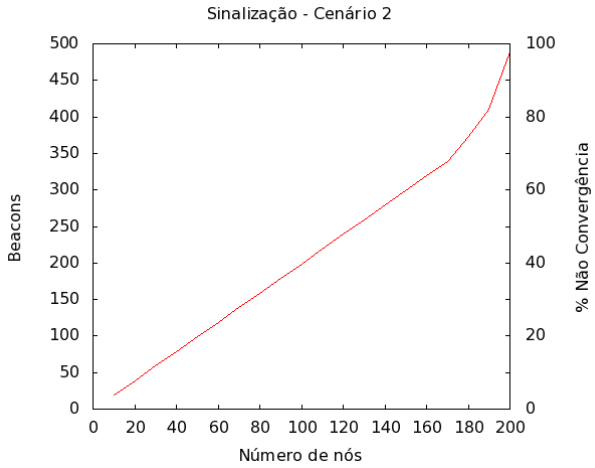

Figura 6.16: CHOPIN V5 – Sinalização, cenário 2

Os resultados, apresentados pelas figuras, 6.15 e 6.16, seguem em concordância, com os resultados obtidos, ao longo do capítulo 4. A versão conseguiu níveis de sinalização, superiores, a qualquer outro dos protocolos.

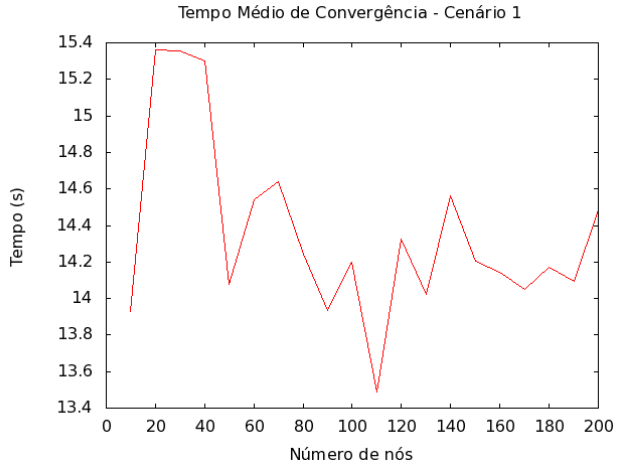

Figura 6.17: CHOPIN V5 – Tempo Médio de Convergência, cenário 1

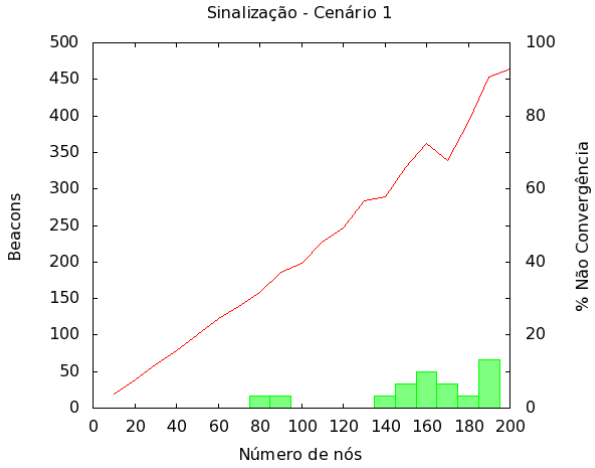

Figura 6.18: Sinalização, cenário 1

As figuras 6.17 e 6.18 mostram os resultados obtidos, para o cenário um.

Comparativamente às outras versões, o tempo médio de convergência aumentou. Este

dado, é explicado, pelo maior intervalo temporal, entre o envio de Base Station Beacons. O valor atribuído de vinte segundos, é bastante superior aos utilizados, para as outras versões.

Este é também a principal causa, de algumas redes não terem convergido, durante os sessenta segundos, o limite de convergência definido previamente. Quando acontece, uma colisão entre pacotes, o nó só poderá voltar a enviar o seu beacon, no próxima difusão. Como cada difusão, acontece a cada vinte segundos, há apenas três tentativas para o conseguir. E se acontecer, que há uma zona na rede, muito densa, a probabilidade de ocorrência de colisões é elevada.

