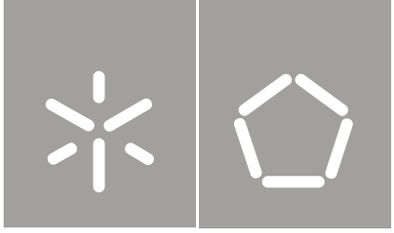
Universidade do Minho
Escola de Engenharia

Tiago Miguel da Cunha Pimenta

Simulação de Protocolos de Encaminhamento para Redes AdHoc usando o NS-3

Simulação de Protocolos de Encaminhamento para Redes AdHoc usando o NS-3
Tiago Miguel da Cunha Pimenta
UMinho | 2013

dezembro de 2013



Universidade do Minho
Escola de Engenharia

Tiago Miguel da Cunha Pimenta

Simulação de Protocolos de Encaminhamento para Redes AdHoc usando o NS-3

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao
Grau de Mestre em Engenharia de Comunicações

Trabalho efetuado sob a orientação de
Professora Doutora Maria João Nicolau
Professor Doutor António Costa

dezembro de 2013

DECLARAÇÃO

Nome: Tiago Miguel da Cunha Pimenta

Correio electrónico: tiago.comunicacao@gmail.com

Tel./Tlm.: 934572916

Número do Bilhete de Identidade: 13311178

Título da dissertação: Simulação de Protocolos de Encaminhamento para Redes AdHoc usando o NS-3

Ano de conclusão: 2013

Orientador(es): Professora Doutora Maria João Nicolau e Professor Doutor António Costa

Designação do Mestrado: Mestrado Integrado em Engenharia de Comunicações

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia de Comunicações

Escola: Escola de Engenharia
Departamento: Sistemas de Informação/ALGORITMI

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA DISSERTAÇÃO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA DISSERTAÇÃO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

3. De acordo com a legislação em vigor, não é permitida a reprodução de qualquer parte desta dissertação

Guimarães, 17/12/2013

Assinatura: _____

Resumo

As redes Ad Hoc móveis consistem numa coleção de nós móveis localizados arbitrariamente e que se deslocam dinamicamente numa região espacial, de tal modo que as ligações entre os nós podem mudar continuamente. Deste modo, de forma a facilitar a comunicação entre os diversos nós constituintes dentro de uma rede Ad Hoc, um ou vários protocolos de encaminhamento são utilizados de forma a descobrir e manter rotas entre os nós, oferecendo redundância à rede. O principal objetivo do encaminhamento numa rede Ad Hoc, consiste no estabelecimento eficiente de uma rota entre um par de nós, de modo a que as mensagens trocadas entre dispositivos possam ser entregues de forma atempada e correta. A construção de uma rota deve ser feita com o mínimo de sobrecarga na rede, utilizando o mínimo de consumo de largura de banda e energia dos dispositivos.

Este trabalho consiste na análise e avaliação dos protocolos de encaminhamento em redes Ad Hoc, através do estudo de desempenho com base num determinado conjunto de parâmetros. Primeiramente são apresentados alguns dos principais protocolos utilizados no contexto de redes Ad Hoc(AODV, OLSR, DSDV) e fazendo a descrição das suas características e funcionalidades (diferentes cenários de aplicação, vantagens e desvantagens, e alguns dos conceitos fundamentais sobre as mesmas redes). Em seguida, é desenvolvida e apresentada uma simulação de uma rede Ad Hoc, a fim de testar vários cenários de aplicação de redes Ad Hoc, com diferentes parâmetros, concretizando assim um estudo do desempenho de cada um dos protocolos. Estes protocolos são implementados no Simulador *Network Simulator*, NS-3. A respetiva avaliação é feita com base na análise dos dados obtidos na simulação dos diferentes cenários.

Palavras-chave: Ad Hoc, AODV, OLSR, DSDV, NS-3

Abstract

The mobile Ad Hoc networks consist of a collection of mobile nodes located arbitrarily and dynamically in a spatial region, so that the connections between the nodes are able to change continuously. Thus, in order to facilitate communication among the various constituent nodes within an Ad Hoc network, one or more Ad Hoc network routing protocols are used to discover and maintain routes between nodes, providing redundancy to the network. The main goal of routing in Ad Hoc network, consists in establishing one efficient route between a pair of nodes so that messages exchanged between devices can be delivered within the time and correctly. The construction of a route should be done with minimal overhead on the network, using the minimum consumption of power and bandwidth of the device.

This work consists in the analysis and evaluation of Ad Hoc network routing protocols, through the study of performance based on a given set of parameters. First, some of the main protocols used in the context of Ad Hoc networks (AODV, OLSR, DSDV) are presented, making the description of its features and functionality (different application scenarios, advantages and disadvantages, and some fundamental concepts on these networks). Then, a simulation of an Ad Hoc network is developed and presented, in order to test several scenarios of the application of Ad Hoc networks, with different parameters, thereby conducting a study on the performance of each protocol. These protocols are implemented in the simulator Network Simulator, NS-3. The evaluation is made based on the results obtained from simulation of distinct scenarios.

Keywords: Ad Hoc, AODV, OLSR, DSDV, NS-3

Agradecimentos

Depois dos dias de trabalho passados no desenvolvimento deste documento e de todo o trabalho desenvolvido, apenas me resta deixar os meus sinceros agradecimentos à minha família. Especialmente aos meus pais, à minha irmã e às minhas duas "trambolhas", Leonor e Margarida, que sempre me encheram de alegria e animo nos momentos de menos vontade e paciência, por todo o apoio emocional, carinho e paciência.

Um maior abraço de agradecimento aos meus orientadores, Professora Maria João e ao Professor António Costa, por todas as horas passadas com os seus ensinamentos e experiências, quer nas unidades curriculares lecionadas quer no longo do trabalho desenrolado, pela sua total disponibilidade e dedicação, por todo o seu apoio e por toda a paciência que tiveram comigo ao longo de todas as contrapartidas que surgiram. Porque graças às suas contribuições, foi possível a continuidade e conclusão desta dissertação.

E sem deixar de esquecer, agradecer a todos os meus colegas e amigos Manuel Rocha, César Oliveira, Hélder Ribeiro, Hugo Leite, João Brito, Susana Pereira, Luís Nascimento, Ricardo Maciel, Rui Rodrigues, Micael Gonçalves, César Sousa, Henrique Guedes, Marisa Alves, Paula Gonçalves, entre outros pela amizade, pela troca de conhecimentos, por todos os bons momentos e contributo para a realização deste projeto.

E por fim, em relação a nomes, gostaria de agradecer o maior apoio à minha namorada Manuela, que sempre lutou ao meu lado e sempre mostrou todo o apoio e dedicação ao longo deste tempo.

Como não poderia deixar de esquecer, agradeço também a todas pessoas que contribuíram indiretamente para a concretização deste trabalho e cujos nomes aqui não poderiam ser também referenciados, tornando-se numa lista extensa que daria resultado a uma outra dissertação.

Um beijinho
para a minha mãe,
para o meu pai
e para ti :-)

"**No meio de toda a a dificuldade encontra-se a oportunidade.**"
Albert Einstein

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ABR	**A**ssociativity-**B**ased **R**outing
ACM	**A**ssociation for **C**omputing **M**achinery
AODV	**A**d Hoc **O**n-Demand **D**istance **V**ector
BER	**B**it **E**rror **R**ate
BTS	**B**ase **T**ransceiver **S**tation
CGSR	**C**lusterhead **G**ateway **S**witch **R**outing
CSMA	**C**arrier **S**ense **M**ultiple **A**ccess
DBF	**D**istributed **B**ellman-**F**ord
DCF	**D**istributed **C**oordination **F**unction
DSDV	**D**estination-**S**equence **D**istance-**V**ector
DSR	**D**ynamic **S**ource **R**outing
EDCA	**E**nhanced **D**CF **C**hannel **A**ccess
HCCA	**H**CF **C**ontrolled **C**hannel **A**ccess
HCF	**H**ybrid **C**oordination **F**unction
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IETF	**I**nternet **E**ngineering **T**ask **F**orce
INRIA	**I**nstitut **N**ational de **R**echerche en **I**nformatique et en **A**utomatique
LLC	**L**ogic **L**ink **C**ontrol
LMR	**L**ightweight **M**obile **R**outing
MAC	**M**edia **A**ccess **C**ontrol
MANET	**M**obile **A**d Hoc **NET**work
MID	**M**ultiple **I**nterfaces **D**eclaration
MPR	**M**ulti **P**oint **R**elays
NS	**N**etwork **S**imulator
OLSR	**O**ptimized **L**ink **S**tate **R**outing
OSI	**O**pen **S**ystems **I**nterconnection

PAN	**P**ersonal **Á**rea **N**etwork
PCF	**P**oint **C**oordination **F**unction
PDA	**P**ersonal **D**igital **A**ssistant
PER	**P**acket **E**rror **R**ate
PHY	**Phy**sical Layer
RERR	**R**oute **ERR**or
RIP	**R**outing **I**nformation **P**rotocol
RREP	**R**oute **REP**ly
RREQ	**R**oute **REQ**uest
RSSF	**R**ede de **S**ensores **S**em **F**io
SSR	**S**ignal **S**tability **R**outing
TC	**T**opology **C**ontrol
TCP	**T**ransmission **C**ontrol **P**rotocol
TORA	**T**emporally **O**rdered **R**outing **A**lgorithm
TTL	**T**ime **T**o **L**ive
UDP	**U**ser **D**atagram **P**rotocol
UID	**U**nique **I**dentification Number
VANET	**V**ehicular **A**d Hoc **NET**work
WI-FI	**Wi**reless **Fi**delity
WIMAX	**W**orldwide **I**nteroperability for **M**icrowave **A**ccess
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WRP	**W**ireless **R**outing **P**rotocol
YANS	**Y**et **A**nother **N**etwork **S**imulator
ZRP	**Z**one **R**outing **P**rotocol

1

Introdução

Este primeiro capítulo divide-se em 5 secções e tem como objetivo fazer um enquadramento do tema, dando uma visão global sobre a base de desenvolvimento do trabalho. Na primeira secção realçam-se as características das redes sem fios Ad Hoc, comparativamente com as tradicionais redes com infraestrutura, bem como o modo como é estabelecido o encaminhamento nestas redes. Nas secções seguintes apresenta-se a motivação para a realização deste trabalho e os principais objetivos que se pretendem a atingir. Posteriormente na quarta secção são apresentadas as contribuições e a abordagem seguida e, por fim, na última secção descreve-se a estrutura geral deste documento.

1.1 Enquadramento

O uso da Internet e o acesso à informação através da mesma, faz com que as

ligações sem fios se tornem cada vez mais uma necessidade. A utilização destas tecnologias está a obter uma crescente popularidade e assiste-se na atualidade ao seu enorme desenvolvimento. Atualmente, a maioria dos indivíduos possuí um dispositivo móvel, como por exemplo *smatphone*, *netbooks*, *PDA*, *tablet*, entre outros, o qual requer uma constante ligação à rede para troca de informação com outros utilizadores. Estes aparelhos oferecem uma vasta gama de soluções atraentes aos utilizadores que pretendem uma rápida e simples ligação à Internet, sem necessidade de utilização de fios ou de uma infraestrutura de rede. Deste modo, pretende-se ter acesso à rede a partir de qualquer ponto ou local geográfico, podendo-se mesmo adaptar a estrutura da rede à mobilidade dos dispositivos.

Existem três tipos de redes sem fios: redes com infraestrutura, redes sem infraestrutura (Ad Hoc) e redes híbridas. As redes com infraestrutura ou redes estruturadas (redes com *gateways* fixos e com fios) são constituídas por Estações Base (por exemplo BTS *Base transceiver station* ou centrais de comunicação), às quais as unidades móveis se ligam (Figura 1.1). Cada dispositivo móvel liga-se à estação mais próxima dentro da sua área de cobertura de forma a obter acesso à rede. Um dos problemas das redes com infraestrutura pré-definida é o espaço de abrangência que pode ser muito limitado, restrito apenas um raio de cobertura específico de cada Estação Base, fazendo com que os dispositivos móveis no momento em que estão a sair da área de abrangência de uma Estação Base e a entrar num raio de cobertura de outra Estação Base, forcem a ocorrência do processo de *handoff*, da antiga Estação Base até à nova Estação Base. Deste modo, com o aumento da mobilidade dos dispositivos (ou nós, como trataremos a partir de agora) e a ocorrência de *handover* dos dispositivos nas distintas redes, as redes sem fios com infraestrutura podem tornar-se pouco funcionais relativamente à mobilidade dos dispositivos [46].

Ao contrário das redes com infraestrutura, as redes Ad Hoc (que em latim significa "para este fim") não necessitam de uma infraestrutura definida. Desta forma, obtemos um modelo de rede que não necessita de *Access Points* entre os dispositivos, de *routers* ou de uma administração centralizada. As redes Ad Hoc ou MANET(*Mobile Ad Hoc Network*), são uma coleção de dispositivos sem fios

Figura 1.1: Exemplo de uma rede sem fios com infraestrutura

ligados entre si, que se movem dinamicamente pelo espaço da rede e que, se ligam e desligam, dinamicamente uns aos outros como ilustrado na Figura 1.2. Neste cenário obtém-se uma rede com topologia dinâmica, na qual os nós operam com funcionalidades comparadas a um *router*, sendo capazes de descobrir a topologia da rede e de propagarem os pacotes de dados entre os diversos dispositivos que estão ligados. Em comparação com as redes fixas, as redes Ad Hoc apresentam um dinamismo muito maior. A mobilidade dos nós causa com muita frequência alterações topológicas, que raramente acontecem nas redes fixas. Numa rede Ad Hoc, os dispositivos móveis são capazes de criar uma rede numa área onde não existe infraestrutura de rede pré-definida.

Por fim, as redes híbridas consistem numa combinação de funcionalidades e aspetos dos dois tipos de redes descritos anteriormente (Figura 1.3). Uma rede híbrida pode utilizar padrões de comunicação diferentes, interligando por exemplo redes *Ethernet* e *WiFi*).

De entre todos os aspetos a ter em conta, a forma como é implementado o encaminhamento do tráfego é, não só um dos pontos-chave, como também uma das questões mais difíceis de resolver neste contexto. Os algoritmos de encaminhamento, habitualmente usados nas redes tradicionais, têm que ser adaptados a estas novas redes. Graças às alterações topológicas frequentes, o *overhead* introdu-

Figura 1.2: Exemplo de uma rede AdHoc

zido pelo tráfego de controlo é muito significativo, podendo consumir grande parte dos recursos de comunicações disponíveis, que neste tipo de redes é mais escasso. Além disso, os ciclos de encaminhamento e as inconsistências da informação de encaminhamento, derivados do funcionamento dos protocolos de encaminhamento, agravam-se com o aumento do dinamismo destas redes. A gestão dos recursos das comunicações, de forma a garantir algum controlo da qualidade de serviço disponibilizada às aplicações, também é muito importante nas redes Ad Hoc.

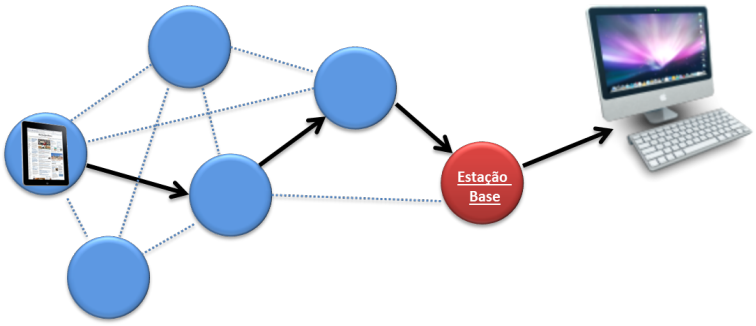


Figura 1.3: Rede AdHoc com ligação a uma rede sem fios com infraestrutura (Híbrida)

Analisando os estudos já elaborados sobre este tema, identificam-se como principais desafios das redes MANET os seguintes aspetos[15]:

1. Encaminhamento *Unicast*

2. Encaminhamento *Multicast*
3. Dinamismo da topologia de rede
4. Velocidade dos nós
5. Frequência de atualizações ou sobrecarga da rede
6. Escalabilidade
7. Qualidade de Serviço QoS
8. Eficiência Energética

1.2 Motivação

Como descrito na secção anterior, as redes Ad Hoc são um tipo de rede sem fios que não necessita de uma infraestrutura de rede pré configurada, onde não existe um controlo centralizado e é possível a sua implementação em qualquer ponto e situação. Deste modo existe um grande interesse no desenvolvimento de metodologias e protocolos de encaminhamento funcionais e eficazes para a implementação destes tipos de redes e da ligação de vários dispositivos em qualquer situação, por mais irregular que seja. Visto que os protocolos de encaminhamento são essenciais para o desempenho das redes sem fios Ad Hoc, o desenvolvimento e estudo de novos protocolos requer comparação com protocolos já bem conhecidos. Pretende-se fazer um estudo comparativo do desempenho dos protocolos de encaminhamento Ad Hoc, em diferentes cenários de utilização e com diferentes aplicações, de forma a podermos identificar possíveis pontos fortes e pontos fracos da sua utilização, para que num futuro e com novas propostas tal possa ser melhorado.

1.3 Objetivos do trabalho

Com a realização deste trabalho pretende-se desenvolver um *testbed* de simulação para avaliação de vários protocolos Ad Hoc, em diferentes cenários de simulação e vários tipos de aplicações. Com os resultados obtidos nas simulações, pretende-se

fazer uma análise do desempenho no encaminhamento de tráfego para cada protocolo Ad Hoc analisado. A análise de desempenho dos diferentes protocolos de encaminhamento atuais, proporciona uma linha de base de comparação para novos casos.

As principais etapas na realização deste trabalho são as seguintes:

1. Realização de um levantamento do estado da arte ao nível de encaminhamento em redes Ad Hoc;
2. Descrição das características do funcionamento e implementação de uma rede Ad Hoc;
3. Compreensão ao pormenor do funcionamento dos protocolos em redes Ad Hoc;
4. Desenvolvimento de aplicações e cenários diferentes para testes dos protocolos a analisar;
5. Estabelecer um *testbed* (simulado) que permita testar soluções de encaminhamento em redes Ad Hoc;
6. Testar e avaliar as soluções estudadas no *testbed* estabelecido;
7. Propor e avaliar alterações às soluções estudadas, no sentido de procurar melhorar o seu desempenho e estabilidade.

1.4 Contribuições

Nesta dissertação faz-se o estudo dos protocolos de encaminhamento de tráfego em redes Ad Hoc, mais propriamente o desempenho dos protocolos AODV, OLSR e DSDV. Existem já alguns estudos e desenvolvimentos sobre as redes Ad Hoc [44] [10] [15], principalmente no que se refere aos protocolos de encaminhamento neste tipo de redes e às suas aplicações em diferentes cenários. Alguns artigos publicados fazem estudos de desempenho destes modelos de rede, tais como [20] [31] [19] e estudos sobre a mobilidade dos dispositivos [17] e desempenho dos protocolos de encaminhamento Ad Hoc. Relativamente aos protocolos estudados, verifica-se que alguns deles já foram aceites como protocolos padrão,

sendo publicados como RFC's (*Request for Comments*) por parte do IETF(*Internet Engineering Task Force*), nomeadamente o protocolo AODV [16] e o protocolo OLSR [21]. Neste trabalho foi então desenvolvido um *testebed* de simulação, com o desenvolvimento de uma script de teste, onde é criada uma topologia de rede de dispositivos sem fios para se fazer a simulação da comunicação entre os dispositivos utilizando os protocolos Ad Hoc estudados. Para esta realização preparou-se um ambiente de simulação que foi executado e testado recorrendo às funcionalidades disponibilizadas de um *cluster*. Os resultados obtidos foram analisados de modo a comparar o desempenho dos diferentes protocolos em diferentes cenários.

1.5 Estrutura do documento

Este documento encontra-se estruturado em 6 capítulos. Neste capítulo 1 apresenta-se um enquadramento do tema, com uma descrição sucinta das redes sem fios Ad Hoc e do encaminhamento nas mesmas. Inclui ainda uma descrição da motivação, dos objetivos e principais contribuições.

No capítulo 2 apresenta-se uma exposição detalhada das redes Ad Hoc, descrevendo a comunicação nas redes sem fios Ad Hoc, mostrando quais as suas principais características, como se classificam, os cenários de aplicação, o modo como é realizado o encaminhamento neste tipo de redes, as suas limitações e as principais vantagens e desvantagens.

No capítulo 3 apresenta-se uma descrição detalhada dos principais protocolos nas redes Ad Hoc, incidindo na exposição de todas as características e funcionalidades de cada protocolo analisado.

No capítulo 4, sendo este o capítulo principal desta dissertação, é exposto todo o projeto desenvolvido, sendo descrita e defendida a escolha do simulador utilizado e de todos os processos desenvolvidos, desde os modelos de mobilidade às aplicações. Descreve-se ainda toda a *script* desenvolvida para a realização dos testes de desempenho dos protocolos estudados.

No capítulo 5, são apresentados todos os resultados obtidos nas simulações realizadas. Os resultados são analisados e comparados para cada protocolo e para

cada cenário gerado.

Finalmente, no capítulo 6, faz-se uma síntese de todo o trabalho desenvolvido, apresentam-se as principais conclusões obtidas e perspetiva-se o trabalho futuro.

2

Redes Ad Hoc

Este segundo capítulo está dividido em 7 secções e apresenta uma análise mais detalhada das redes Ad Hoc. Na primeira secção aborda-se a comunicação e as normas de referência para a comunicação sem fios. Na secção seguinte faz-se uma síntese das principais características das redes Ad Hoc. Na terceira secção mostram-se os principais parâmetros que podem ser usados na sua classificação enquanto na quarta secção se descrevem os cenários possíveis nos quais poderão ser utilizadas as redes Ad Hoc. A quinta secção descreve o encaminhamento neste modelo de redes, descrevendo as principais categorias dos protocolos nas redes Ad Hoc. O capítulo termina com um resumo das vantagens, limitações e principais desafios deste tipo de redes.

2.1 Comunicação em redes Ad Hoc

Uma rede Ad Hoc resulta da comunicação entre vários dispositivos móveis que cooperam entre si para encaminharem o tráfego pela rede. Neste tipo de redes a topologia pode mudar dinamicamente, podendo ser definida como um grupo autónomo de nós móveis que comunicam entre si através de ligações sem fios. Um dos principais desafios neste tipo de redes é pois a eficiência do encaminhamento de tráfego na rede.

Em 1997, o IEEE lançou a primeira norma de comunicações sem fios(wireless) 802.11, sendo esta composta pela camada de acesso ao meio MAC e pela camada física. Esta norma veio inserir-se na família de padrões IEEE 802, como é apresentado na figura 2.1. As motivações que estiveram por trás da introdução desta norma foram: a oferta de serviços, que até ao momento só estavam disponíveis nas redes com fios, a oferta de alto rendimento com aceitável confiabilidade e a disponibilização de ligação continua à rede para todos os utilizadores.

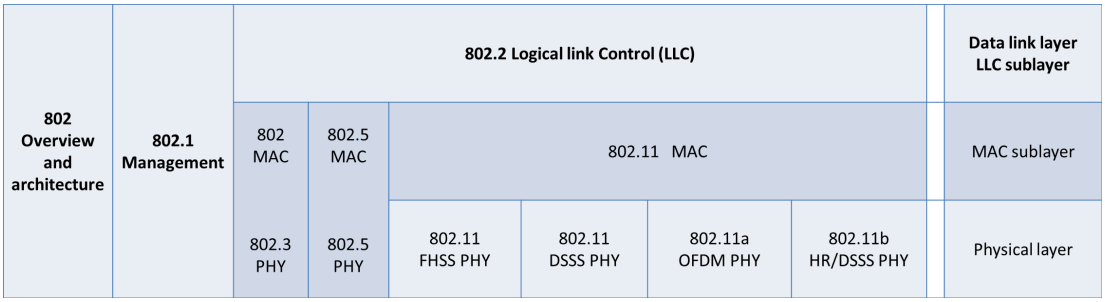


Figura 2.1: Familia IEEE 802 e sua relação com o modelo OSI

O nome 802.11[30] é a designação para a norma desenvolvida pelo grupo IEEE(*Institute of Electrical and Electronics Engineers*) para as redes sem fios, designadas pela sigla WI-FI, sendo esta uma abreviatura do termo inglês *"Wireless Fidelity"*. Através desta tecnologia é possível criar redes de computadores e dispositivos compatíveis(*smartphones*, *tablets*, consolas de jogos, impressoras, etc) desde que estejam próximos geograficamente e dentro da área de cobertura dos dispositivos vizinhos. Este sistema de numeração 802 vem do IEEE, para designar os padrões de redes de computadores, incluindo o Ethernet 802.3.