Contudo, os níveis de sinalização, atingiram os níveis mais baixos, em todos os testes. Em comparação com a versão três, há uma diminuição, em mais de metade, no número de mensagens produzidas.

6.8 Sumário

Neste capítulo, procurou-se repetir o estudo feito, no capítulo 4, com a diferença, na escolha do simulador. O simulador NS-3, foi a ferramenta utilizada, para avaliar o desempenho dos diferentes protocolos.

Neste estudo, surgiu uma nova variável, colisão de pacotes. Ao contrário, do simulador Adevs, o NS-3, simula a colisão de pacotes. A influência desta variável é visível, pela figura 6.2, onde, a maior parte dos testes, não se verificou convergência.

Como os nós não tinham qualquer tipo de sincronismo, os pacotes eram enviados, praticamente em simultâneo, resultando em colisões e consequentemente, pacotes que não chegavam ao destino. Este facto é preocupante, porque um nó, arrisca-se a ter de esperar pela próxima difusão.

Este problema, é especialmente problemático, para redes do cenário um. Aqui, como há muitos nós ao alcance, uns dos outros, são geradas, num curto espaço, demasiadas mensagens, para que não ocorra qualquer colisão. Pelo contrário, no cenário dois, os nós não têm mais de dois vizinhos, não sendo um caso tão problemático.

De forma a reduzir o número de colisões, foi determinado, que cada nó deveria atrasar a sua transmissão, dentro de um dado intervalo.

O próximo passo, consistiu, em analisar os protocolos. Na generalidade, os testes apresentaram algumas redes, não convergentes. É uma desvantagem, da introdução de um atraso, antes da transmissão. Por todos os nós que o beacon passa, o seu tempo de propagação, aumenta, devido ao atraso acumulado, por todos esses nós. Foi então, necessário equilibrar, o valor de atraso de transmissão e o período de difusão de beacons.

Os resultados, reforçam, que o protocolo CHOPIN V5, é aquele, com menos sinalização gerada, tendo conseguido, uma boa vantagem, para os demais protocolos.

7 Tempo Médio de Convergência vs Sinalização

Durante a fase de testes, foram definidas duas variáveis de avaliação: tempo médio de convergência e sinalização. Foram criados diversos protocolos, na procura de um, que conseguisse bons tempos de convergência e sinalização.

Neste capítulo, é pretendido, definir um compromisso, através de análise experimental, entre o tempo médio de convergência e a sinalização gerada. Quanto maior for o número de mensagens difundidas pela rede, mais rápido irá a rede convergir e quanto menos mensagens forem geradas, mais lentamente, irá a rede convergir.

Os protocolos criados, foram executados, para diferentes combinações, do par <atraso máximo; período de difusão de beacons>.

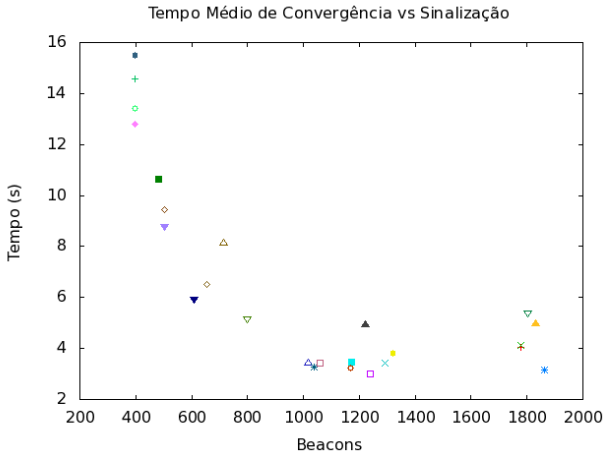

Figura 7.1: Tempo médio de convergência vs sinalização

Cada ponto no gráfico da figura 7.1, representa, um protocolo, com um dado atraso máximo e um dado intervalo de difusão de beacons.

O compromisso entre estas variáveis, está definido no gráfico. A maior parte dos pontos, forma uma curva, descendente no eixo do tempo e ascendente no eixo dos beacons.