A norma 802.11 define as regras de comunicação para as redes de comunicação

sem fios(WLANs). A comunicação neste tipo de redes é realizada através de sinais de radiofrequência, que se propagam pelo meio e podem cobrir áreas de centenas de metros. Consequentemente, surgem algumas vertentes sobre a norma 802.11, que se diferenciam pelas principais características como a velocidade de transmissão, o alcance e o intervalo de frequências.[30] Assim, obtemos uma lista com cada um dos padrões *wireless* IEEE mais utilizados atualmente, realçados na tabela 2.1.

Standard	Descrição
IEEE 802.11	É a primeira versão da norma 802.11 capaz de transmissões de 1 a 2 Mbps, operando na faixa de frequências de 2.4GHz.
IEEE 802.11a	Capaz de transmitir até 54Mbps e opera na gama de frequências 5GHz.
IEEE 802.11b	Capaz de transmitir até 11Mbps, operando na banda de frequências de 2.4GHz.
IEEE 802.11g	Capaz de transmitir até 10Mbps e opera em gamas de frequência de 2.4GHz, 3.6GHz e 5 GHz.
IEEE 802.11n	Este padrão opera com larguras de banda de 2.4GHz e 5GHz, utilizando antenas de entrada e saída múltipla(MIMO) para melhorar velocidades de transferência de dados.

Tabela 2.1: Padrões IEEE 802.11

Da mesma forma que as restantes normas da família IEEE 802 definem a sua camada física (PHY), nível 1 do modelo OSI, e a camada de ligação, respetivamente o nível 2 do modelo OSI, representado na figura 2.2, a norma 802.11 tem como especificação a seguinte:

CAMADA APLICAÇÃO	Módulos de programas de aplicação que utilizam a rede.
CAMADA APRESENTAÇÃO	Trata da estandardização dos dados às aplicações.
CAMADA DE SESSÃO	Trata da gestão das ligações entre aplicações cooperativas.
CAMADA DE TRANSPORTE	Proporciona serviços de deteção e correção de erros.
CAMADA DE REDE	Trata da gestão das ligações através da rede para as camadas superiores.
CAMADA DE LIGAÇÃO	Proporciona serviços de envio de dados através da ligação.
CAMADA FÍSICA	Define as características físicas da rede (estrutura).

Figura 2.2: Camadas do modelo OSI e sua descrição

- Camada física (PHY)

  A camada física refere-se ao nível mais baixo do modelo OSI, definindo a especificação elétrica e física dos dispositivos. De um modo geral, esta camada consiste na relação entre o meio de transmissão e o dispositivo. É através desta camada que são definidas as técnicas de modulação, permitindo que várias estações estejam a comunicar sobre a mesma banda de frequências, com o mínimo de interferências entre si.

- Camada de ligação

  A camada superior à camada física, camada de enlace ou ligação de dados, é dividida em 2 sub-camadas sendo elas a MAC(*Media Access Control*) e LLC(*Logic Link Control*). Isto porque esta é a combinação da camada de acesso ao meio e a camada de ligação lógica respetivamente. Tem como principal característica a deteção, e opcionalmente, a correção de erros que possam chegar do nível físico.

2.2 Características de uma rede Ad Hoc

Como já foi referido, uma rede Ad Hoc consiste na cooperação entre um con-

junto de nós móveis para comunicarem entre si, sem a necessidade de uma infraestrutura ou de um servidor central que faça encaminhamento de dados entre os dispositivos. Os dispositivos ligados a esta rede funcionam como encaminhadores, encaminhando o tráfego nó a nó por toda a rede.

Os dispositivos numa rede do tipo Ad Hoc, "nós", podem movimentar-se livremente pelo espaço da rede, fazendo com que a estrutura/topologia da rede esteja em constante mudança, alterando a sua topologia de forma imprevisível. Este ponto é uma das principais características que distinguem as redes Ad Hoc das redes convencionais com infraestrutura. Uma outra característica das redes Ad Hoc é o encaminhamento de pacotes efetuado ao nível 3, utilizando protocolos de encaminhamento Ad Hoc.

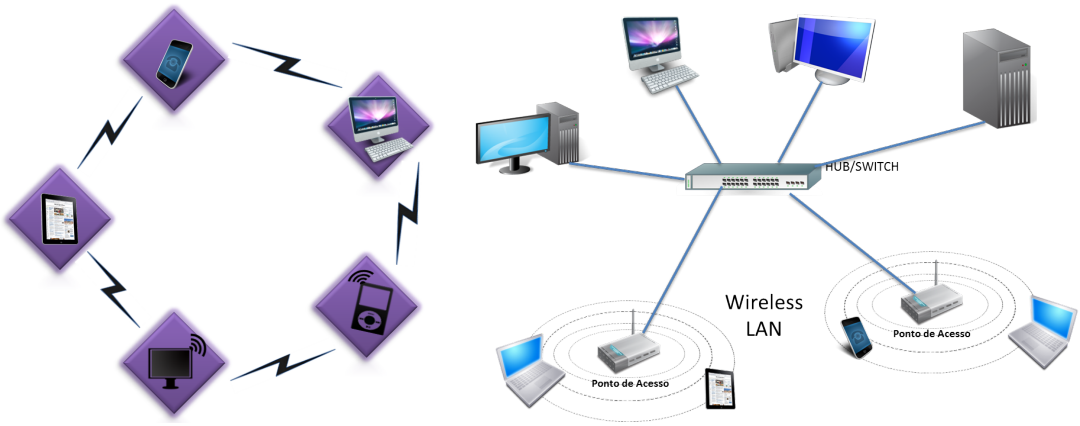


Figura 2.3: Comparação entre uma rede Ad Hoc e uma rede com infraestrutura

Na figura 2.3 pretende-se mostrar a comparação entre uma topologia de rede Ad Hoc isolada, de uma topologia de rede infraestruturada.

Para que seja possível a comunicação numa rede Ad Hoc, os nós são "obrigados"a cooperarem para que sejam disponibilizados os serviços mínimos necessários na rede, formando um sistema autónomo com nós móveis. Um nó é um *host*, com capacidade de comunicação sem fios. Um conjunto de nós pode formar uma rede isolada, isto é, um grupo de nós separados de outras possíveis redes, não conseguindo comunicar com redes próximas. Por outro lado, podem formar redes com conexão à Internet ou a outras redes vizinhas.[22]

Dependendo da posição dos nós ou da área de cobertura de cada dispositivo

de rede sem fios, a topologia de rede varia constantemente à medida que os nós vão variando a sua posição. Este processo de mobilidade dos nós pelo espaço de cobertura estabelecido entre eles, faz com que se obtenha efeitos secundários e problemas na comunicação, provocados essencialmente pela dinâmica da topologia, da limitação de largura de banda, da energia dos dispositivos e da segurança física na comunicação.

As principais características de uma rede Ad Hoc são:

- Redes sem fios, a qual não necessita de um ponto de acesso comum entre os dispositivos de rede;
- Não necessita de uma infraestrutura de rede(*backbone*) configurada inicialmente;
- Os dispositivos movimentam-se livremente pelo meio;
- A topologia de rede muda constantemente e de forma imprevisível;
- Requer uma permanente adaptação de rotas e reconfiguração das tabelas de encaminhamento;
- A possibilidade de dois nós comunicarem diretamente entre si,
  - no caso de um nó destino se identificar como vizinho de um nó fonte, utiliza-se o método de "*single hop*"para troca de dados, sendo efetuada a comunicação direta entre os dispositivos.
- Os dispositivos pertencentes à rede funcionam como encaminhadores *routers*, encaminhando os pacotes entre si;
- As ligações entre um conjunto de nós são independentes das restantes ligações estabelecidas na rede,
  - isto indica que se uma ligação falhar, por problemas de ligação ou mesmo por falha do dispositivo, as outras ligações continuam a funcionar normalmente.

No entanto, para o funcionamento deste tipo de redes, devem ser respeitados alguns requisitos específicos. Os nós da rede têm de identificar-se perante todos

os outros nós, normalmente pelo endereço IP. Este endereço é obtido através da utilização de protocolos de auto-configuração [11], o que faz com que os nós sejam capazes de alcançar outros nós na rede e assim sejam realizadas funções de encaminhamento de tráfego entre eles. Para além do encaminhamento de mensagens, os nós devem ser capazes de saírem da rede e novos nós de entrarem, sem causarem prejuízos na comunicação.

Estes requisitos são, na maior parte das vezes, colmatados com o funcionamento de protocolos dedicados ou através da combinação de diferentes protocolos. A utilização de vários protocolos faz com que sejam combinadas funcionalidades extras, resultado de uns serem mais eficazes no processo de comunicação entre os nós, e outros no fornecimento de serviços.

Numa rede sem fios, constituída por vários intervenientes, o caminho que um pacote deve seguir desde a sua origem até ao seu destino é estabelecida através da execução de rotinas de encaminhamento, processos estes que descrevem a sequência de nós que um pacote deve percorrer desde a sua origem até ao último ponto, destino. Contudo, a função do protocolo de encaminhamento é determinar a melhor rota, respeitando as métricas definidas, até ao destino pretendido.

Nas redes fixas ou estruturadas, os protocolos de encaminhamento determinam as rotas durante a inicialização da comunicação, sofrendo poucas ou mesmo nenhumas alterações nas tabelas de encaminhamento, até ao final da comunicação. Isto em redes Ad Hoc é muito pouco provável acontecer, a mobilidade dos nós faz com que o caminho entre o nó fonte e o nó destino esteja em constante alteração, o que faz com que as rotas tenham de ser recalculadas, e as tabelas de encaminhamento sofram constantes atualizações. Um dos principais desafios neste tipo de redes é pois a eficiência do encaminhamento de tráfego na rede, devido à topologia de rede ser dinâmica e à quantidade de largura de banda necessária na comunicação entre dispositivos.

O encaminhamento envolve duas operações básicas: a determinação dos melhores caminhos e o transporte dos pacotes na rede. As características desejáveis dos algoritmos de encaminhamento são a escolha do melhor caminho de uma origem até ao destino, a simplicidade do algoritmo, a robustez do algoritmo, a sua

imparcialidade, a estabilidade, a rapidez de convergência para o melhor caminho, a flexibilidade, a aceitação de parâmetros de QoS e independência da tecnologia da rede.

As abordagens mais clássicas ao problema do encaminhamento são baseadas na inundação da rede (*flooding*), estado de ligação (*link state*) e vetor distância (*distance vector*), que são os três algoritmos fundamentais para a compreensão do encaminhamento, também nas redes AdHoc.

2.3 Classificação de redes Ad Hoc

As redes Ad Hoc podem ser diferenciadas e classificadas consoante diversos parâmetros. Estes parâmetros classificam as redes Ad Hoc de acordo com as suas características de implementação e de funcionamento, podem ser então classificadas por:

- Tipo de comunicação:

  O tipo de comunicação entre dois dispositivos vizinhos ou diretos, pode ser classificado como comunicação direta "*single hop*" ou comunicação através de múltiplos saltos "*multi hop*".

  Single hop, cada dispositivo comunica apenas com os que estão ao seu alcance, aqueles que são identificados como seus vizinhos diretos;

  Multi hop, dois utilizadores comunicam entre si através de dispositivos encaminhadores, que encaminham o tráfego de um ponto ao seu extremo. Desta forma existe uma sequência de dispositivos através dos quais os pacotes são encaminhados até que seja alcançável o destinatário. Tem por base essencialmente a comunicação entre dois dispositivos através de nós intermediários, sendo necessário múltiplos saltos desde a fonte ao destino. Assim, a comunicação entre dois dispositivos não fica tão limitada ao raio de abrangência de cada dispositivo, mas sim à soma dos raios de abrangência de todos os dispositivos envolvidos.

- Heterogeneidade:

  Consoante vários parâmetros dos dispositivos da rede, a rede pode ser classi-

ficada como simétrica ou assimétrica. Nas redes simétricas, os nós constituintes têm igual capacidade de processamento, sendo caracterizados por diferentes parâmetros que os distinguem das redes assimétricas. Estes parâmetros distinguem os dispositivos das redes simétricas dos das redes assimétricas, porque permitem transmitir e receber pacotes a taxas médias iguais nos dois sentidos de comunicação. Por outro lado, compartilham os mesmos valores de alcance de transmissão, capacidade de processamento, forma de encaminhamento, entre outros. Nas redes assimétricas, os nós têm capacidades de processamento diferentes, logo de nó para nó da rede podemos verificar diversos raios de transmissão/abrangência, com modos de encaminhamento de tráfego diferentes, utilizando taxas de transferências diferentes, entre outros. Estes parâmetros devem-se à possível utilização de diferentes protocolos de rede e com variações de velocidades de movimento na rede. Concluindo, nas redes assimétricas cada nó pode ser responsável por funcionalidades distintas na rede.

- Tipo de tráfego a transmitir:

  O tipo de tráfego a transmitir, neste tipo de redes, não difere muito ao tráfego das redes convencionais, podendo ser tráfego de dados normais(texto, imagens, som, vídeo, etc) ou dados de tempo real para aplicações multimédia(som e vídeo). O que difere principalmente é o modo de transmissão e o funcionamento dos protocolos utilizados pelas diferentes camadas de nós na rede, que são geralmente modeladas para se adaptarem ao tráfego a transmitir.

- Método de encaminhamento aplicado:

  De um modo geral, nas redes Ad Hoc podemos classificar o tipo de encaminhamento utilizado como: encaminhamento *unicast*, encaminhamento *multicast* e encaminhamento *geocast* (Figura 2.4). O encaminhamento *unicast* consiste no reencaminhamento de tráfego com destino a um só dispositivo num conjunto de redes, a partir de um nó origem para um nó destino. Por outras palavras, este tipo de encaminhamento consiste no envio de tráfego para um ponto final na rede. O encaminhamento *multicast* refere-se ao en-

caminhamento de tráfego desde uma fonte para múltiplos destinos(grupo) simultaneamente, e onde os pacotes só passam por uma interface uma única vez, sendo duplicada a informação apenas quando a interface para os destinos se divide em duas direções diferentes. O encaminhamento *multicast* consiste na propagação das informações de escuta do grupo *multicast*. Por fim, o encaminhamento *geocast* consiste no encaminhamento de informação a um grupo de nós destino, identificado através da sua localização geográfica. Ao contrário do *multicast* que permite o encaminhamento de pacotes para um grupo arbitrário de nós, o *geocast* só é direcionado para um grupo de nós de uma região geográfica definida. De notar que o *geocast* é uma subclasse de *multicast*, podendo ser implementado como um serviço *multicast* em que é necessário apenas definir o grupo *multicast* para uma determinada região geográfica. Relativamente às redes Ad Hoc do tipo VANET(*Vehicular ad-hoc network*), este método de encaminhamento pode tornar-se eficiente, podendo a informação geográfica(posição e local) facilitar no encaminhamento dos pacotes para os destinatários [37].

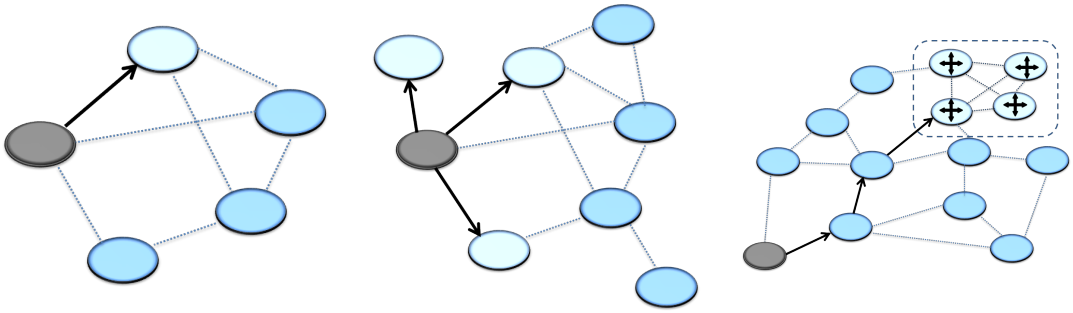


Figura 2.4: Encaminhamento Unicast, Multicast e Geocast

- Métodos de endereçamento:

  Neste tipo de rede, o modo de endereçamento dos dispositivos ao longo da rede pode ser estabelecido de diferentes formas, baseado no *host*, no conteúdo ou mesmo na capacidade de transmissão que cada nó disponibiliza.

Para além destes parâmetros anunciados, podemos ainda classificar as redes Ad Hoc consoante a variação de algumas métricas, tais como, a taxa de transmissão, a mobilidade dos dispositivos, os requisitos de segurança, qualidade de serviço, entre

outras.

2.4 Cenários de aplicação de redes Ad Hoc

Na atualidade, com o avanço tecnológico, com o desenvolvimento dos dispositivos móveis e de comunicação sem fios, as aplicações de redes Ad Hoc tornaram-se muito mais abrangentes e úteis no dia a dia. Devido à já referida mobilidade dos dispositivos e ausência de infraestrutura de rede, vários cenários justificam a implementação de redes do tipo Ad Hoc. Normalmente, as redes deste tipo são configuradas em locais onde não existe uma infraestrutura de rede previamente instalada, uma infraestrutura de rede com base num ponto de acesso comum aos dispositivos(*base station*). Contudo, podemos obter cenários básicos de comunicação entre dois nós da mesma rede ou comunicação de um nó com outro nó de uma outra rede, podendo por vezes ser necessário a ligação da rede com a rede fixa(infraestruturada). De acordo com a situação para qual a rede é necessária, resultam distintos tipos de cenários de aplicação da rede, sendo eles:

- Rede de sensores sem fios (RSSF)

  As redes de sensores sem fios são consideradas uma sub-classe das redes Ad Hoc. Estas redes consistem num grupo de sensores ligados entre si, com o objetivo de monitorizar algum fenómeno. Estas redes têm ganho muita importância devido a terem grande aplicação em locais de difícil acesso ou áreas perigosas, tais como:

  Militar - utilizadas em funções do tipo de monitorização, rastreamento, segurança, controlo e manutenção;

  Industrial - são utilizadas particularmente para funções de monitorização em áreas de difícil acesso;

  Aviação - substituindo as redes com fios, como ainda são utilizadas atualmente;

  Ambiente - de forma a monitorizar variáveis ambientais em habitações, construções, florestas, oceanos,etc;

Tráfego - monitorizando o tráfego nas vias rodoviárias, nos parques de estacionamento, etc;

Engenharia - monitorização e modelagem de estruturas;

- Rede de área pessoal, PAN(*Personal Área Network*)
  Este tipo de rede de área pessoal é um tipo de aplicação de rede Ad Hoc para dispositivos portáteis de pequenas dimensões, sendo principalmente uma rede de uso pessoal, uma rede criada em locais pessoais, como em casa. Habitualmente são utilizados dispositivos como PDA(Personal Digital Assistant), *laptops*, telemóveis, *tablets*, *netbooks*, entre outros com ligação sem fios.
- Rede para fins militares
  Este tipo de cenário ocorre em locais onde poderá ter sido destruída a infraestrutura de rede ou onde é impossível a implementação de uma infraestrutura, como por exemplo em campos de batalha ou na comunicação entre uma frota de navios no mar, permitindo aos utilizadores comunicarem entre si e com as centrais(bases militares).
- Rede para fins civis
  Como não é necessário a criação de uma infraestrutura para obtermos comunicação entre vários dispositivos, este tipo de redes são bastante úteis e funcionais para fins civis. Deste modo podemos utiliza a rede em cenários de eventos em grande escala(ralis, estádios desportivos, festivais, manifestações, etc), em redes de automóveis(táxis, veículos de emergência, etc), em instituições(universidades, escolas, hospitais, etc), em salas de reuniões e em grupos de trabalhos empresariais(palestras, *workshops*, conferências, etc).
- Operações de emergência ou catástrofe
  Por fim, um cenário na qual não será possível obter comunicação a não ser utilizando comunicação por satélite, com agravante de possuir custos elevados, é o caso de cenários de tragédia, em que não é possível qualquer comunicação entre dispositivos devido à falha da infraestrutura ou destruição da infraestrutura de rede, e no qual a rede pode ser implementada e

configurada através do meio livre(ar). Para este caso temos como exemplos, cenários de tragédia ambiental, busca e salvamento, operações dos organismos de segurança(policia, bombeiros), monitorização e vigilância, e qualquer outra situação de quebra da infraestrutura de rede.

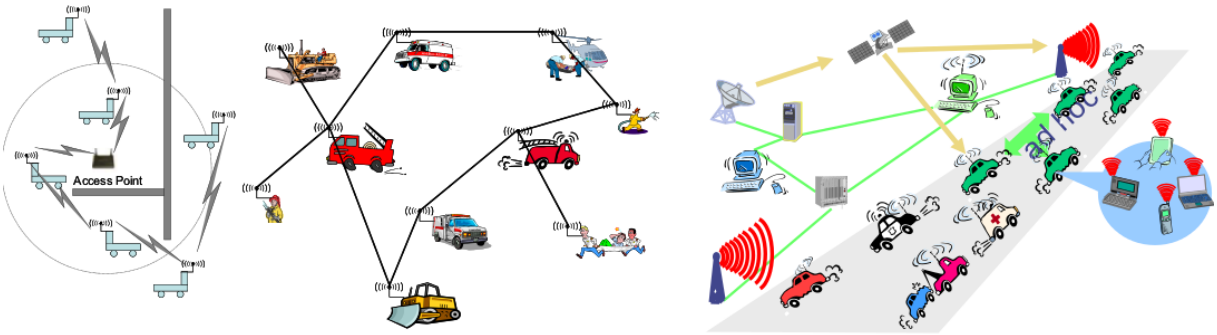


Figura 2.5: Cenários de aplicação de redes Ad Hoc

Ao serem estudadas soluções para existência de comunicação na ocorrência de tragédias que destruam ou impossibilitam o decorrer das comunicações estabelecidas anteriormente, surgem as redes Ad Hoc. Inicialmente, foram apenas pensadas para serem usadas em caso de catástrofes, onde seria impossível manter uma rede fixa de apoio, como em situações de resgate, catástrofes naturais, aplicações militares, entre outras.

2.5 Encaminhamento em redes Ad Hoc

Como as redes Ad Hoc são redes dinâmicas em termos de tamanho, e onde existe uma variação constante de dispositivos na rede, os protocolos utilizados nas redes convencionais, por cabo ou redes fixas, não são adequados para realizarem o encaminhamento de dados corretamente nesta situação.

O encaminhamento de tráfego neste tipo de redes é o ponto principal de estudo. O encaminhamento dos dados é a principal função da camada de rede, esta função consiste essencialmente na determinação dos melhores caminhos, entre uma fonte e um destino, e no correto transporte dos pacotes até ao destino pretendido.

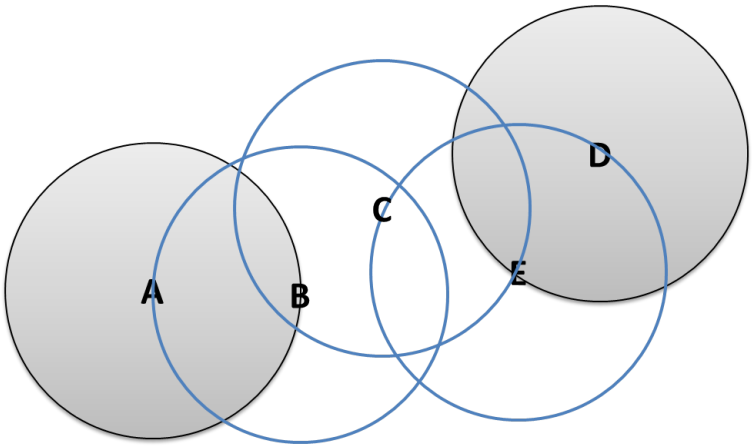


Figura 2.6: Exemplo de disposição dos nós de uma rede Ad Hoc entre uma fonte A e o destino D.

Este processo consiste no encaminhamento de pacotes de dados desde a fonte, através de vários nós na rede, até ao destino pretendido (Figura 2.7), passando estes dados por diferentes dispositivos de encaminhamento, que poderão utilizar diferentes tipos de ligações e diferentes características propostas para redes do tipo Ad Hoc. Para que a entrega seja efetuada corretamente, e dentro de tempo útil, vários protocolos foram desenvolvidos e muitos outros estão ainda a serem estudados.

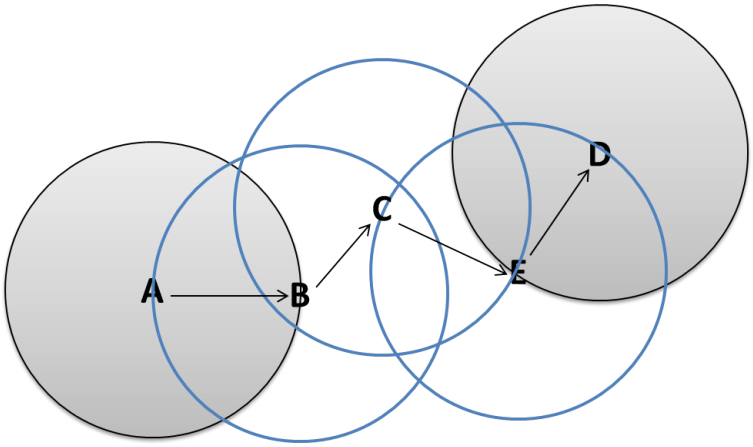


Figura 2.7: Comunicação entre dispositivos origem-destino.

De acordo com vários estudos realizados nos últimos anos, vários protocolos são sugeridos consoante o ambiente em que se aplicam. Os protocolos de encami-

nhamento para redes Ad Hoc agregam-se em duas principais categorias:

- Protocolos de encaminhamento Pro-Ativo (OLSR, DSDV):

  São protocolos de encaminhamento que estão ativamente a determinar a topologia da rede e que obtêm uma visão geral da rede, através da construção de tabelas de encaminhamento. Cada nó da rede tem uma visão geral de toda a topologia de rede, de todos os caminhos para qualquer destino. Isto é conseguido devido às constantes trocas de mensagens entre os nós da rede, e à permanente troca de tráfego de controlo para calcular caminhos otimizados. Este tipo de protocolo torna-se pouco funcional em redes de sensores, quando os nós têm baixas capacidades e quando se pretende obter baixos consumos de energia, o que não se verifica com o funcionamento destes protocolos e com as constantes atualizações das tabelas de encaminhamento. Por outro lado, quando se obtém uma rede com um elevado número de nós, o uso destes protocolos pode não ser eficaz. À medida que o número de nós vai aumentando verifica-se um aumento do número de pacotes de controlo, trocados entre os nós da topologia e uma grande necessidade de memória, por serem criadas muitas tabelas de encaminhamento.

- Protocolos de encaminhamento Reativo/*On demand* (AODV):

  Os protocolos de encaminhamento do tipo reativo, são protocolos que são adequados para topologias de rede de grandes dimensões, isto porque "inundam"a rede com mensagens de controlo *Route Request*(RREQ) para obterem o melhor caminho para o destino, apenas quando é requerida uma comunicação. Este processo introduz algum atraso no envio de mensagens, porque os nós têm de calcular sempre a rota antes de qualquer envio. Quando não existem pacotes a serem encaminhados na rede, os nós da rede não necessitam de estar constantemente a trocar informação sobre a topologia de rede ou sobre algum destino. As rotas são apenas estabelecidas quando é necessário o envio de dados, neste momento os nós da rede trocam informação entre si, com a finalidade de encontrarem a melhor rota até ao destino. Este tipo

de protocolo é mais eficiente no que toca a consumos de energia, principalmente por ser trocada informação entre os nós apenas quando é necessária uma comunicação e devido ao método de estabelecimento de rotas ser mais simples. Uma desvantagem destes tipos de protocolos é a elevada latência no estabelecimento de rotas.

Com isto, obtemos uma diminuição de *overhead*, visto que apenas são trocados pacotes de controlo na rede quando é solicitada uma comunicação.[15][36]

Para concluir, apresentam-se a seguir algumas características que se deve ter em atenção de modo a obter um encaminhamento eficiente, com objetivo de minimizar os problemas impostos pela mobilidade dos dispositivos.

- Escolha da rota

  Deve fazer sempre a escolha da melhor rota para o qual irá encaminhar os pacotes de dados;

- Simplicidade do algoritmo

  Fornecendo os serviços necessários ao sistema, com a quantidade mínima de processamento possível;

- Robustez

  O algoritmo de encaminhamento deve obter sempre os melhores resultados e funcionar corretamente, esperando que a rede funcione corretamente e sem interrupções durante um largo período de tempo;

- Escalabilidade

  O algoritmo de encaminhamento deve responder adequadamente e continuar a responder da melhor forma mesmo aumentando o número de nós na rede;

- Convergência

  O algoritmo deve obter uma convergência para um caminho ótimo, escolhendo rapidamente a melhor e a mais atual rota para atingir um destino;

- Adaptabilidade

  Verificando que as redes Ad Hoc têm uma topologia constantemente dinâ-

mica, os algoritmos de encaminhamento devem ser capazes de funcionar corretamente com as frequentes mudanças da topologia de rede;

- Independência das tecnologias

  É importante a independência das tecnologias utilizadas, fazendo com que os algoritmos funcionem nos diferentes dispositivos e meios físicos com que se deparam;

- Aceitação de parâmetros de QoS

  Em alguns cenários de rede é imprescindível o suporte de parâmetros de QoS, de forma a combater perda de pacotes, atrasos, baixa taxa de transferência e erros de transmissão.

As principais abordagens, relativas ao problema do encaminhamento em redes de comunicação, são sem dúvida o "inundamento"(*flooding*), o estado de ligação(*link state*) e o vetor distância(*distance vetor*). Estes algoritmos de encaminhamento são fundamentais para entendermos o funcionamento do encaminhamento nas redes Ad Hoc.

Flooding é uma das abordagens mais simples de todas, que consiste numa "inundação"de pacotes na rede num curto espaço de tempo. Cada vez que um nó na rede pretende comunicar, faz um *flooding* na rede de pacotes a transmitir, e neste caso, sempre que um nó recebe um desses pacotes, verifica se não é ele o destino desse pacote, e se ele não for o destino, reenvia esse pacote por todos os canais a que está ligado, com a exceção do nó através do qual recebeu o pacote. Esta abordagem é simples mas eficaz apenas em algumas situações, tendo como problema o caso de um nó poder em algum momento receber o mesmo pacote mais do que uma vez. Através deste algoritmo, garante-se que o primeiro pacote a chegar ao destino vai percorrer sempre o melhor caminho. Um dos problemas deste algoritmo é o uso excessivo de pacotes pela rede, verificando que cada nó interveniente que recebe um pacote que não seja para si, vai reencaminha-lo por todas as suas ligações, gerando aqui um problema de aumento do tráfego na rede e de escalabilidade, que com o aumento desta, a largura de banda necessária a este processo o torna quase proibitivo. Outro problema é os *loops* que possam ocorrer

ao longo do percurso, fazendo com que por vezes fiquem pacotes a transitar na rede indefinidamente. Para combater, em parte, os ciclos de pacotes a circularem na rede, os pacotes transmitidos são compostos por um campo TTL(*Time To Live*), que define o tempo de vida do pacote na rede. Quando esse tempo expirar esse pacote é retirado de circulação na rede.

O algoritmo de Estado de Ligação (*Link State*) baseia-se no conhecimento total da topologia. Um encaminhador comunica com os restantes dispositivos na rede, identificando quais são os seus vizinhos e a que distância este nó está dos seus vizinhos. Quando um nó se dá conta de uma alteração do estado dos links dos seus vizinhos, este nó faz *flooding* dessa alteração pela rede, de forma a que toda a rede tenha a mesma perceção da sua nova topologia. Assim, os restantes nós ficam a saber desta mudança quando recebem a notificação, podendo mudar a sua visão da rede. Assim, com toda a informação que um nó recebe dos seus vizinhos e dos vizinhos dos seus vizinhos, os nós constroem uma ideia geral da topologia da rede e como calcular os caminhos ótimos para chegar a um nó destino. Um dos problemas nas redes Ad Hoc é a falha no descobrimento da topologia de rede, possivelmente quando a rede se divide ao meio e posteriormente se liga novamente, criando novos *links* de ligação entre os nós. Esta abordagem é utilizada principalmente no protocolo OSPF, indicado por ter um grande potencial em encontrar caminhos através da utilização de critérios no momento de descoberta e pela sua rápida convergência.

Por fim, o algoritmo de Vetor Distância (*Distance Vector*) ou *Distributed Bellman-Ford* (DBF), como também é designado, define que cada um dos nós na rede contém uma tabela/vetor com o menor caminho, até todos os nós existentes na rede. Essa tabela é atualizada periodicamente através de mensagens recebidas pelos vizinhos. Recebida a informação de um vizinho, o nó compara os valores da sua tabela de encaminhamento com os valores recebidos e caso o valor de uma rota seja menor que o valor da sua tabela de encaminhamento, este atualiza essa rota e armazena de onde veio essa informação. O problema é que apresenta uma baixa convergência quando a topologia muda muito e quando a topologia começa a ficar com um elevado numero de nós. Por outro lado tem a tendência de criação

de *loops*, principalmente em condições não estáveis, como é o caso da constante alteração da topologia de rede que é que mais se espera no caso das redes Ad Hoc.

A maioria dos protocolos de encaminhamento propostos são baseados num destes algoritmos descritos. A escolha dos protocolos a estudar baseia-se essencialmente na importância que estes têm no desenvolvimento das redes Ad Hoc, importância essa que se pode medir com as referências feitas a esses protocolos e nos artigos publicados.

2.6 Vantagens, limitações e principais desafios das redes Ad Hoc

A utilização deste tipo de redes, onde os dispositivos se movem e simultaneamente têm de encaminhar dados pela rede, faz com que os dispositivos móveis apresentem várias limitações de recursos. Estas limitações baseiam-se, principalmente, na largura de banda disponível, na fonte de energia limitada(isto é, baterias), no raio de alcance da comunicação, na segurança contra ligações de dispositivos indesejáveis, no desempenho que cada nó deve oferecer, na taxa de debito(afetada pela carga do sistema), no número de nós da rede(em redes esparsas podem trazer problemas na comunicação), no atraso(em grandes redes pode afetar algumas das aplicações), entre outras, que podem fazer com que o desempenho destas redes diminua drasticamente.

Ao fazermos a comparação das redes Ad Hoc com as redes estruturadas, identificamos as principais vantagens e desvantagens respetivamente a estes tipos de redes. Relativamente às redes Ad Hoc, podemos descrever como suas vantagens e desvantagens as seguintes, descritas na tabela 2.2.

Comparando as vantagens das redes Ad Hoc com as suas principais desvantagens, verifica-se que a implementação de uma rede Ad Hoc pode ser bastante vantajosa para certos cenários ou tipos de aplicações.

Por um lado, as redes Ad Hoc são mais propícias a ataques de segurança do que as redes com fios, devido à possibilidade de qualquer nó, com dispositivos sem fios, poder ligar-se à rede ameaçando o correto funcionamento da rede através de

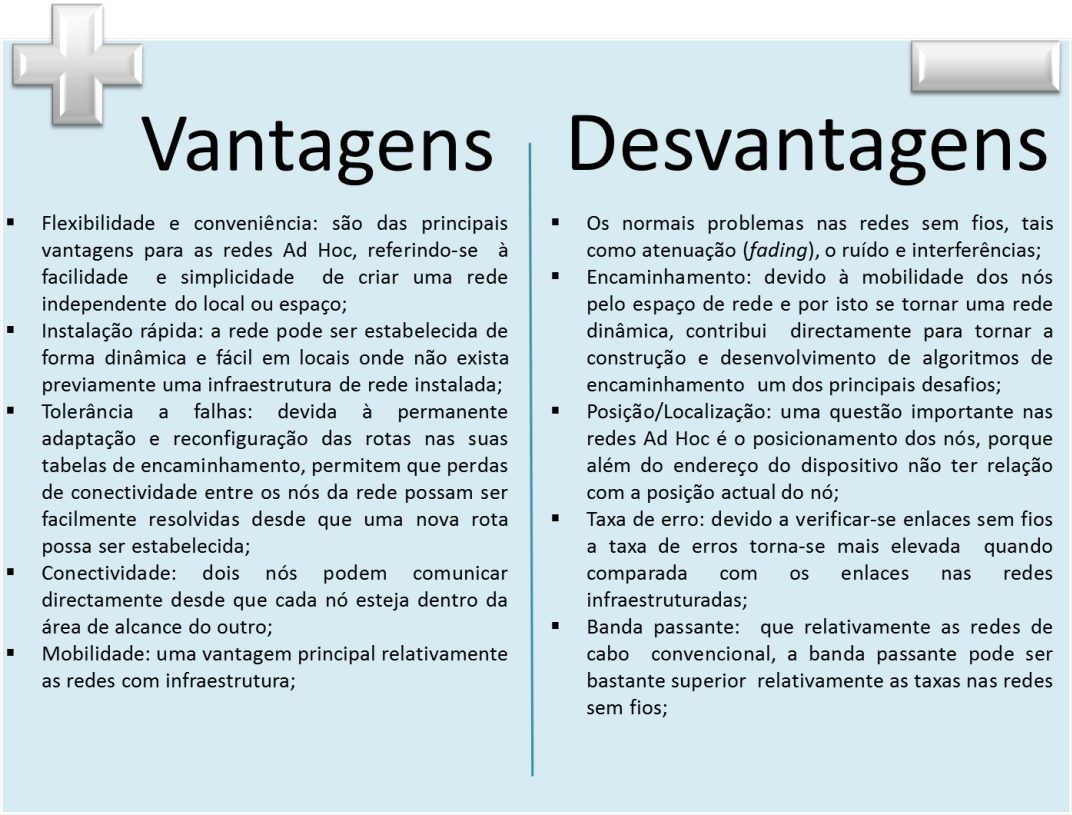


Tabela 2.2: Vantagens e desvantagens apresentadas pelas redes Ad Hoc

espionagem, *spoofing*[1] ou ataques do tipo *denial-of-service*[2].

Mas por outro, verifica-se que as redes Ad Hoc não dependem de alguns terminais críticos para determinar a sua organização e controlo na rede, o seu desempenho não é afetado se um determinado terminal falhar ou mesmo sair da rede. Podem ser adicionados facilmente novos terminais em qualquer momento na rede, sem se detetarem prejuízos na comunicação.

[1]É um método de se fazer passar por outra identidade, marcando pacotes com endereços de remetentes falsos

[2]Consiste num ataque de negação de serviço, sobrecarga do sistema, com o objetivo de fazer com que os recursos de um sistema estejam indisponíveis/bloqueados para os seus utilizadores.

2.7 Sumário

Neste capítulo fez-se numa apresentação das redes Ad Hoc, em particular das características específicas deste tipo de redes. Com base nelas foram apresentados os modos como são classificadas, alguns dos cenários de implementação, as suas limitações, vantagens e principais desafios deste modelo de redes sem fios.

3

Protocolos de encaminhamento nas redes Ad Hoc

Este terceiro capitulo está dividido em 5 secções e tem como objetivo fazer a apresentação do funcionamento de cada um dos protocolos de encaminhamento Ad Hoc estudados. Na primeira secção faz-se uma classificação dos protocolos, evidenciando o modo como eles se diferenciam no que diz respeito ao funcionamento do algoritmo utilizado. Nas secções seguinte descrevem-se, respetivamente, o processo de funcionamento dos protocolos AODV, OLSR e DSDV. O capitulo termina com uma síntese em forma de conclusão.

3.1 Classificação dos protocolos

Os protocolos de encaminhamento baseiam-se em algoritmos, alguns dos quais

já descritos na secção 2.5, que estabelecem rotinas para criar uma visão completa ou parcial da topologia de rede. Esta ideia da topologia de rede criada por cada nó, é concretizada através das tabelas de encaminhamento criadas por cada nó. A construção e manutenção das tabelas de encaminhamento é definida pelo modelo de encaminhamento que é característico de cada protocolo em uso.

O grupo de trabalho do IETF sobre as redes Ad Hoc, MANET IETF *working group*, tem como objetivo padronizar as funcionalidades dos protocolos de encaminhamento IP adequado para aplicações de encaminhamento em redes sem fios. O propósito maior é fazer com que os nós dentro de uma topologia sem fios possam encaminhar tráfego, usufruindo do maior dinamismo do seu livre movimento na rede[24].

Os principais protocolos de encaminhamento em redes Ad Hoc podem dividir-se em:

- *Table-driven* (designados por *Pro-active routing*, onde as rotas são previamente calculadas e armazenadas):
  É um tipo de protocolo onde o atraso na determinação de rotas é menor, podendo em contrapartida guardar rotas na tabela de encaminhamento nunca vão ser utilizadas.
- *On-demand* (designados por *Reactive routing*):
  É um tipo de protocolo onde as rotas apenas são estabelecidas quando solicitado por um nó fonte e onde o atraso na determinação de rotas pode ser elevado, devido ao facto do nós estarem sempre a determinar novas rotas. Tem como base a técnica de *flodding*.
- Híbridos :
  Estes tipos de protocolos combinam as vantagens dos protocolos pro-ativos e reativos.

Alguns dos protocolos mais conhecidos do mundo das redes Ad Hoc são então:

- Table-driven
  - OLSR (*Optimized Link State Routing*);

  - DSDV (*Destination Sequenced Distance Vector Routing*);
  - TBRPF (*Topology Broadcast based on Reverse-Path Forwarding*);
  - WRP (*Wireless Routing Protocol*);
  - CGSR (*Clusterhead Gateway Switch Routing*);

- On-demand
  - AODV (*Ad Hoc On-Demand Distance Vector Routing*);
  - DSR (*Dynamic Source Routing*);
  - LMR (*Lightweight Mobile Routing*);
  - TORA (*Temporally Ordered Routing Algorithm*);
  - ABR (*Associativity-Based Routing*);
  - SSR (*Signal Stability Routing*);

- Híbridos
  - ZRP (*Zone Routing Protocol*);

Uma grande quantidade de protocolos Ad Hoc são estudados/desenvolvidos e têm sido propostos [40], mas apenas quatro protocolos foram aceites como *Experimental* RFC (*Request For Comments*) pelo IETF, sendo eles o AODV [16], OLSR [21], TBRPF [41] e DSR [32].

No entanto não existe um consenso geral no uso de um protocolo específico, sendo implementado o protocolo dependente do cenário em que é concebida a rede. Até ao momento através das pesquisas feitas entre todos os protocolos existentes, os mais conhecidos/referenciados e dos mais utilizados são os protocolos AODV, OLSR e DSDV e algumas das suas vertentes.

Ao longo deste capítulo pretende-se expor as características e o funcionamento de cada um dos protocolos utilizados no desenrolar deste trabalho.

3.2 Protocolo AODV

O AODV(*Ad hoc On-demand Distance Vector*) é um protocolo de encaminhamento pertencente à família dos protocolos reativos, com o propósito de estabeleci-

mento de rotas quando necessário reencaminhar mensagens ou quando é requerida uma comunicação. Apenas são estabelecidas rotas quando é necessário enviar dados para um nó destino, que não conste na tabela de encaminhamento, realizando tanto encaminhamento *unicast*, quanto encaminhamento *multicast*[16].

O AODV permite dinâmica na rede e tem como característica fundamental a iniciação automática do encaminhamento *multi hop*, entre os vários participantes de uma rede Ad Hoc. É um protocolo que inicia a descoberta de melhores caminhos automaticamente, permitindo que cada nó obtenha de forma rápida rotas para novos destinos não requerendo que sejam guardadas as rotas durante todo o tempo de vida da rede. Apenas se inicia o processo de descobrimento de rota quando uma rota é realmente requerida. Sempre que a rota é descoberta, utiliza um procedimento de manutenção de rotas para que essas rotas continuem ativas. Portanto, apenas mantém guardado as rotas entre os nós que é necessário comunicar, não guardando uma tabela de encaminhamento com rotas para toda a topologia, cada nó da rede apenas guarda uma tabela de encaminhamento que contém apenas o próximo salto para o nó destino. Contudo, o AODV permite que os nós da rede consigam responder corretamente em caso de uma falha de ligação, e a mudanças de topologia em tempo útil.

Por outro lado, este protocolo utiliza um mecanismo para evitar o acontecimento de *loops*, o que neste tipo de redes poderá ocorrer muito facilmente, visto que todos os intervenientes da rede podem funcionar como encaminhadores e poderão receber o mesmo pacote vindo de vários destinos. Para isso o AODV utiliza nas suas mensagens um campo designado como número de sequência[1], impedindo assim a realização de contagens até ao infinito. Com isto obtém-se uma convergência rápida quando ocorrem mudanças da topologia de rede.

Sendo assim, quando um pacote chega a um nó da rede e é necessário encaminhá-lo, como este protocolo não troca constantemente mensagens de controlo de rotas fazendo com que se economize largura de banda e energia dos dispositivos, o nó analisa o pacote recebido verificando se é ele o destinatário ou então reencaminha

[1]Um número crescente mantido por cada nó originador da mensagem. Este campo é utilizado pelos outros nós para determinarem qual a informação mais recente.

o pacote na rede. Por outro lado, não tendo tabelas atualizadas com todas as rotas possíveis na rede, os nós não têm uma completa visão da topologia de rede.

No caso de uma falha na ligação ou rutura de ligação, os nós diretamente ligados são capazes de dar conta desse problema e de notificar os nós vizinhos, invalidando a rota utilizada por esse *link*, e estabelecer assim rapidamente um novo caminho entre os intervenientes afetados, sendo reposta a comunicação entre os nós. O AODV notifica nestes casos o conjunto de nós afetados na falha de ligação, para que todos estes invalidem a rota utilizada e reparem as rotas utilizadas rapidamente.

Como foi identificado anteriormente, o AODV adota o conceito de número de sequência, diferenciado na fonte e no destinatário. O número de sequência da fonte é utilizado para manter na sua tabela de encaminhamento as mais recentes informações do caminho inverso para a origem. O número de sequência do destino representa o quão recente uma rota para um destino é, comparando com os números de sequência de informação anteriormente recebida da mesma rota. Este número de sequência do destino é criado pelo destino, sendo enviado juntamente com a restante informação requerida pelos nós. Assim através destes números de sequência, dadas duas ou mais rotas para um mesmo destino, o encaminhador verifica qual a informação mais recente e escolhe a rota mais atualizada possível, sendo essa a melhor rota para atingir o destino.

Quando é necessário iniciar uma comunicação para um nó destino, o protocolo AODV faz propagar pela topologia de rede uma mensagem de RREQ(*Route Request*), com o objetivo de obter uma visão parcial da topologia e do melhor caminho para atingir o destino pretendido. Este protocolo utiliza assim o procedimento de descoberta de rotas através do nó origem, através do processo designado *path discovery*[2]. Quando um nó envia a mensagem RREQ este é conhecido como originador(requerente/fonte), sendo esta requisição de rota transmitido eficientemente pela rede entre os diversos nós, evitando a ocorrência de *loops* e retransmissões indesejáveis. Estes problemas são evitados através da utilização do respetivo número de sequência incorporado na mensagem, que é distinto a cada solicitação.

Ao longo do processo de descoberta de rotas e do estabelecimento de comu-

[2]Processo de descoberta de um caminho desde uma origem para um destino desconhecido.

nicação entre os nós, são trocadas mensagens de RREQ, RREP(*Route Reply*) e RERR(*Route Error*), sendo estas as mensagens típicas definidas pelo AODV. Este tipo de mensagens são recebidas via UDP [3]. Deste modo, no AODV, o processo de descoberta de uma rota de um nó até ao seu destino realiza-se através da inundação de pacotes RREQ por todas as ligações dos nós. Assim o nó origem que requer a comunicação, usa o seu endereço IP como endereço IP Originador, transmitindo as mensagens, utilizando o endereço IP de *broadcast* [4] limitado. Isto significa que as mensagens não são transmitidas às cegas pela rede. A distância que estas mensagens irão percorrer pela rede será limitado e indicado pelo campo TTL(*Time To Live*) no cabeçalho IP do pacote enviado.