Esta curva representa o compromisso, atrás referido, entre tempo médio de convergência e número de beacons propagados.

8 Conclusão

O presente trabalho, teve como objectivo, o estudo do protocolo CHOPIN. Este protocolo, foi idealizado, para a gestão de uma MANET, em cenários de catástrofe de busca e salvamento. O estudo revelou que o protocolo apresentava algumas debilidades, pois gerava, uma larga quantidade de mensagens .

Para tentar resolver este problema, foram criadas mais quatro versões. O CHOPIN V2, visava a redução de Base Station Beacons e por isso, acabou por não apresentar melhorias significativas, pois a maior parte do tráfego gerado é constituído por Node Beacons.

As três versões seguintes focaram-se, então, na redução de Node Beacons. À excepção, do CHOPIN V4, os resultados foram satisfatórios. A versão cinco, foi aquela, que conseguiu gerar o menor número de mensagens, com números agradáveis.

Os testes foram executados em dois simuladores distintos: Adevs e NS-3. No simulador NS-3, surgiu uma nova variável, a colisão de pacotes. Para resolver o problema, foi necessário Introduzir, um atraso a cada retransmissão, de forma a evitar possíveis colisões. Atrasar a transmissão de pacotes, tem a vantagem de diminuir as perdas, mas, por outro lado, também obriga, a que o intervalo de tempo entre difusão de beacons, seja superior.

Por último, foi definida uma linha de compromisso, de forma experimental, entre o tempo de convergência e a sinalização gerada. Essa linha de compromisso, define que quanto maior for o número de mensagens difundidas, menor será o tempo de convergência médio. O Contrário também se pode afirmar.

Como trabalho futuro, é pretendido demonstrar, formalmente, qual a linha de compromisso óptima. Para além disso, há mais duas experiências a realizar. Desenvolver a versão cinco do CHOPIN, de modo, a que consiga gerar valores adaptativos para os temporizadores dos nós, dependendo do tamanho do caminho e posição em que se encontra. Nós mais próximos da Base Station terão valores mais elevados nos seus temporizadores. Por último, realizar testes de comparação de desempenho, entre o protocolo CHOPIN e outros protocolos Ad Hoc existentes, utilizando os módulos do simulador NS-3, como por exemplo, o OLSR.