As mensagens RREQ podem ser enviadas para todos os nós da rede ou para um grupo *multicast*. A rota pode então ser determinada quando um RREQ chega ao destino, ou a um nó intermediário que contém na sua tabela de encaminhamento uma rota para o destino, onde o número de sequência associado é pelo menos tão grande como o número contido no RREQ. Neste caso, é enviada de volta por *unicast* uma mensagem do tipo RREP para o originador do RREQ. Cada nó que reencaminha uma mensagem de RREQ recebida, armazena em *cache* uma rota de volta, guardando o endereço do vizinho a partir do qual o RREQ foi enviado. Assim, a mensagem de RREP é reencaminhada nó após nó até ao originador, mantendo uma rota ativa em todos os nós que encaminham as mensagens. Quando um nó recebe um pacote RREQ, este verifica os campos Endereço da Fonte (origem do pacote) e o Identificador de *Broadcast*, para verificar se o nó já tinha recebido anteriormente alguma requisição com o mesmo Identificador de *Broadcast* e o mesmo Endereço da Fonte, analisando se o pacote recebido é ou não duplicado. Sendo um pacote duplicado, o nó descarta o pacote RREQ recebido.

Quando um nó recebe um pacote RREQ, depois de verificados os campos necessários para saber que não é um pacote duplicado, não sendo o destino desejado e não possuindo uma rota válida para alcançar o destino pretendido, o nó incre-

[3]Protocolo da camada de transporte, descrito pela RFC768, que não oferece garantias de que um pacote chegue ao seu destino. Concede às aplicações acesso direto de entrega de *datagramas*, comparado com o serviço de entrega que o IP oferece.

[4]A difusão consiste na transmissão para endereços IP limitados (255.255.255.255).

menta uma unidade no campo Contador de Saltos e reencaminha o pacote pelos seus vizinhos.

O originador poderá receber várias mensagens de RREP, consoante o número de caminhos possíveis até ao destino pretendido, sendo este capaz de escolher o melhor caminho, a rota mais adequada com base nas métricas utilizadas. Usualmente utiliza-se o número de saltos ou a largura de banda disponível para a escolha da melhor rota. Uma rota apenas é válida quando os pacotes enviados por esse caminho são bem encaminhados, se os pacotes não forem encaminhados até ao destino dentro de um tempo definido, essa rota é descartada e torna-se uma rota inválida. Com este mecanismo é reduzida a informação de rotas incorretas ou rotas antigas que possam já não estar ativas nas tabelas de encaminhamento. A figura 3.1 descreve os processos de descoberta de rota e de resposta a um erro de ligação entre os nós.

Figura 3.1: Descoberta de uma rota e resposta a erro de ligação

Depois de uma rota estar estabelecida, os nós intervenientes que acompanham a ligação trocam informação entre eles, monitorizando o estado das ligações para os seus vizinhos, utilizando informação do *driver* ou através de mensagens RREQ especiais, designadas de mensagens "Hello". Estas mensagens servem para verificar se não existe quebras de ligações ou mudanças na topologia que interditem a comunicação. Caso contrário, quando é detetada uma quebra de ligação de uma rota ativa ou um mau funcionamento de um nó, o nó vizinho que deteta esta irregularidade envia uma mensagem de RERR para o remetente, de modo a notificar o ocorrido a todos os nós intervenientes na comunicação. Um RERR indica que aqueles destinos mencionados não estão alcançáveis devido a uma rotura de ligação ou erro de comunicação, fazendo com que os nós apaguem esta entrada nas tabelas de encaminhamento, invalidando a rota atual para esse destino. Depois do

originador receber a mensagem de erro na ligação, este inicia de novo o processo de descoberta de uma nova rota para atingir o destino.

O AODV utiliza os seguintes campos para cada entrada(rota) na tabela de encaminhamento[16]:

- Endereço IP destino
- Número de sequência do destino
- *Flag* de validação do número de sequência do destino
- Outros estados e *flags* de encaminhamento(por exemplo válido, não válido)
- Interface de rede
- Contador de saltos(indica o número de saltos para atingir o destino)
- Próximo salto
- Lista de precursores
- Tempo de vida(*lifetime*) da rota.

Todas as mensagens que são trocadas pelo uso do protocolo AODV são enviadas pela porta 654 utilizando o protocolo UDP. Relativamente aos números de sequência, que é um dos pontos fundamentais deste protocolo, estes são incrementados no nó destino em duas situações:

1. Imediatamente antes do nó gerar uma descoberta de rota, sendo necessário incrementar o próprio número de sequência. Isto realiza-se para evitar conflitos com rotas previamente estabelecidas, em direção ao originador da mensagem RREQ.
2. Imediatamente antes do nó destino originar a RREP em resposta ao RREQ, obrigando a atualizações do seu próprio número de sequência para o valor máximo, superior ao número de sequência atual e ao número de sequência do pacote RREQ.

Um nó pode alterar o número de sequência de uma entrada na tabela de encaminhamento de um nó destino apenas se for ele próprio o nó destino, oferecendo uma nova rota para ele próprio. Se o nó receber mensagens AODV com novas

informações sobre o número de sequência para um nó destino, ou se o caminho em direção a um nó destino expirar, o nó atualiza esse valor na sua tabela de encaminhamento.

Processamento das mensagens RREQ

As mensagens do tipo RREQ são utilizadas essencialmente para fazer o descobrimento de uma nova rota, sempre que um nó fonte deseja comunicar com outro nó e para o qual não possui nenhuma entrada na tabela de encaminhamento. No caso de um destino ser alcançável, os nós da rede através dos pacotes RREQ, adquirem informações suficientes para retornar uma resposta até a fonte. Desta forma cada nó, que encaminha para a rede um pacote RREQ, deve armazenar automaticamente o endereço do vizinho de quem recebeu a primeira cópia do pacote RREQ, o número de sequência da fonte que originou o RREQ e o tempo de vida [42] [49]. Os nós intervenientes no caminho inverso aprendem a rota para o destino, como consequência da descoberta da rota de origem. Entretanto, os nós que não fazem parte do caminho inverso apagam a entrada na tabelas de encaminhamento de rotas inversas, quando o tempo associado a esse processo expirar. A mensagem RREQ tem a seguinte estrutura:

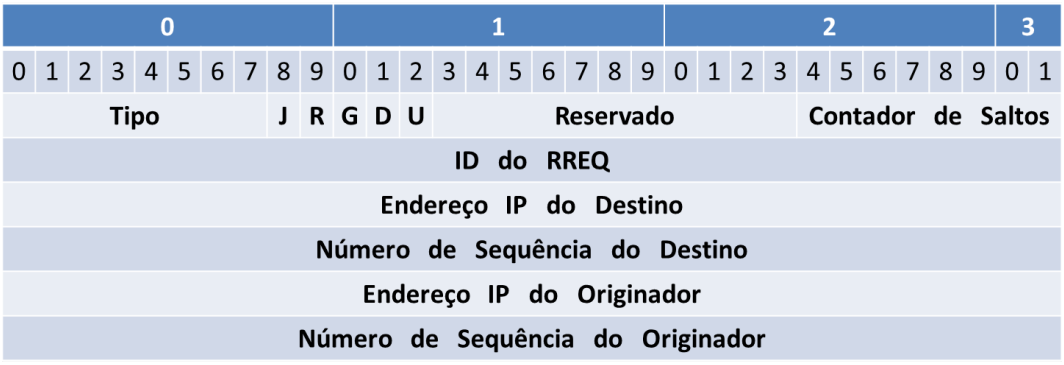


Figura 3.2: Estrutura da mensagem RREQ

Esta mensagem (Figura 3.2) possui os seguintes campos: Endereço IP da Fonte, Número de Sequência da Fonte, Identificador de Broadcast (ID do RREQ), Endereço IP do Destino, Número de Sequência do Destino e um campo de Contador de Saltos(inicializado a zero) entre outras *flags* que compõe o pacote para alguma

possíveis instruções. Deste modo temos a seguinte descrição:

- Tipo - Identifica o tipo de mensagem;
- Contador de Saltos - Refere-se ao contador de *hops* iniciado a 0 quando o nó origem envia a mensagem;
- ID do RREQ - É um identificador único incrementado localmente na fonte para cada pedido;
- Endereço IP do Destino - Refere-se ao endereço IP do nó destino para o qual são enviado pacotes/mensagens;
- Número de Sequência do Destino - Indica o tempo lógico da última informação relativa ao nó destino conhecida pelo nó fonte. No caso inicial como não tem essa informação, envia 0.
- Endereço IP do Originador - Refere-se ao endereço IP do nó fonte ou originador de pacotes;
- Número de Sequência do Originador - tempo lógico referente ao nó fonte, relativo ao envio do pedido.

Processamento das mensagens *"Hello"*

As mensagens *Hello* (Figura 3.3), são mensagens geradas pelos nós para oferecerem informações sobre o seu estado de ligação. Os nós da rede podem fazer esta verificação através da escuta de pacotes *broadcast*, que são endereçados a outros vizinhos, ou enviando periodicamente mensagens *Hello* com a sua identidade e o seu número de sequência. Neste caso, um nó só enviará mensagens *Hello* se fizer parte de uma rota ativa e para verificar se as suas ligações estão ativas. A cada HELLO_INTERVAL, o nó faz uma atualização do estado das suas ligações através do *broadcast* de mensagens, como por exemplo com o RREQ ou mesmo com mensagens da camada 2.

Caso o nó não tenha a rota ativa, este pode enviar por *broadcast* uma mensagem RREP com o campo TTL=1(configurado deste modo para que apenas os seus vizinhos diretos recebam este pacote e atualizem a sua informação de ligação local), designando assim a mensagem *Hello*, e com os campos definidos da seguinte forma:

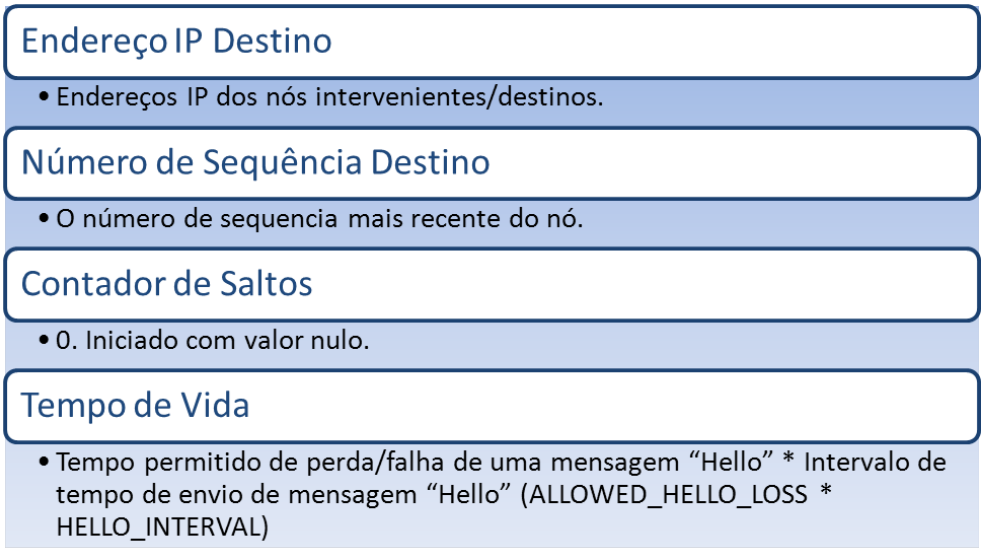


Figura 3.3: Campos de uma mensagem *"Hello"*

Ao receber esta mensagem *Hello*, os nós criam ou atualizam uma entrada na sua tabela de encaminhamento, relativa ao vizinho do qual recebeu a mensagem. Caso não seja recebida a mensagem *Hello*, o nó considera que houve quebra de ligação e que, portanto, poderá ter ocorrido alguma mudança na topologia de rede. Com isto, um nó pode determinar a conectividade através da receção destes tipos de pacotes vindo dos seus vizinhos.

Processamento das mensagens RREP

As mensagens RREP são geradas pelos nós no caso de estes serem o nó destino ou se o nó que recebe o RREQ conter na sua tabela de encaminhamento um caminho válido para atingir o destino desejado. Neste caso, se o nó tiver uma rota ativa e o número de sequência de destino for melhor ou igual à do RREQ, o nó gere uma mensagem de RREP de volta atualizando os campos da mensagem que envia de volta.

Quando um nó gera uma mensagem de RREP, ele copia o endereço IP do destino e o número de sequência do originador da mensagem RREQ, para fazer corresponder com os campos da mensagem RREP. Depois de criada a mensagem, o RREP é enviado para o originador do RREQ. À medida que o RREP é encaminhado de volta para o nó que originou o RREQ, o campo de Contador de Saltos

é incrementado por cada nó que atravesse. Quando o RREP chega ao originador, o campo Contador de Saltos corresponde à distância, em saltos(*hops*), do destino até ao originador.

Finalizando, o pacote RREP (Figura 3.4), é formado pelos seguintes campos: Endereço IP da Fonte, Endereço IP de Destino, Número de Sequência do Destino, Contador de Saltos e um campo com o Tempo de Vida do pacote.

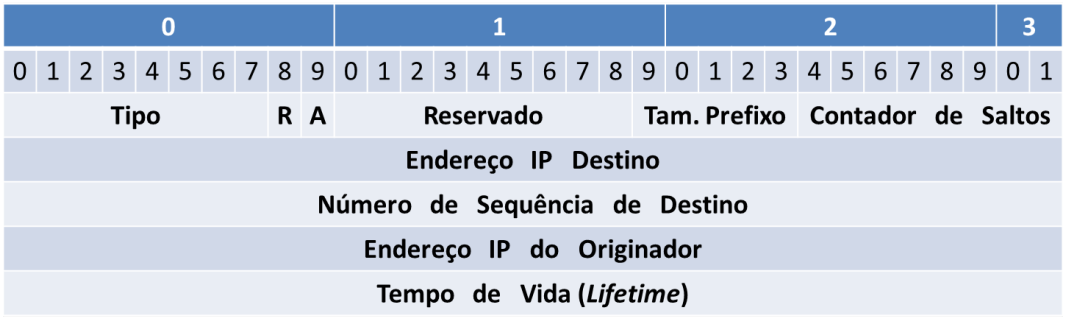


Figura 3.4: Estrutura da mensagem RREP

Deste modo temos a seguinte descrição:

- Tipo - Identifica o tipo de mensagem;
- Tam. Prefixo - Representa um prefixo reservado para subredes, iniciado com valor zero;
- Contador de Saltos - Refere-se ao contador de *hops* iniciado a 0 quando o nó origem envia a mensagem;
- Endereço IP do Destino - Refere-se ao endereço IP do nó destino para o qual são enviado pacotes/mensagens;
- Número de Sequência do Destino - Indica o tempo lógico da última informação relativa ao nó destino conhecida pelo nó fonte. No caso inicial como não tem essa informação, envia 0.
- Endereço IP do Originador - Refere-se ao endereço IP do nó fonte ou originador de pacotes;
- Tempo de vida, *Lifetime* - Representa o tempo de vida do pacote em circulação na rede.

Por outro lado, quando um nó recebe um RREP ele procura, utilizando o prefixo mais longo correspondente, uma rota para o nó anterior. Em seguida o nó, incrementa a contagem de saltos do valor do RREP para representar um salto mais através de um nó intermediário. Deste modo um novo caminho é construido se não existe no momento para esse destino. Caso contrário o nó compara o valor do número de sequência do destino armazenado para o endereço IP de destino na mensagem RREP. Após feita esta comparação, esta entrada apenas é atualizada somente nas seguintes circunstâncias:

1. se o número de sequência estiver marcado como inválido na entrada da tabela de encaminhamento;
2. se o número de sequência de destino no RREP é maior do que a cópia do número de sequência de destino e se este for válido, ou
3. se os números de sequência são iguais, mas o percurso está marcado como inativo, ou ainda
4. se os números de sequências são os mesmos e o novo valor de saltos é menor do que a contagem de número de saltos na entrada da tabela, sendo desta vez um caminho mais curto.

Processamento das mensagens RERR

As mensagens RERR (Figura 3.5)que indicam erro de rota, rota expirada e rota apagada ou inválida, podem ser também enviadas por *broadcast*, no caso de ter muitos nós antecedentes, e por *unicast* no caso de ter apenas um nó antes dele próprio.

Estas mensagens são trocadas quando é detetado um erro de ligação de uma rota, uma rota que expirou ou uma rota que foi apagada da tabela de encaminhamento. Geralmente, um erro de rota e uma quebra de ligação são processados da seguinte forma: num primeiro momento são invalidadas as rotas existentes, de seguida elabora-se uma listagem de destinos afetados, verifica-se se os vizinhos podem ser afetados com erros de possíveis rotas utilizadas por esses vizinhos e envia-se uma mensagem RERR adequada a esses vizinhos. Contudo, os nós ape-

nas iniciam o processamento de RERR caso seja detetada uma quebra de ligação para um nó seguinte, de uma rota ativa na tabela de encaminhamento, ou quando um nó obtém um pacote de dados destinado a esse nó, para o qual já não tem uma rota ativa. Por último, será também processado um RERR caso um nó receba um RERR de um nó vizinho, para uma ou mais rotas ativas, tendo sido neste caso criados RERR errados ou já antigos.

O RERR deve conter os destinos que fazem parte da lista de destinos não acessíveis pelo nó. Para tal o pacote RERR é formado pelos seguintes campos:

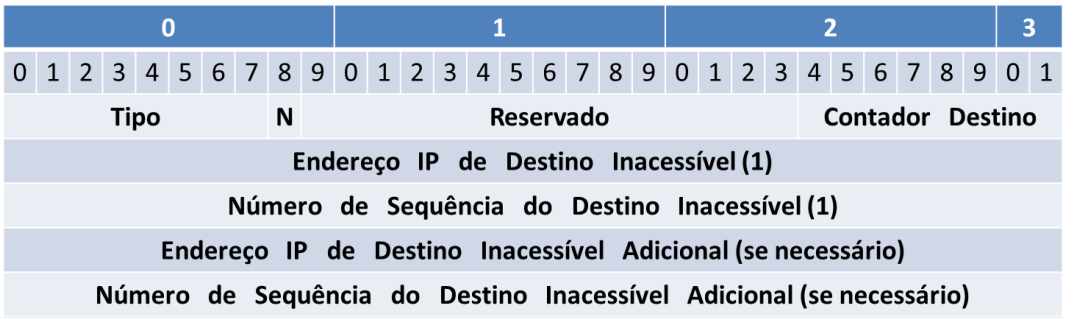


Figura 3.5: Estrutura da mensagem RERR

- Tipo - Identifica o tipo de mensagem;
- Contador Destino - Refere-se ao contador de *hops* iniciado a 0 quando o nó origem envia a mensagem;
- Endereço IP do Destino Inacessível - Refere-se ao endereço IP do nó destino para o qual não é possível enviar pacotes/mensagens;
- Número de Sequência do Destino Inacessível - Indica o tempo lógico da última informação relativa ao nó destino inacessível de momento conhecida pelo nó fonte. No caso inicial como não tem essa informação, envia 0;
- Endereço IP do Destino Inacessível Adicional- Refere-se ao endereço IP do nó destino para o qual não é possível enviar pacotes/mensagens, endereço extra se necessário devido ao encaminhamento;
- Número de Sequência do Destino Inacessível Adicional - Indica o tempo lógico da última informação relativa ao nó destino inacessível de momento

conhecida pelo nó fonte. No caso inicial como não tem essa informação, envia 0. Pode conter informação extra se necessário devido ao encaminhamento ou a um novo valor;

O AODV é um protocolo de encaminhamento muito utilizado nas redes Ad Hoc, isto por oferecer rápida adaptação às condições da rede dinâmica, necessita de pouca utilização da rede, baixo processamento, sobrecarga de memória (tabela de encaminhamento) e determina rotas *unicast* para destinos dentro da rede Ad Hoc.

Verificou-se a existência de algumas implementações do protocolo AODV [45], tanto em ambientes de simulação como em ambiente real, suportando vários sistemas operacionais, diferentes arquiteturas e versões IP.

3.3 Protocolo OLSR

O OLSR é considerado um dos principais protocolos de encaminhamento nas redes MANET. Este protocolo faz parte da família dos protocolos pró-ativos, desenvolvido especificamente para o tipo de redes Ad Hoc, através do INRIA(*Institut National de Recherche en Informatique et en Automatique*) e proposto pela primeira vez na conferência *IEEE International Multitopic Conference* em 2001[31]. Mais tarde e com o decorrer dos estudos em volta deste protocolo, foi desenvolvida dentro do grupo MANET IETF, sua estrutura, sendo proposto e definido através da RFC3626 [21]. É um protocolo de encaminhamento IP otimizado para dispositivos móveis, baseado no modo de encaminhamento ponto-a-ponto, e que faz uso do tradicional algoritmo de estado de ligação(*link state*).

Algoritmo Estado de Ligação

O algoritmo Estado de Ligação (*Link State*) baseia-se na construção de um grafo, onde a cada ligação entre os nós é atribuído um custo que é proveniente de um cálculo. Pode utilizar diversas métricas, entre elas a largura de banda disponível e a latência ou tempo de resposta de um nó. A melhor rota é assim

definida através do percorrer de um grafo, feito por um algoritmo, onde a rota com menor custo será a considerada a rota mais curta e assinalada como o melhor caminho de um nó até ao outro nó.

No OLSR cada nó da rede contém a informação da topologia de rede, devido à constante troca de mensagens de estado de ligação entre os nós. A principal melhoria imposta como o uso deste protocolo é minimizar o tamanho de cada mensagem de controlo e o facto de serem menos nós a transmitir cada atualização de rotas na rede. Esta diminuição de retransmissões deve-se à estratégia da escolha de nós que retransmitam as mensagens pela rede, designados por *MultiPoint Relay* MPR[9]. Em cada atualização da rede em que ocorra mudança de topologia, cada nó da rede seleciona um conjunto de nós vizinhos para retransmitir os seus pacotes, sendo esses nós retransmissores os MPR's desse nó. Todos os outros nós que não são designados como MPR, recebem os pacotes de dados, leem e processam os pacotes mas não os transmitem, sendo a função de retransmissão apenas disponível e aplicável aos MPR's. Estes nós MPR's têm um papel importante, pois todas as informações que chegam dos nós vizinhos são utilizadas para cálculo do caminho mais curto até ao nó destino. Este cálculo utiliza as informações dos estados das ligações entre os nós, através de informações como balanceamento de carga e redundância, que são também utilizados para o cálculo do melhor caminho. Este melhor caminho, ou caminho mais curto pode ser o caminho com mais saltos, mas com melhores *links*, diminuindo a latência da comunicação [12].

Este processo tem como finalidade fazer com que sejam trocadas menos mensagens na rede, com o fim de eliminar mensagens redundantes. No caso mais simples, de não utilização dos MPR's, quando um nó recebe um pacote de controlo, ele normalmente vai retransmitir o pacote recebido por todos os seus vizinhos, menos pelo qual recebeu o pacote, num processo este designado de *flooding* (inundação). Com isto os nós vão receber periodicamente, e de diferentes vizinhos, pacotes repetidos, fenómeno este designado por *overhead*[5]. Com a utilização dos nós MPR, só

[5] *Overhead* é um termo utilizado para descrever os custos associados à troca de meta-dados e informações de encaminhamento da rede entre dispositivos. Considerado como qualquer processamento ou armazenamento em excesso, seja de tempo de computação, de memória, de largura

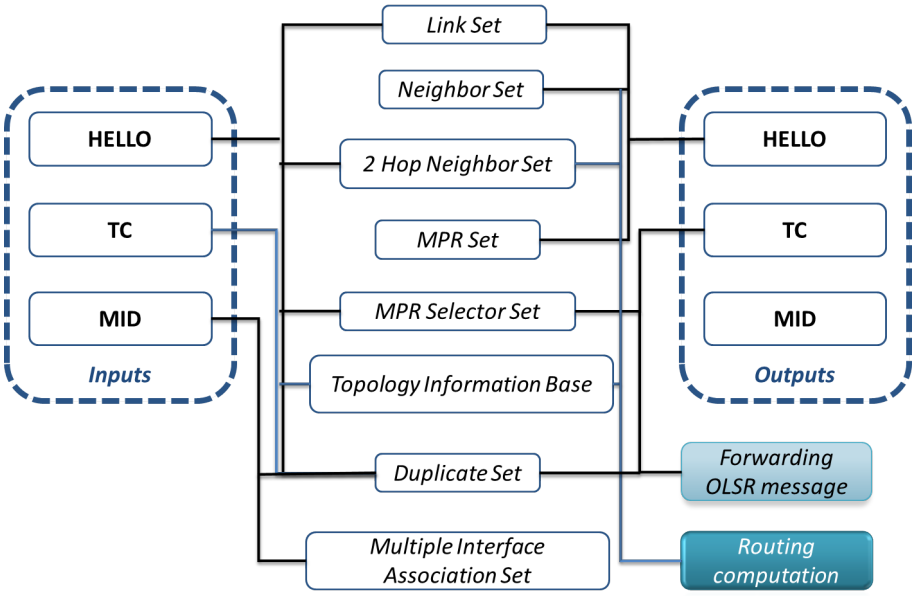


Figura 3.6: Arquitectura do protocolo OLSR (adaptado de [25])

estes é que irão retransmitir os pacotes de controlo, limitando assim o número de retransmissões pela rede.

A RFC3626 descreve todos os pré-requisitos mínimos de funcionamento do protocolo OLSR, indicando que este protocolo se baseia numa otimização do algoritmo "Estado de Ligação".

Para a escolha dos MPR's, por parte de um nó, é enviada periodicamente uma mensagem do tipo *"Hello"*, com uma lista dos seus vizinhos a um salto de distância(diretamente ligados), como mostra a Figura 3.7. A partir da lista que cada nó recebe, cada nó seleciona um subconjunto mínimo de vizinhos a um salto de distância, de forma a que cubra todos os caminhos para os seus vizinhos a dois saltos de distância. Esse subconjunto mínimo é o conjunto de nós MPR. Um nó não MPR, identifica os seus MPR que reencaminham os seus pacotes. A principal função dos MPR é diminuir a quantidade de mensagens *broadcast* repetidas que são transmitidas na rede.

Como descreve a primeira imagem da Figura 3.8, sem a definição de nós MPR,

de banda ou qualquer outro recurso que seja requerido para ser utilizado ou gasto, para executar uma determinada tarefa.

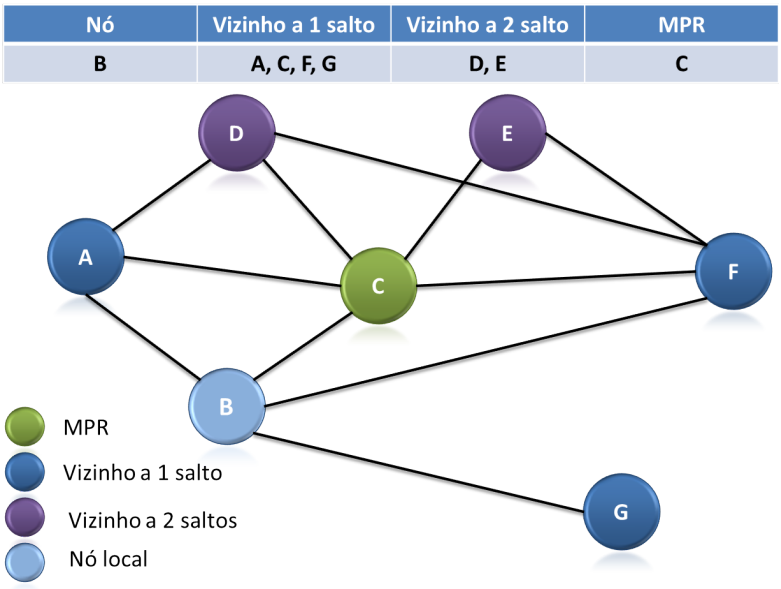


Figura 3.7: Processo de escolha do nó MPR

um nó envia um pacote aos seus vizinhos a um salto e todos estes que recebem o pacote fazem a sua retransmissão pelos restantes nós da rede. Como todos os nós retransmitem o pacote aos seus vizinhos, este vai percorrer toda a rede, salto após salto, fenómeno designado por *flooding*. Deste modo todos os nós receberão pacotes, sendo a maior parte replicados, provocando uma sobrecarga na rede (*overhead*). Com o uso de nós MPR (segunda imagem da Figura 3.8), apenas os MPR's retransmitem os pacotes na rede. Neste caso o primeiro nó envia pacotes aos seus vizinhos, sendo que os nós que não são MPR não retransmitem os pacotes recebidos, apenas os nós MPR é que tratam de fazer essa retransmissão. Portanto, os próximos nós recetores só receberão a informação uma vez, o que reduz a chegada de pacotes redundantes.

De entre toda esta troca de informação entre os nós da rede, dois tipos de pacotes padrão são utilizados para disseminar estas informações sobre toda a topologia da rede. Assim, existem os pacotes do tipo *hello* e TC (*Topology Control*), e os pacotes MID (*Multiple Interfaces Declaration*), tal como ilustrado na Figura 3.6 da arquitetura do protocolo OLSR. Este último tipo de mensagens serve para correlacionar os endereços das múltiplas interfaces de um nó com o seu endereço principal [21].

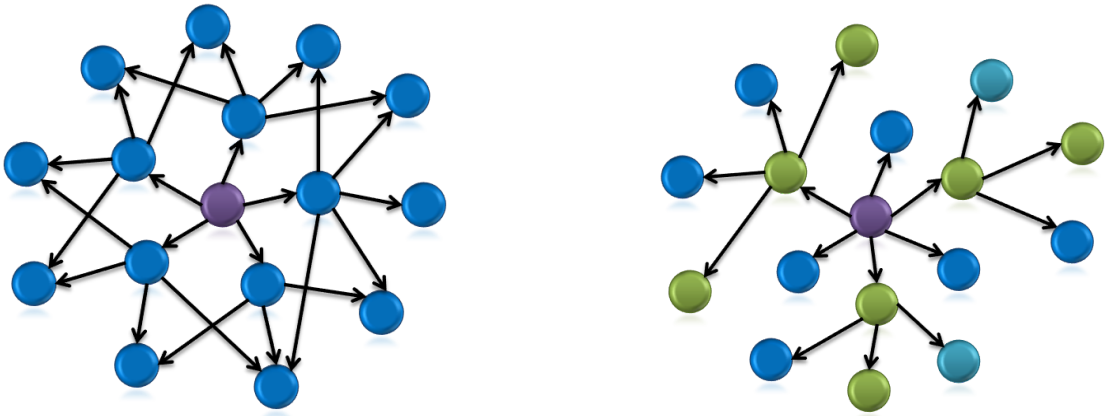


Figura 3.8: Comparação do protocolo OLSR com e sem nós MPR

Formato de pacote OLSR

O protocolo OLSR utiliza um formato de pacote único para transportarem todos os tipos de mensagens. Assim as mensagens são agrupadas dentro de pacotes OLSR, descritos na Figura 3.9, que por sua vez são agrupados dentro de datagramas UDP. Por outro lado, cada mensagem utiliza um formato próprio, como será representado de seguida, consoante o tipo "*hello*", TC ou MID. Com a utilização de um formato único facilita a criação de extensões para o protocolo sem causar problemas de compatibilidade, possibilitando o envio de diferentes tipos de mensagens em uma única transmissão.

Este pacote é então constituído pela informação do tamanho do pacote (em bytes) e o número de sequência do mesmo, que é incrementado sempre que um novo pacote é transmitido. Como já referido, este número de sequência serve então para verificar o quão recente o pacote é, evitando assim a retransmissão do mesmo pacote mais do que uma vez. Continuando, o campo Tipo, indica o tipo de mensagem que é anexada ao pacote (*hello* ou TC). O campo Vtime indica o tempo de vida que esta mensagem é considerada válida. Seguidamente, indicam o tamanho da mensagem, o endereço de origem, endereço este que representa o endereço do nó que criou esta mensagem e o campo TTL (*Time To Live*) que indica o número de saltos que esta mensagem pode percorrer até ser considerada inválida.

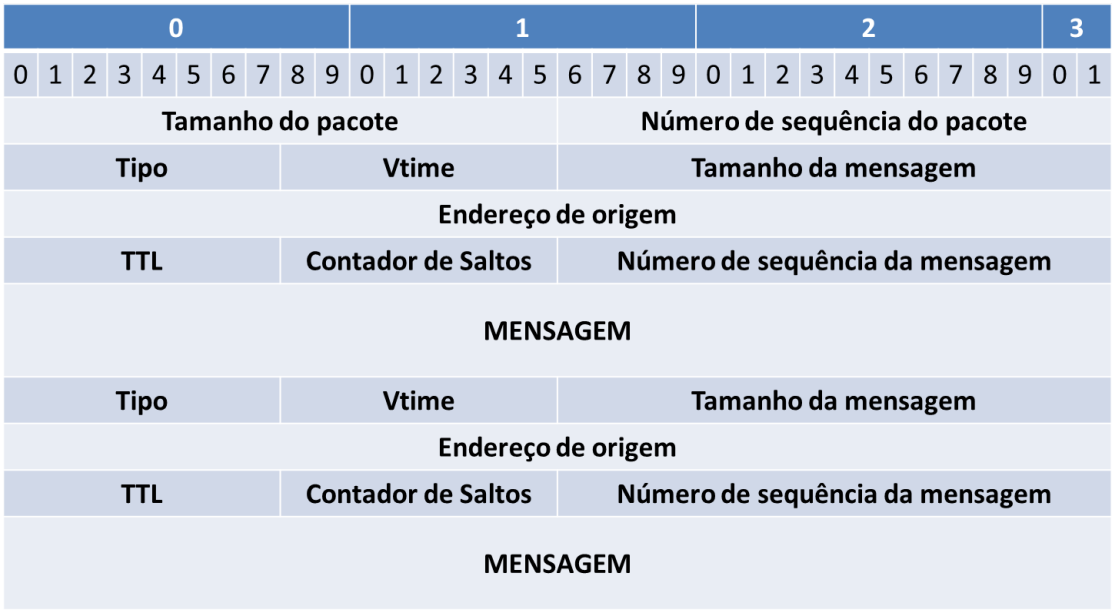


Figura 3.9: Formato do pacote de transporte OLSR

Mensagem *hello*

As mensagens *hello* são utilizadas para o descobrimento e atualização de novos vizinhos diretos e através das respostas destes, da lista de vizinhos a dois saltos de distância. Com esta estratégia, de troca periódica de informação com o estado das suas ligações, cada nó mantém as suas tabelas de vizinhos atualizadas. . O formato proposto pela RFC3626 para as mensagens *hello* é o seguinte:

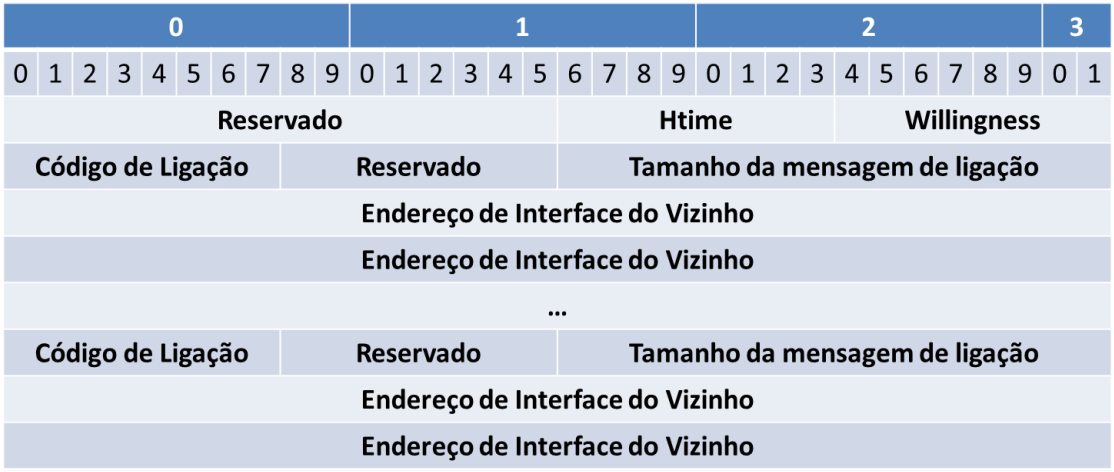


Figura 3.10: Formato da mensagem *hello*.

Como estas mensagens são enviadas periodicamente aos vizinhos a um salto

(TTL igual a 1), não é necessário nenhum mecanismo de retransmissão da mensagem. As mensagens de *hello* não são retransmitidas.

Esta mensagem é então composta pelo campo *Htime*, que especifica o intervalo de emissão de mensagens deste tipo, indicando a validade das informações da mensagem em segundos. O campo *Willingness* que representa a disponibilidade do nó para a retransmissão dos dados. Se este campo estiver marcado como WILL_NEVER, o nó não será capaz de retransmitir pacotes. Caso este campo esteja marcado como WILL_ALWAYS, o nó poderá sempre retransmitir e assim terá grande potencial em ser designado como nó MPR. O campo *Código de ligação* indica qual o tipo de ligação o nó recetor da mensagem tem com o seu vizinho. E por fim o tamanho da mensagem de ligação, indica qual o comprimento total utilizado pela lista de vizinhos com um mesmo código de ligação. De seguida são representados os campos com o endereço do nó vizinho ao nó emissor da mensagem *hello*.

Mensagem TC

A mensagem TC (Figura 3.11) trata das informações armazenadas pelas mensagens *hello* que servem como base a disseminação de informações topológicas por toda a rede e que serão utilizadas na construção das rotas.

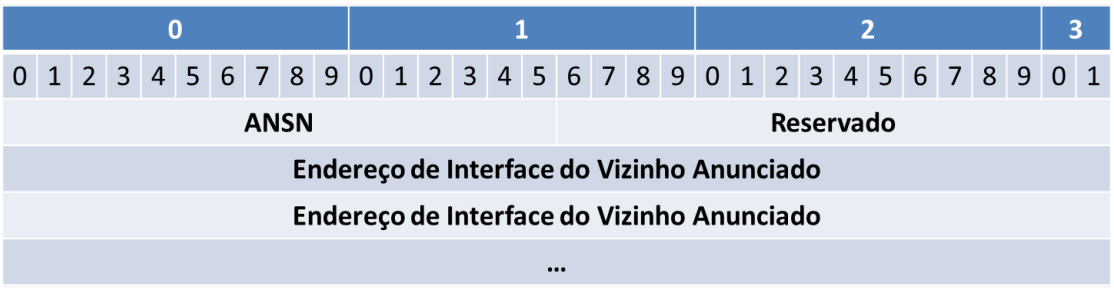


Figura 3.11: Formato da mensagem TC.

Como apenas os nós MPR's é que retransmitem os pacotes pela rede, estes disseminam a lista dos nós a que estão ligados e para os quais tem uma rota ativa, através das mensagens TC. Assim cada nó da rede reconhece qual o seu MPR e através do qual um determinado nó pode ser alcançado. O campo ANSN da

figura 3.11, indica o número de sequência associado aos anúncios dos nós vizinhos, que deve ser incrementados a um cada vez que o nó deteta uma alteração no conjunto dos seus vizinhos. Através dele o nó recetor saberá se a mensagem é recente e se deve ser considerada ou rejeitada. Com toda esta informação, cada nó é capaz de calcular as rotas localmente através do algoritmo *Dijkstra*. Por outro lado o campo Reservado pode ser utilizado por extensões do protocolo. Por fim, o campo Endereço de Interface do Vizinho Anunciado descreve o endereço principal do vizinho do nó remetente da mensagem.

Mensagem MID

Finalizando, as mensagens MID (*Multiple Interfaces Declaration*), figura 3.12, são enviadas apenas por nós que têm dois ou mais interfaces de rede.

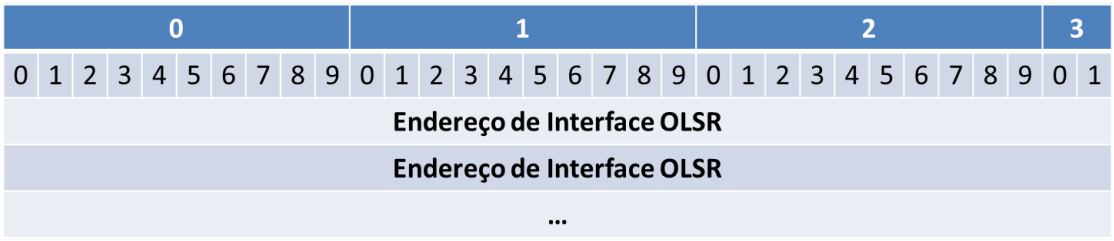


Figura 3.12: Formato da mensagem MID

Toda esta informação é utilizada para resolver o endereço principal que identifica o nó remetente de alguma mensagem, já que essa mensagem pode ser enviada por várias interfaces diferentes, com endereços diferentes do endereço principal. Assim, o nó que possui mais de uma interface deve anunciar a sua situação através deste tipo de mensagens, sendo enviadas periodicamente na rede. Esta informação, quando recebida pelos nós, são armazenadas e utilizadas para cálculo de novas rotas. Esta mensagem é então enviada com o tipo de mensagem definido como MID_MESSAGE. Contudo o campo TTL deverá ser composto pelo valor máximo, de forma a garantir que a mensagem seja transmitida por todos os elementos da rede. O campo Vtime deve ser atribuído de acordo com o definido pelo protocolo[21].

O protocolo OLSR tem um comportamento mais organizado e eficiente, no

que toca à troca de mensagens entre os nós da rede, procurando sempre o caminho mais curto entre os nós. Por outro lado, este protocolo apresenta aspetos negativos como o *flooding* na rede, visto que periodicamente envia mensagens para verificar atualizações das tabelas de estado das ligações. Por sua vez, não se preocupa com a fiabilidade das mensagens com a topologia da rede, para evitar problemas de sincronização.

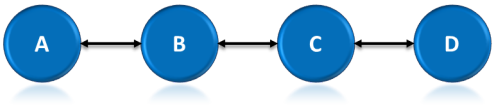
O protocolo tenta avaliar continuamente as rotas de modo que quando um pacote necessitar de encaminhamento a rota já seja conhecida e possa ser utilizada imediatamente. Neste caso, os nós mantêm uma ou mais tabelas com informações referentes à topologia de rede e respondem a mudanças na sua topologia propagando atualizações de modo a manter a consistência da rede. Estas atualizações são iniciadas por mecanismos de temporização, o que faz com que haja sempre um número constante de transmissões em andamento, mesmo quando a rede esteja em equilíbrio.

3.4 Protocolo DSDV

Por último, o protocolo DSDV pertence à família dos protocolos pro-ativos tal como o OLSR, que tem como base o uso do mecanismo de encaminhamento *Bellman-Ford* e adaptado do convencional algoritmo RIP (*Routing Information Protocol*), com eliminação de ciclos(*loops*) no encaminhamento em redes Ad Hoc. Este protocolo foi inicialmente desenvolvido por *Charles E. Perkins e Pravin Bhagwat* em 1994 [43].

Este protocolo adiciona um novo atributo, número de sequência, para cada entrada na tabela de encaminhamento relativamente ao convencional RIP. Com o uso deste mecanismo apenas é fornecido um único caminho para cada destino, obtido pelo algoritmo Vetor Distância. Os números de sequência permitem aos encaminhadores saberem e distinguirem quando uma entrada na tabela de encaminhamento é alterada, se a informação recebida é passada ou recente, bem como o autor desta ação, resultando daí o caminho mais curto para o destino [26]. Como referido, o DSDV vem adicionar duas novas coisas ao algoritmo de encaminha-

mento vetor distância: um número de sequência para evitar ciclos(como descrito no protocolo AODV) no encaminhamento de mensagens e a verificação de anúncios mais atuais, fazendo com que não seja guardada informação sobre anúncios de mudanças de curta duração.



Destino	Próximo	Métrica	Número de Sequência	Tempo de Registo	Apontador para Tabela
A	A	0	A-550	001000	Ptr_A
B	B	1	B-102	001200	Ptr_B
C	B	2	C-588	001200	Ptr_C
D	B	3	D-312	001200	Ptr_D

Figura 3.13: Exemplo de topologia e respetiva tabela de encaminhamento DSDV

Este protocolo caracteriza-se pelo facto de cada nó conter uma tabela de encaminhamento com uma lista de todos os caminhos possíveis na rede, conhecendo dessa forma toda a topologia de rede e o número de saltos necessários a atingir cada um desses nós destinos, utilizando a contagem de saltos como métrica para a seleção da melhor rota para um destino. Assim, cada nova rota que é inserida na tabela é identificada por um número de sequência, determinado pelo nó destino, fazendo com que sejam distinguidas as rotas mais antigas das mais recentes, com o objetivo de não formarem ciclos, nem de se obterem entradas nas tabelas de encaminhamento que não estão ativas de momento (Figura 3.13).

Periodicamente ou imediatamente quando é detetada uma alteração na topologia de rede, cada nó anuncia as suas informações através da comunicação por *broadcasting* ou *multicasting* das atualizações das suas tabelas de encaminhamento. Este pacote com as atualizações inicia-se com métrica igual a 1 para se dirigir apenas aos nós próximos, nós vizinhos a um salto.

As tabelas de encaminhamento neste protocolo são constituídas pelos seguintes campos:

- Destino

Este campo descreve os nós da rede que são atingíveis pelo nó, indicando os nós para onde é possível enviar dados;

- Próximo

  Neste campo é definido qual o próximo nó para o qual o pacote deve ser enviado, para que chegue ao seu destino, ou para que seja encaminhado com fim a atingir o destino pretendido;

- Métrica

  Consiste na medida de distância entre o nó fonte e o nó destino, descrevendo o número de saltos a que se encontra o nó destino;

- Número de sequência

  Consiste no número criado para que o nó consiga verificar a veracidade da informação recebida, utilizado para a sincronização das informações recebidas de forma a garantir que a informação que tem na sua tabela de encaminhamento é a mais atual possível, garantindo também a não ocorrência de ciclos indesejáveis na rede.

- Tempo de registo

  Valor que define o tempo que este registo existe, calculando-se assim o tempo máximo que esta informação se pode considerar. Através deste campo o nó decide se informação topológica recebida se deve apagar ou não.

- Apontador para a Tabela

  Ponteiro para uma tabela com informações da estabilidade da rota, utilizado para verificações de rotas recentes ou ativas[43];

Desta forma, quando um nó recebe informações dos seus vizinhos, este compara o número de sequência da informação recebida, com o número de sequência que têm na sua tabela de encaminhamento. Se o valor recebido for mais recente que o guardado, esta informação é atualizada sem que sejam comparados outros campos recebidos, caso o valor do número de sequência seja igual ao contido na tabela, é comparado então o campo métrica e assim será atualizada a tabela, caso a última informação recebida contenha melhor métrica. Caso contrário o nó incrementa o campo métrica a um e retransmite o pacote de atualizações nó após nó.

Com todas estas trocas de mensagens na rede e a periódica troca de mensagens de controlo é necessário evitar o congestionamento do meio. Para converter este processo, o protocolo DSDV tem como solução para evitar o congestionamento duas formas de atualizações:

- *Incremental Update*: Este é um tipo de atualizações em que o nó envia num só pacote todas as alterações ocorridas desde a última mensagem. O nó ao receber esta mensagem, adiciona todas as novas informações na sua tabela de encaminhamento e reencaminha esta informação para os nós seguintes. Quando um nó necessita enviar muita informação de mudanças ocorridas, e isto não é possível num só pacote, o nó utiliza o processo de *Full Update*.
- *Full Update*: Neste caso, os nós enviam toda a informação contida na sua tabela. É uma situação em que o nó envia informações sobre a totalidade da sua tabela de encaminhamento.

Este protocolo apresenta vantagens no que se refere à sua simplicidade, quase como o algoritmo vetor distância, combate à existência de *loops* devido aos números de sequência nas mensagens trocadas e devido a não causar latência no descobrimento de rotas. Algumas das suas desvantagens são o *overhead* criado na rede, o consumo de energia pelos dispositivos e o consumo de uma pequena quantidade de largura de banda da rede mesmo quando não é necessária transmissão de pacotes entre dispositivos. Com tudo isto este protocolo não se adequa muito a redes altamente dinâmicas.

3.5 Sumário

Com este capítulo, pretendeu-se fazer uma introdução e exposição do funcionamento dos protocolos de encaminhamento nas redes Ad Hoc, explorando o seu funcionamento e as principais características do encaminhamento de cada protocolo, incidindo nos protocolos utilizados (AODV, OLSR e DSDV) através da elaboração de uma descrição do funcionamento de cada um deles.