Referências

[1] R. P. Rocha, D. Portugal, M. S. Couceiro, F. Araújo, P. Menezes and J. Lobo, The CHOPIN project: Cooperation between Human and rObotic teams in catastroPhic INcidents, 2013, In Proc. of 11th IEEE Int. Symp. on Safety, Security, and Rescue Robotics (SSRR 2013), Linköping, Sweden, Oct. 21-26, 2013. DOI: 10.1109/SSRR.2013.6719322
[2] F. Araújo, J. Santos e R. P. Rocha, Implementation of a Routing Protocol for Ad Hoc Networks in Search and Rescue Robotics, 2014, In Proc. of 2014 IEEE Wireless Days (WD'14), Rio de Janeiro, Brazil, Nov. 12-14, 2014. DOI: http://dx.doi.org/10.1109/WD.2014.7020821
[3] José Santos, Implementation of a Mobile Ad Hoc NETwork communication protocol for human-robot search and rescue teams, 2013, Dissertação de Mestrado, Universidade de Coimbra, Jul. 2013
[4] 2015, http://web.ornl.gov/~1qn/adevs/adevs-docs/manual.pdf
[5] 2015, http://en.wikipedia.org/wiki/SimPy
[6] 2015, http://en.wikipedia.org/wiki/DESMO-J
[7] 2015, http://desmoj.de/
[8] 2015, http://web.ornl.gov/~1qn/adevs/
[9] 2015, https://www.nsnam.org/docs/tutorial/html/getting-started.html
[10] 2015, https://www.nsnam.org/wiki/Installation
[11] 2015, https://www.nsnam.org/release/
[12] Muneer Alshowkan, Eman Abdel Fattah, Ammar Odeh, Performance Evaluation of DYMO, AODV and DSR Routing Protocols in MANET, 2012, http://www.researchgate.net/profile/Ammar_Odeh/publication/257769212_Performance_Evaluation_of_DYMO_AODV_and_DSR_Routing_Protocols_in_MANET/links/0046352803ea3795a8000000International Journal of Computer Applications
[13] Anuj K. Gupta, Jatinder Kaur, Sandeep Kaur, Comparison of DYMO, AODV, DSR, and DSDV MANET Routing Protocols over Varying Traffic, 2011, http://www.researchgate.net/profile/Anuj_Gupta4/publication/220029132_A_Comparative_analysis_of_AODV_DSR__DYMO_reactive_routing_protocols_for_MANETs/links/09e4150ec3fda3003c000000International Journal of Research in Engineering & Applied Science
[14] Parma Nand, Dr. S.C. Sharma, Performamce study of Broadcast based Mobile Adhoc Routing Protocols AODV, DSR and DYMO, 2011, http://www.sersc.org/journals/IJSIA/vol5_no1_2011/5.pdfInternational Journal of Security and its Applications
[15] U. Devee Prasan, Dr. S. Murugappan, A Survey on Issues in Designing a Routing Protocol, 2013, http://www.theglobaljournals.com/ijar/file.php?val=May_2013_1367501470_b5612_44.pdfInidian Journal of Applied Research
[16] Vandana Jindal, A.K. Verma, Seema Bawa, How the two Adhoc networks can be different: MANET & WSNs, 2011, http://www.ijcst.com/vol24/1/vandana.pdfInternational Journal of Computer Science & Technology
[17] Kavita Pandey, Abhishek Swaroop, A Comprehensive Performance Analysis of Proactive, Reactive and Hybrid MANETs Routing Protocols, 2011, http://ijcsi.org/papers/IJCSI-8-6-3-432-441.pdfInternational Journal of Computer Science
[18] Surendra H. Raut, Hemant P. Ambulgekar, How the two Adhoc networks can be different: MANET & WSNs, 2013, http://www.ijarcsse.com/docs/papers/Volume_3/4_April2013/V3I3-0374.pdfInternational Journal of Advanced Research in Computer Science and Software Engineering
[19] Alok Singh Thakur, Anita Ganpati, A Comparative Study of DYMO, AODV, DSR & DSDV Routing Protocols in VANET, 2013,

http://www.ijarcsse.com/docs/papers/Volume_3/10_October2013/V3I10-0242.pdf
[20] 2011, https://tools.ietf.org/html/rfc6126#page-3
[21] Miss Laiha Mat Kiah, Liana Khamis Qabajeh, Mohammad Moustafa Qabajeh, Unicast Position-based Routing Protocols for Ad-Hoc Networks, 2010, http://www.uni-obuda.hu/journal/Kiah_Qabajeh_Qabajeh_26.pdfActa Polytechnica Hungarica
[22] David B. Johnson, David A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, , http://www.utdallas.edu/~ksarac/Papers/DSR.pdf
[23] David B. Johnson, David A. Maltz, Josh Broch, DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, , http://www.monarch.cs.rice.edu/monarch-papers/dsr-chapter00.pdf
[24] Sukant Kishoro Bisoyi, Sarita Sahu, Performance Analysis of Dynamic MANET On-demnd (DYMO) Routing protocol, 2010, http://interscience.in/SpIss_ijcct_accta_2010vol1_nol2/NSC_Paper8.pdfInternational Journal of Computer Trends and Technology
[25] Anuj K. Gupta, Harsh Sadawarti, Anil K. Verma, Performance Enhancement of DYMO Routing Protocol with Ant Colony Optimization, 2014, http://www.ijeee.net/uploadfile/2014/0224/20140224025813293.pdfInternational Journal of Electronics and Electrical Engineering
[26] 2007, http://www.ietf.org/rfc/rfc4728.txt
[27] 2003, https://www.ietf.org/rfc/rfc3626.txt
[28] Justimiano Andrade Alves, Route Alternating Multipath AODV (RAM-AODV), 2012
[29] , http://en.wikipedia.org/wiki/DYMO
[30] , Dynamic Source Routing, , http://en.wikipedia.org/wiki/Dynamic_Source_Routing