4

Conceção e desenvolvimento do cenário de testes

Este capítulo quatro é constituído por 5 secções, abordando todos os aspetos relacionados a conceção e desenvolvimento do cenário de testes. Na primeira secção começa-se por justificar a escolha da plataforma de simulação, que é depois descrita com algum detalhe na secção seguinte. A terceira secção apresenta os parâmetros de avaliação de desempenho considerados. A secção seguinte descreve todo o processo de desenvolvimento do *testebed* experimental, desde a criação do modelo de rede sem fios, os modelos de mobilidade dos vários dispositivos no espaço de rede, as aplicações geradoras de tráfego e todas as tarefas desenvolvidas para a obtenção dos valores das métricas e apresentação dos resultados. O capítulo termina com um pequeno sumário do processo e dos passos desenvolvidos.

4.1 Plataformas de Simulação

A simulação de redes é uma componente essencial para análise e validação na conceção de protocolos de rede. Embora a simulação não seja a única ferramenta utilizada para análise e conceção de redes, esta ferramenta é extremamente útil porque permite frequentemente analisar os dados que circulam na rede e o modo como estes circulam ou são postos a circular. Desta forma, vários protótipos de redes e protocolos de rede, podem ser explorados com maior ordem de grandeza, com menor custo e tempo do que seria requerido em experiências ou aplicações reais.

Existe uma série de simuladores de redes na comunidade de pesquisa e desenvolvimento, sendo que através de algumas pesquisas sobre simuladores e sobre importantes projetos em torno de redes de comunicação, verificamos que os mais populares são: NS-2 [39] [3], OMNet++ [5] [50], GloMoSim [1], JiST/SWANS [2], NS-3 [4] e OPNET[6], sendo este último um simulador de redes comercial.

Depois de várias pesquisas sobre estes simuladores, principalmente através da biblioteca digital do IEEE (Institute of Electrical and Electronics Engineers), em todos os artigos científicos identificados, entendemos que o simulador NS é dos mais influentes nesta área, com enorme vantagens de publicações relacionadas com o mesmo. Verificamos assim que, através da pesquisa na respetiva biblioteca digital, em que o título contenha as palavras "Network Simulator 2" obtemos 8.755 resultados retornados[1] e utilizando o título que contenha as palavras "Network Simulator 3" obtemos um resultado de 6.556 resultados retornados[2], o que demonstra que em este é dos simuladores mais utilizados nestes âmbitos de estudos. Através dos artigos [28] e [27] podemos verificar que o simulador NS-2 é referido em mais de 50% dos artigos publicados neste área da simulação de redes.

De acordo com os requisitos das simulações desenvolvidas e de acordo com a análise de algumas comparações destes simuladores foi então decidida a escolha do simulador NS-3. O simulador NS-3 apresenta sempre melhores cotações relativamente aos restantes simuladores analisados, fazendo referência ao artigo[51], onde

[1]Pesquisa efetuada para verificação do número de artigos com este tema, feita a 09-07-2013.

[2]Pesquisa efetuada para verificação do número de artigos com este tema, feita a 09-07-2013.

foram feitas as mesmas simulações nos diferentes simuladores e se obtiveram os resultados da Figura 4.1.

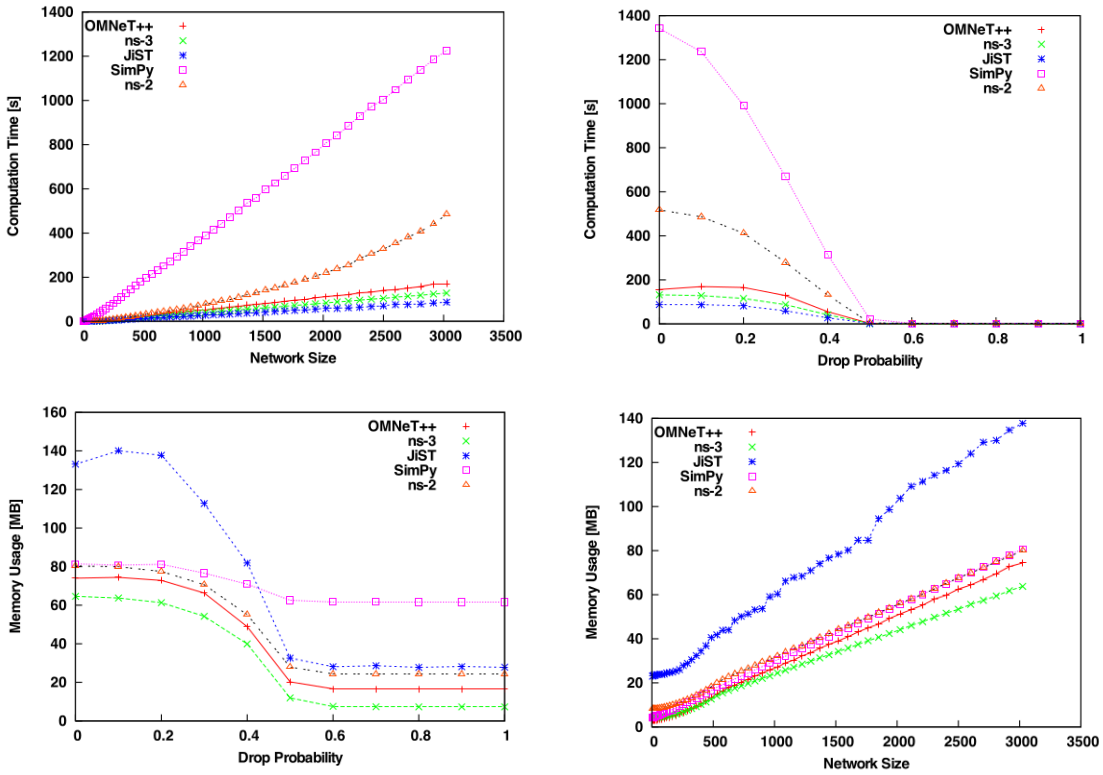


Figura 4.1: Resultados do desempenho dos vários simuladores, retirado de [51].

Daqui podemos concluir, que aumentando o tamanho da rede ou mesmo a probabilidade de perdas em relação ao tempo de computação, que o simulador NS-3 é um dos simuladores que melhores resultados obtém. Relativamente a testes de utilização de memória, podemos observar que o simulador NS-3 é o simulador que melhores resultados consegue com o aumento do tamanho da rede. Assim com base nas características do simulador e de acordo com os objetivos que pretendemos adquirir, de forma a criar um ambiente de testes eficiente e que se aproxime o mais possível da realidade, foi escolhida a plataforma de simulação NS-3 para a realização das nossas simulações.

4.2 Network Simulator 3 - NS-3

O Network Simulator 3 (NS-3) é um novo simulador de redes destinado a

substituir o antigo NS-2. É um simulador desenvolvido para efeitos académicos cujo desenvolvimento se iniciou em meados de 2006, constituído por modelos para TCP/IP, Wi-Fi, CSMA(Ethernet) e *links* ponto-a-ponto.

A escolha deste simulador, deve-se ao facto do mesmo ser um simulador recente e que surge com algumas vantagens em relação ao seu antecessor NS-2, principalmente devido à manipulação dos nós com múltiplas interfaces de rede e ao maior detalhe do modelo 802.11, entre outras. O NS-3 é organizado como um conjunto de vários módulos, conforme mostra a figura 4.2:

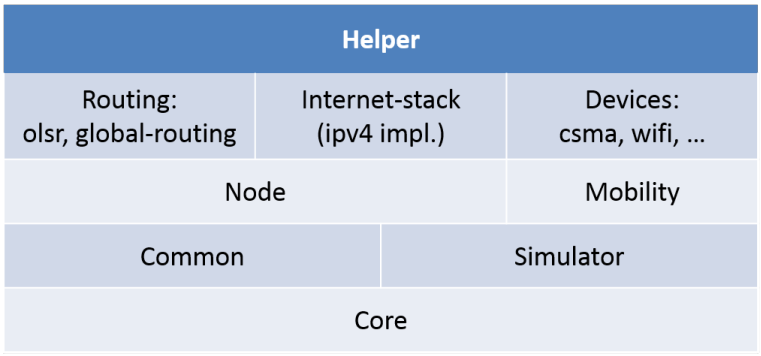


Figura 4.2: Arquitetura geral dos módulos do NS-3.

O módulo "core"oferece, adicionalmente à linguagem *python*, a sintaxe C++ para facilitar a programação e o desenvolvimento de módulos e *scripts* de teste.

Para além disto, a monitorização de redes em ambientes deste tipo, simulados, podem apresentar alguns problemas: os modelos de simulação podem não ser corretamente projetados/desenvolvidos e obtermos valores de simulação que possam divergir com os resultados do funcionamento dos protocolos reais. A recolha dos resultados das simulações pode exigir ao investigador o desenvolvimento de uma grande quantidade de código. Em NS-3 também é possível obter os resultados gerados por eventos, através de ficheiros de texto ou através dos ficheiros *.pcap* ou *.tr*. Uma das formas possíveis para armazenar eventos, em especial eventos de transmissão e receção de pacotes, é através dos ficheiros *.pcap*. No entanto estes sistemas de "seguir o rasto"são apenas alternativas ao sistema de rastreamento fornecido pelo NS-3.

A a favor do NS-3 temos, em resumo:

- É um simulador relativamente recente (lançado oficialmente em 2008);
- Tem um maior detalhe do modelo 802.11, do nosso interesse;
- É Opensource – GNU GPLv2;
- Está a ganhar cada vez mais, popularidade entre os investigadores de redes;
- Possibilidade de executar simulação distribuída;
- Possibilidade de correr código real e a ligação a ambientes reais.

Principais objetos do NS-3

O simulador NS-3 é composto por um conjunto de objetos. Estes objetos são agrupados e utilizados de forma a criarmos uma rede conforme os requisitos impostos para a simulação. Estes objetos correspondem a abstrações de entidades do mundo real, sendo alguns deles os seguintes:

Node

Um objeto do tipo *Node* representa um nó ou dispositivo de rede. De uma forma geral, contém um conjunto de objetos chave que são descritos de seguida, em que geralmente formam um encaminhador ou um terminal. Respetivamente ao nosso sistema, os *Node* são quaisquer dispositivos móveis que fazem parte da rede desenvolvida.

Net Device

Um objeto do tipo *NetDevice* representa uma interface de rede. Por exemplo uma interface *WiFi* ou *Ethernet* são consideradas como um *NetDevice*. Neste caso um *Node* poderá conter vários *NetDevice*.

Channel

O *Channel* faz a descrição do canal de comunicação entre dispositivos. Entre os respetivos *NetDevice* de um determinado tipo deve existir um *Channel* de um tipo específico. Este objeto do tipo *Channel* representa o canal de comunicação entre dois *NetDevice*. Se os *NetDevice* for do tipo *WiFi*, o *Channel* deve também ser do tipo *WiFi*.

Packet

O objeto do tipo *Packet*, tal como o nome indica, representa um pacote de dados. Estes *Packets* são compostos por um conjunto de octetos contendo cabeçalhos de protocolos e dados.

Application

Objetos do tipo *Application* representam processos (aplicações) que geram ou recebem tráfego, através da utilização da rede implementada para simulação. Referente à estrutura da nossa simulação desenvolvida, foram criados dois tipos de *Application* (SendPackets e ON/OFF). Geralmente as aplicações estão associadas a um objeto do tipo *Node*.

Socket

Os objetos do tipo *Socket* representam as ligações da camada de transporte. Estes Objetos, que geralmente se associam a um *NetDevice*, possuem um conjunto de funcionalidades para envio e receção de pacotes que são escalonadas e executadas seguindo uma ordem de escalonamento.

Escalonador de Eventos

Para qualquer simulação, o NS-3 corre um escalonador de eventos. Este escalonador é responsável pela execução de qualquer processo no NS-3. Ao contrário de implementações reais, onde quem escreve o código deve permitir e controlar múltiplos processos e interações entre estes, no NS-3 os processos são definidos como eventos e escalonados e executados sequencialmente. O NS-3 é pois um simulador discreto baseado em eventos.

Um evento, ao ser escalonado, deve ter associado um tempo de execução e uma função associada. Estas funções associadas poderão ser objetos do tipo *Callback*. Os processos referentes a alguns eventos podem assim ser modificados através de alteração da *Callback* associada.

4.3 Parâmetros de avaliação de desempenho

Descreve-se de seguida a lista de métricas quantitativas, que podem ser uti-

lizadas para avaliar o desempenho de protocolos de encaminhamento. Com base nestas métricas, serão executados vários testes nos protocolos referidos, analisando o seu desempenho e comparando o desempenho de cada um, e m diversas situações simulando ambientes reais:

Transferência de dados fim a fim(*end-to-end*) e atraso(*average delay*), é uma das principais medidas estatísticas a ter em conta para o cálculo do desempenho no encaminhamento de dados, obtendo valores que podem ser provocados por meios externos relativamente ao contexto da rede. Estas são as medidas de eficácia de uma política de encaminhamento. Estas medidas caraterizam o tempo desde a geração do pacote pela fonte até à sua receção na camada de aplicação do destino. Por conseguinte, esta medida inclui todos os atrasos na rede, tais como, filas de *buffer*, tempos de transmissão e atrasos induzidos por processos de encaminhamento.

Routing Overhead, é uma métrica que define o número de total de pacotes de controlo gerados pelo protocolo de encaminhamento e transmitidos pela rede, sendo expresso em bits por segundo ou pacotes por segundo. É muitas vezes proporcional ao número de nós que compõe a rede, mais propriamente o número de nós entre a fonte e o destino. À medida que o número de nós entre a fonte e o destino aumenta na rede, estando a fonte e o destino cada vez mais dependentes dos nós intermediários para encaminhar o tráfego, maior será o tráfego na rede devido à multiplicação do tráfego por difusão.

Tempo de aquisição de rota, é obtido contando o tempo entre o início de uma ligação e a ocorrência de troca de dados entre os dispositivos. Mais inclinado para os protocolos de encaminhamento *Reativos(on-demand)*, referindo-se ao tempo necessário para estabelecer uma rota(s) quando é requerida a comunicação.

Percentagem de entrega fora de ordem, reporta-se a uma medida externa do desempenho de encaminhamento, quando não existe ligação ou existe falha na ligação entre dispositivos. É de particular interesse aos protocolos da camada de transporte, como a camada TCP, que preferem na ordem de entrega.

Eficiência, é uma medida "interna"aos protocolos de encaminhamento. A eficiência do protocolo pode ou não afetar diretamente o desempenho do encaminhamento

de dados. Por exemplo, se o tráfego de pacotes de controlo e pacotes de dados partilharem o mesmo canal e a capacidade do canal for limitada, o excesso de pacotes de controlo poderá congestionar o canal e afetar o desempenho de encaminhamento de dados.

Para além destas métricas estudadas, existem outros rácios úteis para monitorizar a eficiência interna de um protocolo (havendo outros que os autores de [22, 13] não consideram):
O número médio de bits de dados transmitidos/bits de dados entregues; sendo isto considerado como uma medida na eficiência da entrega de dados.
O número médio de bits de controlo transmitidos/bits de dados entregues; este valor mede a eficiência do protocolo, dando assim uma contagem dos pacotes de controlo necessários trocar entre os dispositivos durante a permuta de pacotes de dados na rede, para que seja estabelecido o canal de comunicação entre os dispositivos. Assim pode-se analisar o *overhead*, que é criado pelos pacotes de controlo na rede.
O número médio de pacotes de controlo e de dados transmitidos/pacotes de dados entregues; este valor por sua vez, em vez de medir a eficiência algorítmica pura em termos de contagem de bits, tenta capturar a eficiência de um protocolo no que toca ao acesso do canal, devido ao custo de acesso ao canal ser alto, consequência da contenção assentada na camada de ligação.

Para estas métricas que avaliam o desempenho dos protocolos e estruturas de redes sem fios Ad Hoc, devemos considerar o "contexto"em que o desempenho é medido. Para isto, os parâmetros essenciais que deverão ser variados são os seguintes:

1. o tamanho da rede, alterando o número de nós que formam a rede;

2. a conectividade da rede, que consiste no número médio de vizinhos de um nó (grau médio de um nó);

3. percentagem de mudança da topologia, resume-se à velocidade com que uma topologia de rede muda a sua estrutura geográfica (velocidade dos nós);

4. capacidade de ligação, velocidade efetiva do link de ligação em bits/segundo, após a contabilização de perdas devido ao múltiplo acesso;

5. aplicação ou padrões de tráfico, sendo um dos parâmetros essenciais no desempenho dos protocolos de redes. Este parâmetro descreve a adaptação dos protocolos aos padrões de tráfego uniforme, não uniforme ou por rajadas;

6. A mobilidade dos dispositivos na rede, tratando um estudo de quando e em que circunstâncias a correlação temporal e espacial é relevante para o desempenho de um protocolo de encaminhamento.

Com o uso destes e outros parâmetros de avaliação dos protocolos de rede Ad Hoc, faz-se a analise e teste aos protocolos referenciados ao longo do capitulo, avaliando o desempenho de cada um e a comparação dos resultados oferecidos por cada um destes. Contudo, um protocolo Ad Hoc deve funcionar de forma eficaz através de uma grande variedade de contextos de funcionamento e do ambiente onde é implementada.

Em resumo, as características e métricas de avaliação de desempenho dos protocolos de rede apresentados, diferenciam as redes Ad Hoc relativamente às redes tradicionais com fios ou redes infraestruturadas. De certa forma, nestas questões de avaliação do desempenho dos protocolos de redes sem fios Ad Hoc, destacam-se as métricas de desempenho que podem ajudar e promover comparações significativas do desempenho do protocolo em cada ambiente simulado. Assim, reconhece-se que cada um dos protocolos de redes Ad Hoc tendem a ser os mais adequados para contextos particulares das redes e menos adequados para outros.

4.4 Construção do cenário de testes

De modo a se fazer uma comparação do desempenho de cada um dos protocolos para cada tipo de cenário, cenários estes onde se fazem variar parâmetros como velocidade dos nós, número de nós em um espaço da rede definido, o modo de mobilidade dos nós nesse espaço, o protocolo a ser utilizado e por fim alterando a aplicação entre duas aplicações de envio de pacotes entre um nó fonte e nó

destino, desenvolveu-se uma *script* de teste de simulação onde tudo isto poderá ser configurado e simulado na plataforma descrita em 4.2, constituída pelos seguintes passos:

- Inicialmente estabelece-se a topologia de rede com a criação dos nós consoante o número definido por parâmetro;
- Configura-se os canais de ligação, nível físico (PHY) e nível MAC, entre os nós consoante a descrição do modelo *YANS*;
- Associa-se o canal ao meio físico (sem fios), de forma a partilharem o mesmo meio;
- Define-se o modelo de encaminhamento Ad Hoc que a simulação irá utilizar na camada MAC;
- Constrói-se os dispositivos *Wi-Fi* em cada nó com os parâmetros MAC e PHY definidos;
- Adiciona-se o modelo de mobilidade aos nós criados;
- Define-se e instala-se a pilha protocolar do protocolo que será utilizado no encaminhamento de tráfego entre os nós;
- Atribui-se os endereços IPv4 a cada interface de cada nó;
- Escolhe-se a aplicação que irá funcionar durante o período de simulação;
- E por fim, configuram-se os parâmetros e os métodos que serão utilizados para recolha dos resultados das simulações de forma a gravar esses resultados em ficheiro no final das simulações.

Para cada simulação, o número de nós (*nWiFi*), o tipo de mobilidade (*mobilidade*), o tipo de aplicação (*application*), o protocolo a utilizar (*protocol*) e o número de simulações (*numSimulation*) entram como parâmetro. Isto faz com que seja possível para a mesma estrutura de rede, modificando apenas um destes valores, seja feita uma simulação com uma estrutura diferente.

```
CommandLine cmd;
 cmd.AddValue ("protocol", "1=AODV; 2=OLSR; 3=DSDV", protocol);
 cmd.AddValue ("nWiFi", "Number of wifi STA devices", nWifi);
```

```
cmd.AddValue ("application", "2=SendPackets; 3=ON/OFF SendPackets
    ", application);
cmd.AddValue ("mobility", "Tipe of mobility model the nodes. 1 or
     2", mobilidade);
cmd.AddValue ("numSimulation", "The number of simulations",
    numSimulation);
cmd.Parse (argc,argv);
```

Na construção dos dispositivos móveis (nós), sendo estes dispositivos *Wi-Fi*, torna-se necessário configurar os canais de ligação, métodos de acesso, protocolos e algoritmos adequados para a comunicação entre os dispositivos. Deste modo, consoante a descrição do modelo *Yans* definimos a camada de rede. Este modelo *Yans* [34] descreve a implementação do modelo de propagação, baseando-se na implementação da camada física (PHY) e da camada MAC conforme a especificação 802.11a.

Devido às dificuldades conhecidas por diversos pesquisadores e investigadores no domínio da rede e à sua aplicação em redes reais, os simuladores de rede tornaram-se uma ferramenta indispensável para a investigação e validação de ideias em todas as camadas da rede. No entanto, muitos dos investigadores de redes não estão familiarizados com as implicações dos pressupostos que devem ser definidos na camada física em diferentes cenários. Para dar resposta a estes problemas, vários investigadores estudaram e projetaram um modelo da camada física 802.11a quase realista, com todos os fenómenos associados a esta camada.

À medida que são formados os nós da rede, estes podem ter uma coleção de objetos "NetDevice", comparável com um computador real que é constituído por placas com diferentes interfaces de ligação, *Ethernet*, *Wi-Fi*, *Bluetooth*, etc.

O modelo 802.11 foi desenvolvido independentemente do modelo *Yans* conhecido, sendo que este último foi originalmente escrito para o NS-2. Este modelo implementa um modelo PHY e um modelo MAC, que estão em conformidade com a especificação 802.11a [8] [14] (utilizada no desenvolvimento deste projeto). Foi utilizado o modelo de *Yans* para descrever o canal de propagação *Wi-Fi* seguindo a seguinte configuração:

```
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
```

```
YansWifiPhyHelper phy = YansWifiPhyHelper::Default();
                phy.SetChannel (channel.Create());

WifiHelper wifi = WifiHelper::Default();
        wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default();
```

Aqui é feita a definição dos dispositivos *Wi-Fi* e os canais de ligação entre eles, utilizando o modelo *Yans* por defeito como já referido. Primeiro, criamos o canal de comunicação entre os dispositivos definindo o canal por defeito do modelo, definindo a camada de rede (canal), e assim à posteriori adicionar a esse canal a camada física *YansWifiPhyHelper phy* associando o canal ao meio físico de forma a partilharem o mesmo meio *phy.SetChannel (channel.Create());*. Em seguida definimos o tipo de algoritmo de controlo de débito, o AARF (*Adaptive AutoRate Fallback*), para a camada MAC e PHY configurada por defeito para a o modelo 802.11a, descrito pelo código intermédio do bloco, que implementa o algoritmo de controlo de taxa AARF, inicialmente descrito pelo IEEE por Taxa de Adaptação. E por fim definimos a camada MAC sem QoS.

Depois de definido o canal ou o meio de transmissão, define-se o tipo de protocolo Ad Hoc que que utiliza a simulação na camada MAC, criando no final os dispositivos *Wi-Fi* com os parâmetros MAC e PHY configurados até aqui e instalando assim estes parâmetros nos respetivos nós criados.

```
wifiMac.SetType ("ns3::AdhocWifiMac");

NetDeviceContainer devices;
 devices = wifi.Install (phy, wifiMac, nodes);
```

Deste modo, seguimos a configuração deste modelo descrito em [34], porque se verificou que este é um dos modelos mais utilizados na área de simulações de redes *Wi-Fi*.

4.4.1 Modelo YANS em 802.11

Esta secção apresenta resumidamente as características do modelo 802.11, já

referenciadas na secção 2.1, em *Yans*. Ambas as camadas MAC (*Medium Access Control*) e PHY (*Physical Layer*) são tratadas, embora a camada MAC não seja muito aprofundada no desenvolvimento do nosso trabalho, apenas se pretende mencionar, resumidamente, as funcionalidades disponíveis do módulo MAC existente. Na camada física, no entanto, será feita uma observação mais profunda sobre a sequência de ações tomadas durante a receção de pacotes. Mais detalhe sobre as especificações das camadas MAC e PHY no standard 802.11 é descrito nos documentos [30][14].

Modelo YANS

O *Yans* é um protótipo de um simulador de redes desenvolvido pelo grupo "INRIA's Planète group". O principal objetivo do desenvolvimento deste simulador, *Yet Another Network Simulator* (*Yans*), tem sido construir um ambiente com um núcleo baseado em eventos. O código base, devido à parceria do grupo "Planète group"com o grupo "NS-3 project initiative"[4], tem vindo a ser portado para o simulador NS-3. O módulo do *Yans* é composto, devido ao interesse de investigação, pelo módulo IEEE802.11. De seguida resumimos os principais recursos existentes neste módulo que são aplicados como descrito até ao momento.

Camada MAC

A camada MAC tem como principal funcionalidade, fornecer fiabilidade nos mecanismos de entrega de dados através da interface sem fios. É através desta camada que se realiza o controlo de acesso ao meio comum. A implementação do modelo MAC em *Yans* tanto dá suporte ao modo Ad Hoc como ao modo com infraestrutura. No modo Ad Hoc a função DCF(*Distributed Coordination Function*) é implementada juntamente com a nova função DCF com QoS em IEEE802.11e, isto é, tratada por *Enhanced DCF Channel Access*(EDCA). No modo de rede infraestruturada, temos a função HCF(*Hybrid Coordination Function*) e *HCF Controlled Channel Access*(HCCA) implementadas no simulador. Neste modelo, a função de coordenação distribuída 802.11 é utilizada para se calcular o momento e tempo de acesso ao meio de transmissão.

DCF A DCF é a camada básica do modelo 802.11. Esta função utiliza o método CSMA/CA para acesso ao meio de comunicação entre os nós. Pode também, opcionalmente, utilizar o método de RTS/CTS.

PCF A função *Point Coordination Function* (PCF) é outra função básica de controlo, esta função apenas é definida no modo de rede com infra-estrutura, onde as estações se ligam a um ponto de acesso[33].

Neste trabalho, num entanto, como apenas nos referimos ao modo Ad Hoc dá-mos mais ênfase às questões da camada física, PHY.

Implementação da camada PHY no *Yans*

A camada física tem como principal funcionalidade, a ligação entre a camada MAC e o meio de transmissão(no nosso caso o ar).

Nesta camada são tratados todos os campos relacionados com os modelos de propagação, modelação, esquemas de codificação(FEC-*Forward Error Correction*) e métodos de cálculo(BER-*Bit Error Rate*[3] e PER-*Packet Error Rate*[4]). É através da camada física que se define as taxas de transmissão possíveis, as faixas de frequência de funcionamento e os números de canais disponíveis [30] [14] [33]. No entanto apenas nos concentramos na mecânica da camada física relativamente ao padrão IEEE802.11a, devido à sua maturidade e à vasta utilização ao nível das redes, esclarecendo as ações tomadas quando um pacote é recebido. Como mencionado pelos autores em [48], os seguintes fatores da camada física são muito importantes para a avaliação do desempenho da camada superior dos protocolos.

- Método de receção do sinal (*BER-based ou SNRT-based*)
- Perda de rota (*Path Loss, Fading*)
- Interferência e ruído computacional

[3]BER corresponde ao número de bits ou blocos de bits recebidos incorretamente, relativamente ao número total de bits enviados durante um intervalo de tempo específico.

[4]PER é o número de pacotes recebidos incorretamente, dividido pelo número total de pacotes de dados recebidos. Um pacote é considerado incorreto se pelo menos um bit desse pacote der erro.

- Tamanho físico do cabeçalho

Como o "Yans" é um simulador que se baseia em eventos, este consiste em primeiro lugar num evento de início de receção (primeiro bit do pacote) e em segundo lugar, um evento de fim de pacote (último bit do pacote). Deste modo a função SNIR(t)[5] é avaliada duas vezes para cada pacote. Para o primeiro bit, decidindo se deve ou não receber o pacote, considerando o estado atual do PHY e do nível de SNIR(t), e para o último bit, de forma a calcular o valor final da função SNIR(t), considerando o que aconteceu durante a receção dos pacotes através do valor de PER(*Packet Error Rate*).

A camada PHY pode estar em um dos quatro estados possíveis:

TX o estado TX define o momento em que a camada PHY está a transmitir um sinal. No momento em que a PHY está neste estado, um pacote recebido vai ser descartado independentemente do nível de SNIR(t).

SYNC neste estado a PHY é sincronizada com o sinal e espera até que seja recebido o último bit. Enquanto a PHY está neste estado, outro pacote recebido será descartado independentemente do seu nível de SNIR(t).

BUSY a PHY não está em estado TX ou SYNC, mas a energia medida no meio de comunicação é superior ao limiar de deteção de energia(*Energy Detection Threshold*). Enquanto a PHY estiver neste estado, um pacote poderá ser recebido se o seu valor de SNIR(t) for superior ao limiar.

IDLE neste caso o PHY não está em nenhum dos estados indicados acima. O comportamento é o mesmo como no estado BUSY, ou seja, enquanto o PHY estiver neste estado, um pacote poderá ser recebido se o seu valor de SNIR(t) for superior ao limiar.

Ao receber o último bit do pacote, em que a PHY está sincronizada, é novamente avaliada a função SNIR(t) e calculado o valor de PER.

[5]SNIR(t) descreve a função da relação sinal ruído mais interferência do sinal. Esta função é a razão entre a potência do sinal com a combinação da potência do ruído e a potência de interferência.

Estudado o modelo descrito até aqui, foi então criado um modelo de canal padrão *Yans* que por defeito contém um atraso de propagação (*delay*), igual ao valor constante da velocidade da luz, e uma perda de propagação (*loss*) com base no modelo de distância entre dispositivos. Este modelo representa uma perda por referência de 46,6777dB a uma distância de referência de 1m. Simultaneamente, definimos a camada física utilizando o modelo *Wi-Fi* 802.11a, que como mostrado antes, este modelo de canal físico (PHY) depende de uma perda do canal e um modelo de atraso conforme definido na classe "ns3::PropagationLossModel"e "ns3::PropagationDelayModel", sendo estes, membros da classe "ns3::YansWifiChannel". Na *script*, não atribuímos valores as estes parâmetros, utilizando os valores por defeito, atribuindo apenas na camada física ganho ao emissor ("TxGain") e ganho ao recetor ("RxGain"), alterando o alcance de transmissão e receção dos dispositivos móveis. Ao atribuirmos igual valor a estes dois parâmetros, de 2.0 dBm, obtemos um alcance de aproximadamente 130 metros em cada nó ou dispositivo, sendo um valor que achamos razoável praticamente em média a todos os dispositivos móveis, obtendo assim cada dispositivo uma área de cobertura de aproximadamente $53093m^2$. Através destes parâmetros modelizamos o canal de propagação através da sua associação ao meio físico. Ainda nos dispositivos *Wi-Fi*, definimos o algoritmo AARF para controlo de taxa de débito através da classe "ns3::AarfWifiManager", sendo este algoritmo descrito em [35].

Estando a rede criada, é então definido o protocolo que estes nós irão utilizar para cada simulação e feita a instalação da respetiva pilha protocolar, acrescentando o algoritmo de encaminhamento do próprio protocolo Ad Hoc escolhido (através configuração apresentada em baixo). Para isso é necessário a definição de um objeto *helper* para criar uma lista de prioridade de protocolos, que no nosso caso terão todos a mesmo nível de prioridade, "100".

```
AodvHelper aodv;
OlsrHelper olsr;
DsdvHelper dsdv;

Ipv4ListRoutingHelper list;
case 1:
```

```
     list.Add (aodv, 100);
case 2:
     list.Add (olsr, 100);
case 3:
     list.Add (dsdv, 100);
```

De forma a finalizar este processo é feita a instalação da pilha protocolar, acrescentando o algoritmo de encaminhamento Ad Hoc, como referido, e instaladas as suas características em cada nó, fazendo também a atribuição dos endereços IPv4 a cada interface dos nós(*devices*), representado pelo seguinte código:

```
InternetStackHelper stackprotoc;
 stackprotoc.SetRoutingHelper (list);
 stackprotoc.EnableAsciiIpv4All (stream);
 stackprotoc.Install (nodes);

Ipv4AddressHelper ipv4_address;
 NS_LOG_INFO("Assign IPv4 Addresses.");
 ipv4_address.SetBase ("10.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer interfaces;
 interfaces = ipv4_address.Assign (devices);
```

4.4.2 Definição dos modelos de mobilidade

Terminando a criação e definição dos nós e do meio de transmissão, é escolhido o modelo de mobilidade. Estes modelos de mobilidade caracterizam a forma como os nós se deslocam no espaço da rede, as suas velocidades, as suas orientações e como se dispõe inicialmente dentro desse espaço. Para este ponto foram desenvolvidos 2 modelos de mobilidade. Estes modelos são associados a um objeto "mobility"onde será adicionado o modelo de mobilidade escolhido, criado inicialmente pela instrução

```
MobilityHelper mobility;
```

Modelo de Mobilidade 1

Na utilização do modelo de mobilidade 1, os nós são colocados aleatoriamente

dentro de uma área plana definida, limitada pelos valores atribuídos às variáveis "limiteX"e "limiteY". Dentro desta área, os nós são dispostos em posições (x,y) aleatoriamente por todo o espaço de rede. Esta disposição dos nós na rede é estabelecida através da definição de parâmetros da classe "ns3::RandomRectanglePositionAllocator", que consiste em alocar os nós criados anteriormente dentro de um retângulo definido de acordo com o par de variáveis aleatórias("limiteX"e "limiteY"). Estando os nós posicionados e definido o espaço por onde estes se podem movimentar, requer agora caracterizar o modo como estes se vão movimentar nesse espaço. Para este processo, utilizou-se as funcionalidades disponíveis da classe

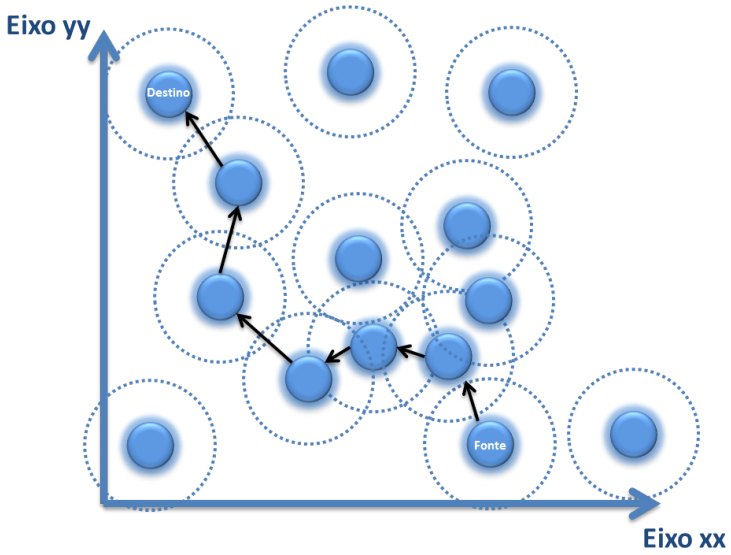


Figura 4.3: Representação do modelo de mobilidade 1.

"ns3::RandomWaypointMobilityModel". Este modelo de mobilidade consiste em uma movimentação aleatória dos nós dentro do espaço da rede em direção a um ponto especifico. No momento inicial, cada objeto (nó) faz uma pausa no tempo (de acordo com um valor aleatório definido dentro de um intervalo de valores atribuído à variável "Pause"). Após esta pausa, cada nó vai escolher um novo ponto aleatório dentro do espaço da rede(*waypoint*) através de um objeto ponteiro "PositionAllocator"(que contém as coordenadas atribuídas aleatoriamente dentro dos limites impostos anteriormente) e um novo valor de velocidade de acordo com a variável "Speed", começando assim a se mover em direção a esse *waypoint* com

uma velocidade constante. Estas variáveis "Pause"e "Speed"são constituídas por valores aleatórios dentro de uma gama definida, fazendo com que o valor da variável "Speed"seja um valor inteiro uniforme dentro do intervalo Min=0.0 a Max=2 m/s, sendo esta velocidade (2m/s) considerada a velocidade média de passeio de uma pessoa, em que "m_nodeSpeed=2"e "m_nodePause=1"(código seguinte).

```
std::stringstream speed;
speed<<"ns3::UniformRandomVariable[Min=0.0|Max="<< m_nodeSpeed<<"]
    ";
std::stringstream pause;
pause<<"ns3::ConstantRandomVariable[Constant="<< m_nodePause<<"]";
```

Quando esse nó chega a esse destino, o processo inicializa após fazer uma pausa no tempo de acordo com o tempo definido na variável "Pause"e repete todo o processo descrito. Este processo realiza-se durante todo o tempo da simulação.

Assim, os nós vão-se movimentando de forma aleatória com variação da direção ao longo do seu percurso. O estrato de código que é apresentado de seguida mostra como este modelo de mobilidade é implementado:

```
double limiteX = 500;
double limiteY = 500;
 std::stringstream distX;
 distX << "ns3::UniformRandomVariable[Min=0.0|Max=" << limiteX <<
    "]";
 std::stringstream distY;
 distY << "ns3::UniformRandomVariable[Min=0.0|Max=" << limiteY <<
    "]";
 ObjectFactory pos;
  pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
  pos.Set ("X", StringValue (distX.str()));
  pos.Set ("Y", StringValue (distY.str()));

 Ptr <PositionAllocator> taPositionAlloc = pos.Create ()->
    GetObject <PositionAllocator> ();
  mobility.SetMobilityModel ("ns3::RandomWaypointMobilityModel",
   "Speed", StringValue (speed.str()),
   "Pause", StringValue (pause.str()), "PositionAllocator",
      PointerValue (taPositionAlloc));
```

```
mobility.SetPositionAllocator (taPositionAlloc);
```

As dimensões do espaço de rede por onde os nós irão ser dispersos foi escolhido de modo a simular o Centro Histórico de Guimarães, ou um outro espaço, simulando um número de nós(dispositivos) a circularem pelo Centro Histórico, numa área total de $250000m^2$.

Modelo de Mobilidade 2

O modelo de mobilidade 2, comparativamente com o modelo de mobilidade 1, neste é criada uma área onde os nós se vão movimentar com diferente disposição, sendo desta vez uma área tridimensional. Neste modelo pretende-se distribuir os nós dentro de um bloco 3D aleatoriamente. Desta forma, fazendo uso dos parâmetros oferecidos pela classe "ns3::RandomBoxPositionAllocator"os nós são colocados dentro de uma "caixa 3D"de acordo com um conjunto de três variáveis definidas "limiteX", "limiteY"e "limiteZ", onde o "limiteZ"pretende simular o valor de altura máxima possível, podendo ser interpretada como o número de andares (altura) em um edifício ou a altura máxima de uma plataforma. Deste modo podermos simular a movimentação dos dispositivos em espaços tridimensionais (por exemplo espaços comerciais ou habitações) ou até mesmo a deslocação de um avião no espaço tridimensional. Relativamente à forma como se movimentam

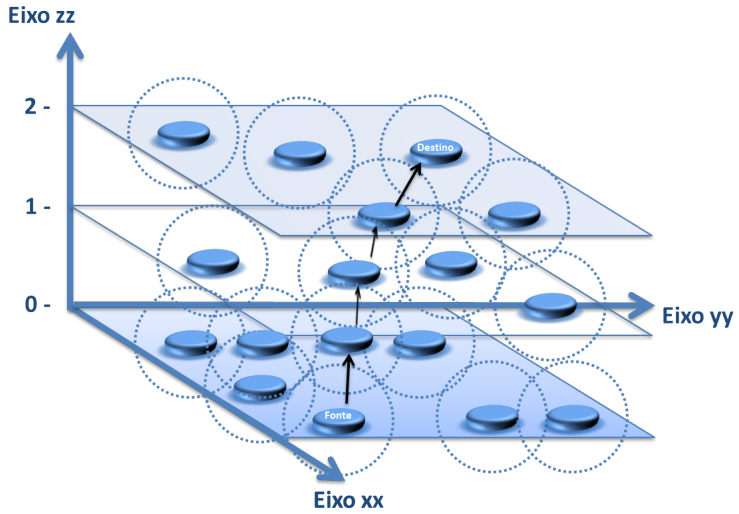


Figura 4.4: Representação do modelo de mobilidade 2.

os nós, este modelo de mobilidade torna-se mais complexo, devido aos nós serem distribuídos aleatoriamente dentro de um bloco 3D onde agora terá três variáveis chave, sendo elas a direção (para onde os nós se dirigem), velocidade dos nós e altura (altura da plataforma). Assim é utilizado uma versão do modelo de mobilidade *Gauss-Markov* descrito em [17], sendo este modelo concebido para se adaptar a diferentes níveis de aleatoriedade através de parâmetros ajustáveis. Este modelo diferencia-se dos restantes disponíveis no simulador NS-3 devido a este modelo ter memória e variabilidade. É constituído por um parâmetro alfa, ajustável, que determina a quantidade de memória e causalidade que se pretende modelar. Inicialmente, a cada nó é atribuída uma velocidade, direção (radianos) e ângulo de inclinação (radianos) equivalente as variáveis "MeanVelocity", "MeanDirection e "MeanPitch". A cada intervalo fixo no tempo definido pela variável "TimeStep", uma velocidade, direção e ângulo são gerados com base no valor anterior, o valor médio e uma variável gaussiana aleatória. Especificamente, os valores da velocidade e direção em um instante t é calculado com base no valor da velocidade e direção no instante $(t-1)^{st}$ e uma variável aleatória através das seguintes equações:

$$s_t = \alpha s_{t-1} + (1-\alpha)\overline{s} + \sqrt{(1-\alpha^2)} \times s_{X_{t-1}}$$

$$d_t = \alpha d_{t-1} + (1-\alpha)\overline{d} + \sqrt{(1-\alpha^2)} \times d_{X_{t-1}}$$

onde s_t e d_t são agora a nova velocidade e direção para o determinado nó no intervalo de tempo t, onde $0 < \alpha < 1$. Para obtermos valores de velocidade e direção totalmente aleatórios, obtendo um movimento *Brownian*[6], terá se ser obtido um valor de $\alpha = 0$ e caso contrário, para obter-mos um valor continuo com movimento linear teremos de obter um $\alpha = 1$. No nosso caso, com níveis intermédios de aleatoriedade, definimos um α com valor de 0.85, obtendo alguma memória relativamente aos valores passados. A cada intervalo de tempo definido pela variável "TimeStep", a próxima localização é calculada em função da posição,

[6]Movimento *Brownian* é um movimento aleatório de elementos, descoberto em 1827 pelo botânico inglês Robert Brown quando este fazia um estudo sobre partículas de pólen. Descreve este movimento como um movimento aleatório continuo de partículas sólidas quando em suspensão num fluido.

velocidade e direção do movimento atual. Especificamente, num intervalo de tempo t, uma posição dos nós móveis é dada pelas seguintes equações:

$$X_n = X_{n-1} + S_{n-1} \cos d_{n-1}$$

$$Y_n = Y_{n-1} + S_{n-1} \sin d_{n-1}.$$

Através deste modelo de mobilidade, conseguimos simular cenários diferentes onde a direção, velocidade e altura são as variáveis chave. De certa forma, atribuímos valores às variáveis "limiteX", "limiteY"e "limiteZ"os valores de 200m, 200m e 15m respetivamente. Estes valores tem como base a simulação da movimentação dos nós de forma a simular o Centro Comercial Colombo(Lisboa) que tem aproximadamente $120000m^2$ constituído por 3 andares.

```
double limiteX = 200;
double limiteY = 200;
double limiteZ = 15;
 std::stringstream distX;
 distX << "ns3::UniformRandomVariable[Min=0.0|Max=" << limiteX <<
    "]";
 std::stringstream distY;
 distY << "ns3::UniformRandomVariable[Min=0.0|Max=" << limiteY <<
    "]";
 std::stringstream distZ;
 distZ << "ns3::UniformRandomVariable[Min=0.0|Max=" << limiteZ <<
    "]";
  mobility.SetMobilityModel ("ns3::GaussMarkovMobilityModel",
   "Bounds", BoxValue (Box (0, limiteX, 0, limiteY, 0, limiteZ)),
   "TimeStep", TimeValue (Seconds (1.5)),
   "Alpha", DoubleValue (0.85),
   "MeanVelocity", StringValue ("ns3::UniformRandomVariable[Min=1|
      Max=4]"),
   "MeanDirection", StringValue ("ns3::UniformRandomVariable[Min
      =0|Max=6]"),
   "MeanPitch", StringValue ("ns3::UniformRandomVariable[Min=0.05|
      Max=0.05]"),
   "NormalVelocity", StringValue ("ns3::NormalRandomVariable[Mean
      =0.0|Variance=0.0|Bound=0.0]"),
```

```
    "NormalDirection", StringValue ("ns3::NormalRandomVariable[Mean
        =0.0|Variance=0.2|Bound=0.4]"),
    "NormalPitch", StringValue ("ns3::NormalRandomVariable[Mean
        =0.0|Variance=0.02|Bound=0.04]"));
  mobility.SetPositionAllocator ("ns3::RandomBoxPositionAllocator"
      ,
    "X", StringValue (distX.str()),
    "Y", StringValue (distY.str()),
    "Z", StringValue (distZ.str());
```

Este excerto de código apresentado, representa toda a configuração necessária para implementação do modelo de mobilidade 2 nos nós.

Para finalizar, feita a escolha do modelo de mobilidade, esta é instalada em todos os nós criados, através da seguinte instrução:

```
mobility.Install (nodes);
```

instalando o modelo de mobilidade escolhido em todos os nós criados.

4.4.3 Definição das aplicações geradoras de tráfego

Finalizada toda a implementação da rede e as suas configurações, falta escolher qual a aplicação a ser aplicada para iniciar a sua simulação. Desta forma, foram desenvolvidas duas aplicações com diferentes características de modo a simular aplicações reais. Antes disso, para que todo o processo de estruturação e construção da rede fique definida, é necessário definir qual será o nó fonte e o nó destino entre os quais será realizada a transferência de pacotes, para que a instalação das respetivas aplicações sejam realizadas nos nós adequados. Assim a nossa escolha recai em definir aleatoriamente o nó fonte e o nó destino a cada simulação. Desta forma, para obtermos sempre um valor aleatório para cada simulação e para o qual, repetindo a mesma simulação obtenhamos os mesmo valores comparativamente com outras simulações com os mesmos parâmetros, utilizamos um gerador de números aleatórios do NS-3 ("ns3::SeedManager"), representado pelas seguintes instruções:

```
ns3::SeedManager::SetSeed (m_seed);
ns3::SeedManager::SetRun (m_run);
```

```
Ptr<UniformRandomVariable> m_randVar = CreateObject<
    UniformRandomVariable>();
```

obtendo para cada simulação um valor de "m_seed"e "m_run"que são característicos de cada simulação. Com isto podemos verificar que a mesma simulação (com os mesmo parâmetros definidos) obtém sempre os mesmos valores e a simulação obtém sempre o mesmo resultado. Utilizando então esta variável aleatória criada "m_randVar", obtemos um valor para o nó fonte e para o nó destino, dentro do limite do número de nós criados (*nWifi-1*), representado pelo seguinte código:

```
uint32_t no_source = m_randVar->GetInteger(0, nWifi-1);
uint32_t no_dest = m_randVar->GetInteger(0, nWifi-1);
 while (no_source == no_dest) {
  no_dest = m_randVar->GetInteger(0, nWifi-1);
 }
```

Sempre que é obtido o valor do nó fonte e nó destino é sempre feito um teste para verificar se estes dois parâmetros não obtiveram o mesmo valor, o que é muito raro ou quase impossível acontecer numa gama elevada de nós, mas que será necessário descartar no caso de acontecer, para que não se obtenha a situação de o nó fonte ser designado também como o nó destino. Com isto estamos prontos para definir a aplicação e instala-la no respetivo nó.

Aplicação 1(SendPackets)

Depois de escolhidos os nós "servidor"e "cliente", é então instalada uma das aplicações disponíveis. Neste caso, a aplicação 1, SendPackets, consiste na ligação do nó fonte com o nó destino através de um canal UDP. Este canal faz a ligação entre os endereços dos nós escolhidos, endereços estes obtidos pelos nós designados fonte e destino da rede. Inicialmente e depois de estabelecida a comunicação entre os dois nós, é configurado um evento para iniciar o envio de dados. Este evento é invocado no segundo 15.0 da simulação e chama uma função geradora de pacotes, passando para essa função os parâmetros "tamanhoPacote"em bytes, "numPacotes"e "intervaloPacote"definidos. Dentro desta função é criado um pacote com tamanho definido pela variável "tamanhoPacote"e enviado para o endereço do nó

destino.

```
uint32_t tamanhoPacote = 1500;
uint32_t numPacotes = 4090;
double interPacote = 0.033;
```

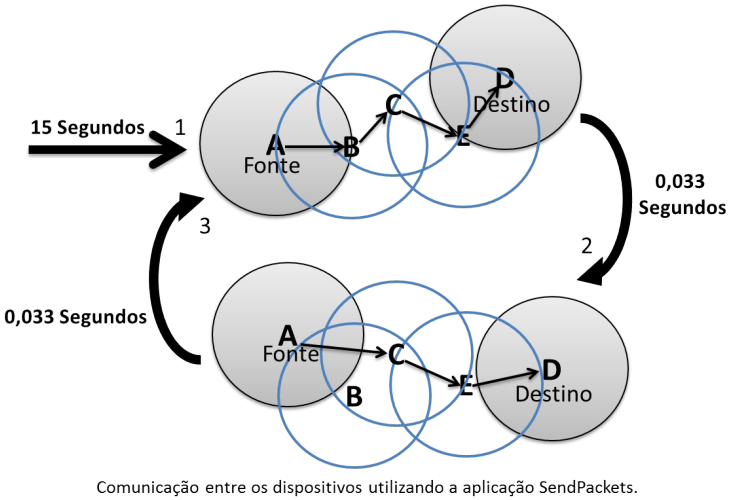


Figura 4.5: Representação do funcionamento da aplicação SendPackets.

Esta função invocada pelo evento característico é então repetida de "interPacote"em "interPacote"segundos, que vai enviando pacotes desde o nó fonte para o nó destino até ao final da simulação ou até que seja atingido o número de pacotes definidos pelo utilizador a enviar, configurado através da variável "numPacotes"ou até que a aplicação termine. Este valor "numPacotes"foi escolhido de forma a ser o número máximo que a aplicação pode enviar dentro do tempo de simulação sem interrupções no envio. A criação do pacote a enviar é realizada através do método *Create<Packet> (tamanhoPacote)*, que cria o pacote com um tamanho recebido por parâmetro preenchido a zeros. A memória necessária para o tamanho do pacote não está alocada, sendo alocada num momento posterior, sendo este pacote preenchido com zeros até atingir o tamanho definido. A cada pacote é atribuído um novo UID(*Unique Identification Number*) que identifica cada pacote a enviar.

A aplicação SendPackets é assim definida pelo seguinte código.

```
TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
 Ptr<Socket> recvSink = Socket::CreateSocket (nodes.Get (no_dest),
     tid);
```

```
InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny
    (), 80);
 recvSink->Bind (local);
 recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

Ptr<Socket> source = Socket::CreateSocket (nodes.Get (no_source),
    tid);
InetSocketAddress remote = InetSocketAddress (Ipv4Address (addr),
    80);

source->SetAllowBroadcast (true);
source->Connect (remote);

Simulator::ScheduleWithContext (source->GetNode ()->GetId (),
   Seconds (15.0), &GenerateTraffic, source, tamanhoPacote,
   numPacotes, intervaloPacote, start\_node);
```

Este código (função) representa então a criação do caminho entre o nó fonte e o nó destino, recebidos por parâmetro, utilizando *Sockets UDP* e inicia uma aplicação de geração de tráfego, *GenerateTraffic*, passando como argumentos para essa função um ponteiro para o *socket* criado entre os nós, o tamanho do pacote a ser criado, o numero de pacotes que serão enviados, o intervalo de tempo que essa função será chamada para enviar um pacote e o endereço do nó fonte, que enviará o pacote de dados.

Esta função que trata da geração de pacotes é assim definida no simulador da seguinte forma:

```
static void GenerateTraffic (Ptr<Socket> socket, uint32_t
   tamanhoPacote, uint32_t contadorPacote, Time intervaloPacotes,
   Ptr<Node> start_node){
 if (contadorPacote > 0){
  socket->Send (Create<Packet> (tamanhoPacote));
  TransmitePacket (start_node);
  Simulator::Schedule (intervaloPacotes, &GenerateTraffic, socket,
      tamanhoPacote, contadorPacote-1, intervaloPacotes,
     start_node);
 }else{
```

```
  socket->Close ();
  }
}
```

Esta aplicação pretende descrever uma aplicação real, como por exemplo uma aplicação de vídeo do tipo *video-on-demand*, que envia pacotes continuamente, sendo que os valores escolhidos definindo o modo como a aplicação opera, tamanho dos pacotes e o intervalo de tempo de envio entre pacotes têm como referência a codificação de video para transmissão em redes móveis [47].

Aplicação 2(ON/OFF)

A aplicação 2 foi desenvolvido e configurada de forma a simular o exemplo de uma conversa (ex. chamada telefónica). Esta aplicação consiste na geração de tráfego de dados para um destino determinado, nó destino ou cliente, com um padrão ON/OFF. Este padrão ON/OFF descreve os momentos de geração de tráfego na rede e os momentos em que não se gera tráfego de dados na rede, isto é, quando um nó fonte está durante um intervalo de tempo, a gerar tráfego na rede em direção ao nó destino (estado ON) e quando esse mesmo nó fonte está um intervalo de tempo parado sem gerar tráfego de dados na rede (estado OFF).

São inicialmente configurados os parâmetros básicos da aplicação, o tamanho dos pacotes a enviar "ns3::OnOffApplication::PacketSize", a taxa de transmissão da aplicação "ns3::OnOffApplication::DataRate"e os tempos de funcionamento ON e OFF representados pelos parâmetros "OnTime"e "OffTime".

```
std::string sizePacket ("372");
std::string AppPacketRate ("12.2kbps");
Config::SetDefault ("ns3::OnOffApplication::PacketSize",
   StringValue (sizePacket));
Config::SetDefault ("ns3::OnOffApplication::DataRate",
   StringValue (AppPacketRate));

OnOffHelper clientHelper ("ns3::UdpSocketFactory", Address ());
 clientHelper.SetAttribute ("OnTime",  StringValue("ns3::
    ConstantRandomVariable[Constant=3]"));
```

```
clientHelper.SetAttribute ("OffTime",  StringValue("ns3::
    ConstantRandomVariable[Constant=5]"));
```

Estes parâmetros de tempo definem o tempo que a aplicação está a gerar tráfego de dados e o tempo em que a aplicação está parada sem gerar qualquer tráfego de dados entre os nós (fonte e destino). Este tráfego é caracterizado pela taxa de transmissão estabelecida e pelo tamanho dos pacotes. Contudo, configuramos a aplicação para enviar pacotes de dados com o tamanho de 372bytes a uma taxa de transmissão de 12.2Kbps, conforme [29]. Continuando, a aplicação foi configurada de forma a gerar tráfego durante 3.0 segundos e estar os seguintes 5.0 segundos sem gerar tráfego de dados na rede. Para além disso, definiu-se que a aplicação apenas é iniciada aos 15.0 segundos do tempo de simulação, dando este intervalo de espera (0 seg – 15.0 seg) para que os nós da rede troquem informação de controlo entre si de forma a criarem uma visão da topologia de rede, através do atributo "StartTime".

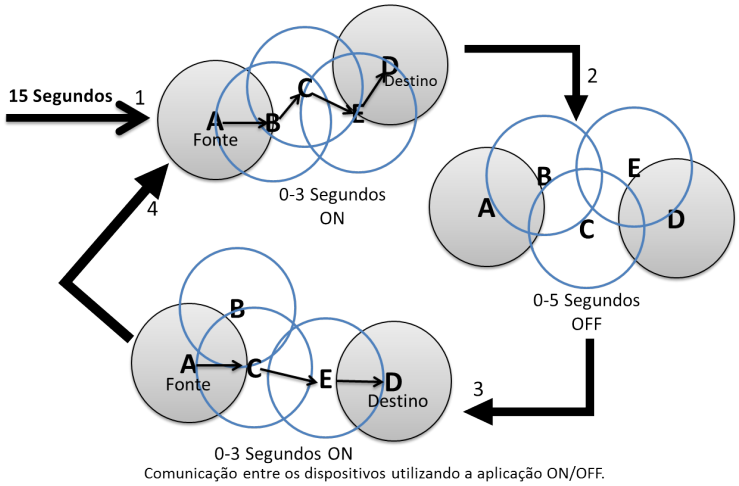


Figura 4.6: Representação do funcionamento da aplicação ON/OFF.

A aplicação termina o seu funcionamento 15 segundos antes do final do tempo de simulação, indicado pelo atributo "StopTime"da classe "ns3::OnOffApplication Class".

```
AddressValue remoteAddress (InetSocketAddress (Ipv4Address (
    no_dest), 80));
 clientHelper.SetAttribute ("Remote", remoteAddress);
```

```
clientApps.Add (clientHelper.Install (nodes.Get (no_source)));

clientApps.Start (Seconds (15.0));
clientApps.Stop (Seconds (timeSimulation-15.0));
```

De notar que quando a aplicação é iniciada, a primeira transmissão de pacotes de dados apenas ocorre, depois de no mínimo um atraso de tamanho_pacotes/bit_rate, não iniciando no tempo "StartTime"concretamente, por exemplo:
tempo de inicialização da aplicação = 15.0 seg,
tamanho pacote = 1000bytes = 8000bits,
taxa de bits=500Kbps=500000bits/s,
então terá um atraso de (15.0 + 8000/500000)=15.016 logo terá um atraso de 0.016 segundos. É compreensível que respetivamente com o algoritmos utilizados por cada um dos protocolos, estes iniciem a transmissão com maior atraso, devido por exemplo à construção e procura de rotas para atingir o destino.

Finalizando toda a descrição da construção e funcionamento da rede em geral apenas nos falta definir o número de simulações que irão ser efetuadas para cada grupo de números de nós, sendo estes grupos definidos por grupos de 20-40-60-80-100 e 120 nós. Desta forma para cada grupo de nós foram efetuadas 100 simulações para cada caso, combinando um dos protocolos com um tipo de mobilidade e uma das aplicações. No final de cada simulação é obtida uma coleção de dados resultantes dessa simulação. Estes dados são guardados constantemente em um "array"tridimensional, figura 4.7, diferenciando cada posição do "array"os elementos que avaliam o desempenho dos protocolos (descritos na secção seguinte), o número que representa cada simulação e o número de nós. No final da execução de todas as simulações para cada grupo de nós, é lido cada campo correspondente desse "array"e calculados valores como médias e desvio padrões de cada parâmetro. Respetivamente ao número de simulações efetuadas para cada grupo de nós, este valor foi acordado de forma a obtermos uma estimativa coerente, visto que a partir de um valor de simulações o intervalo de confiança se tornaria quase constante e não ocorreriam grandes variações nos valores finais, por isso a escolha de executar 100 simulações para cada grupo de nós na rede. Apesar que à medida que se

aumente o número de simulações o intervalo de confiança se torna mais específico obtendo valores de médias e desvio padrão mais linear ao longo do tempo.

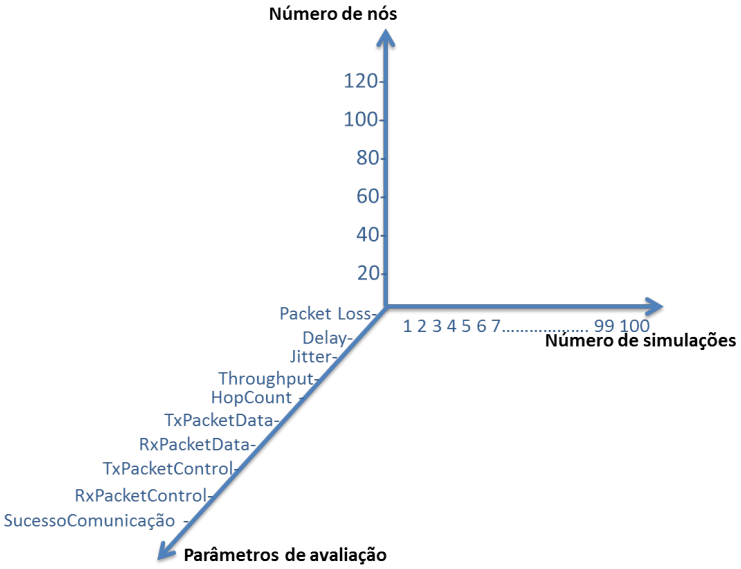


Figura 4.7: Array tridimensional onde se guardam os resultados correspondentes a cada simulação.

PacketLoss - soma o número de pacotes que são perdidos, isto é, os pacotes que foram transmitidos por uma fonte, mas que não foram recebidos no destino ou retransmitidos por um longo período de tempo, por defeito é 10 segundos;
Delay - Contém a média da soma dos atrasos de ponta a ponta para todos os pacotes recebidos;
Jitter - contém a média da soma da variação dos atrasos de ponta a ponta para todos os pacotes recebidos, jitter de um pacote é a variação do atraso(delay) relativamente ao ultimo pacote recebido $Jitter(P_n) = [Delay(P_n) - Delay(P_{n-1})]$;[23]
Throughput - representa a taxa de transferência desde a fonte para um destino($Mbits$);
HopCount - representa o número de salto que o pacote percorre desde a fonte até chegar ao destino, isto é a distância que a fonte está do destino;

TxPacketsData - é o número total de pacotes de dados transmitidos pela fonte para o destino(definido pelo fluxo);
RxPacketsData - é o numero total de pacotes de dados recebido pelo destino, enviados pela fonte(mesmo fluxo que o TxPacketData);
TxPacketsControl - representa o número de pacotes de controlo enviados de um nó para os restantes nós da rede;
RxPacketsControl - representa o número de pacotes de controlo recebidos por um nó a partir dos restantes nós da rede;
SucessoComunicação - este parâmetro consiste na identificação das simulações que obtiveram o valor de pacotes recebidos acima de 50% dos pacotes enviados.

Num ponto final são criados ficheiros de relatórios com todos os valores obtidos e gráficos com exposição destes valores obtidos de modo a ser possível fazer uma análise comparativa do desempenho do funcionamento de cada um dos protocolos com os seus modelos de mobilidades e a sua aplicação a funcionar.

4.4.4 Definição e processamento das métricas

Nesta subsecção pretendemos identificar os passos utilizados e a ferramenta utilizada para a obtenção dos valores finais. Assim será descrita a ferramenta utilizada, os cálculos utilizados e as métricas avaliadas até a obtenção dos resultados finais.

FlowMonitor

Em ambientes de simulação de redes, a monitorização é utilizada principalmente para caracterizar o desempenho da rede, em particular o desempenho dos protocolos Ad Hoc.

Com a execução de várias simulações no simulador NS-3, torna-se necessário obter resultados e avaliar os referidos modelos simulados com cada um dos protocolos em execução. Para isso, existe a necessidade de fazer a medição de um conjunto de métricas conhecidas, no que refere à avaliação de desempenho da rede, e valores importantes na área das redes de comunicação.

A monitorização de redes em ambientes de simulação pode por vezes apresentar alguns problemas ao longo da sua avaliação, devido aos modelos de simulação nem sempre serem projetados precisamente para produzir resultados para avaliação, o que faz com que por vezes produza resultados que podem divergir dos resultados obtidos pelo funcionamento real dos protocolos.[18]

Existem duas estratégicas básicas no NS-3 para gerar informação de estudo: usar mecanismos pré-definidos de saída e processar o conteúdo para extrair informações relevantes, através do processamento de *scripts* que utilizam *awk*, *grep* ou *sed*, ou então, desenvolver/configurar mecanismos de saída que resultem somente na extração da informação pretendida.

No simulador NS-3 existem uma serie de contribuições que geram ou utilizam ficheiros de rastreio que reúnem informações estatísticas da simulação. O Trace graph[38] é um analisador do simulador NS-2 com base no MatLab que processa quase todo o tipo de ficheiros de rastreamento do NS-2, produzindo simultaneamente uma variedade de gráficos que oferecem e facilitam uma visão geral do desempenho da rede, através do cálculo de métricas como atraso, *jitter* e pacotes perdidos (*packetloss*). Em particular, para o NS-3 apenas temos conhecimento das ferramentas TraceMetrics[7][7] e do módulo FlowMonitor. Inicialmente, foram desenvolvidas três scripts em linguagem AWK para analisar cada um dos ficheiros trace (*.tr*), produzidos pelas nossas simulações, para cada um dos protocolos. Mas ao longo que íamos aumentando a complexidade da rede e o número de métricas que queríamos analisar, obtínhamos ficheiros trace muito complexos e gigantescos, com muita informação irrelevante, tornando assim a análise deste modo pouco funcional de se utilizar para analisar os resultados obtidos.

Flowmonitor

- Vantagens

[7] É um analisador de ficheiros de rastreamento produzidos pela simulação. Esta ferramenta faz uma análise dos ficheiros obtidos (*.pcap*, *.trace* e *.xml*), que seriam muito complicados de analisar manualmente, e calculam métricas úteis para medição do desempenho da rede.

1. mantém variáveis de soma;
2. realiza uma análise rápida;
3. obtém dados de memória;
4. fácil de ser implementado na simulação;
5. *overhead* de processamento é pequeno;
6. necessita de pouca alteração na *script* de implementação da simulação para funcionar;

- Desvantagens

1. Tem como principal desvantagem não fornecer suporte para o calculo de medidas que precisem do conhecimento de todos os pacotes armazenados, como por exemplo, a variação do atraso.

TraceMetrics

- Vantagens

1. Flexibilidade, pois com este método tem o registo de tudo o que aconteceu na simulação, pacote a pacote, podendo calcular a medida que se deseje;
2. Apenas necessita de uma pequena alteração no código fonte da simulação, apenas para gerar o ficheiro trace;

- Desvantagens

1. Tem como principal desvantagem o desempenho, pois necessita de acessos a disco e a armazenamento de informações em memória.

Com isto obtém-se um *tradeoff* entre desempenho e a quantidade de métricas comparativamente com estes dois métodos de análise. (Figura 4.8)

Assim, para a coleta de dados ao longo das simulações e de forma a realizarmos os cálculos de análise no final de cada simulação, utilizamos o módulo *FlowMonitor* implementado no NS-3. Este módulo de monitorização de fluxo consiste num coletor de dados que ao longo da simulação vai observando os fluxos de dados

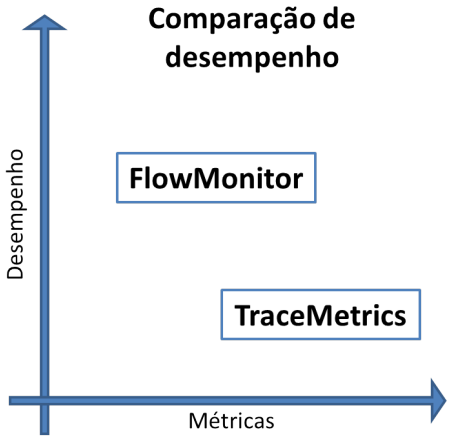


Figura 4.8: Comparação entre métodos de análise de acordo com o nível de desempenho e o número de métricas

que percorrem a rede. É utilizada a estrutura "FlowStats"da classe "FlowMonitor"(ns3::FlowMonitor::FlowStats) que é responsável por recolher dados estatísticos através da utilização de sondas em cada link de ligação, fazendo com que se escute e recolha dados ponto a ponto na rede.

Através desta estrutura conseguimos obter as medidas de um fluxo de pacotes individual. Este módulo é instanciado na simulação e são manipulados uma série de variáveis de soma, através das quais, no final da simulação, podemos utiliza-las e a partir delas efetuar cálculos e obter os valores dos parâmetros desejados. As métricas resultantes que são calculadas no final de cada simulação são:

- número de pacotes perdidos na comunicação($PacketLoss$),
- atraso médio($DelaySum$),
- *jitter* médio($JitterSum$),
- *throughput* médio($Troughput(Mbps)$),
- número médio de saltos($HopCount$),
- número de pacotes de dados transmitidos($TxPacketsData$),
- número de pacotes de dados recebidos($RxPacketsData$),
- número de pacotes de controlo transmitidos($TxPacketsControl$),
- número de pacotes de controlo recebidos($RxPacketsControl$),

- sucesso da comunicação(Comunicação % Sucesso),
- percentagem de pacotes de dados recebidos(% *RxPackets*)

Algumas destas métricas são derivadas a partir de alguns atributos que nos são disponíveis por outros parâmetros e que facilitam o cálculo das mesmas. Sendo assim, estas métricas são obtidas pelo uso dos seguintes parâmetros:

TxPacketsData = st.txPackets;

RxPacketsData = st.rxPackets;

TxPacketsControl = st.txPackets;

RxPacketsControl = st.rxPackets;

lossLido = st.lostPackets;

delayLido = (st.delaySum.GetSeconds() / st.rxPackets);

jitterLido = (st.jitterSum.GetSeconds() / (st.rxPackets-1));

throughputLido = (st.rxBytes * 8.0 / (st.timeLastRxPacket.GetSeconds()-st.timeFirstTxPacket.
/ 1024 / 1024);

hopLido = (st.timesForwarded / st.rxPackets + 1);

```
double valor = (RxPacketsData * 100)/TxPacketsData;
                if(valor>=50){
                      sucesso ++;
                 } else insucesso ++;
                perc_RxPackets = valor;
```

Em que a estrutura *st* representa as métricas da medição de um fluxo de pacotes individuais entre um nó (fonte) até outro nó (destino).

O módulo "FlowMonitor"está organizado em três grupos de classes, como mostra a figura (4.9). O grupo de classes do "core"é constituidor pelas classes "FlowMonitorFlowProbe"e "FlowClassifier". A classe "FlowMonitor"é responsável pela coordenação das sondas e recolha das estatísticas ponto-a-ponto. A classe "FlowProbe"é responsável pela escuta de eventos em pontos específicos da simulação, fazendo o relato desses eventos ao "FlowMonitor"e recolhe as suas próprias estatísticas dos fluxos referentes, apenas, aos pacotes que passam por essa sonda. E finalmente o "FlowClassifier"é responsável pela classificação e diferenciação dos

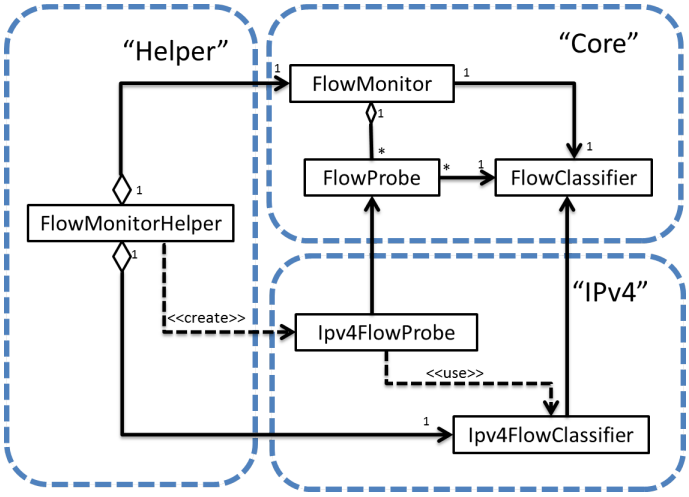


Figura 4.9: Arquitetura do "FlowMonitor"[18]

pacotes consoante o seu fluxo e tipo de pacote, através dos parâmetros "flow identifier"e "packet identifier"é identificado o tipo de fluxo e o tipo de pacote desse fluxo, para toda a simulação e independentemente do ponto onde é capturado esse pacote. Genericamente, o principal resultado deste processo de monitorização de fluxo é a recolha de dados estatísticos, sendo estes mantidos em estruturas de dados em memória. A figura 4.10, mostra-nos as duas estruturas de dados para distintos fluxos, "FlowMonitor::FlowStats"e "FlowProbe::FlowStats", em que o primeiro contém as estatísticas do fluxo ponto-a-ponto enquanto a segunda contém apenas um pequeno subconjunto de estatísticas do ponto de vista de cada sonda ("Probe").

Um descrição mais detalhada de cada um dos atributos do fluxo de dados, obtidos através da estrutura "FlowMonitor :: FlowStats", é feita da seguinte forma:

timeFirstTxPacket , contém o tempo absoluto, quando o primeiro pacote de um fluxo foi transmitido, isto é, o tempo quando a transmissão de fluxo começa;

timeLastTxPacket contém o tempo absoluto, quando o último pacote de um fluxo foi transmitido, isto é, o tempo quando a transmissão de fluxo termina;

timeFirstRxPacket contém o tempo absoluto, quando o primeiro pacote de um fluxo foi recebido pelo nó destino, isto é, o momento em que se inicia a recepção de pacotes;

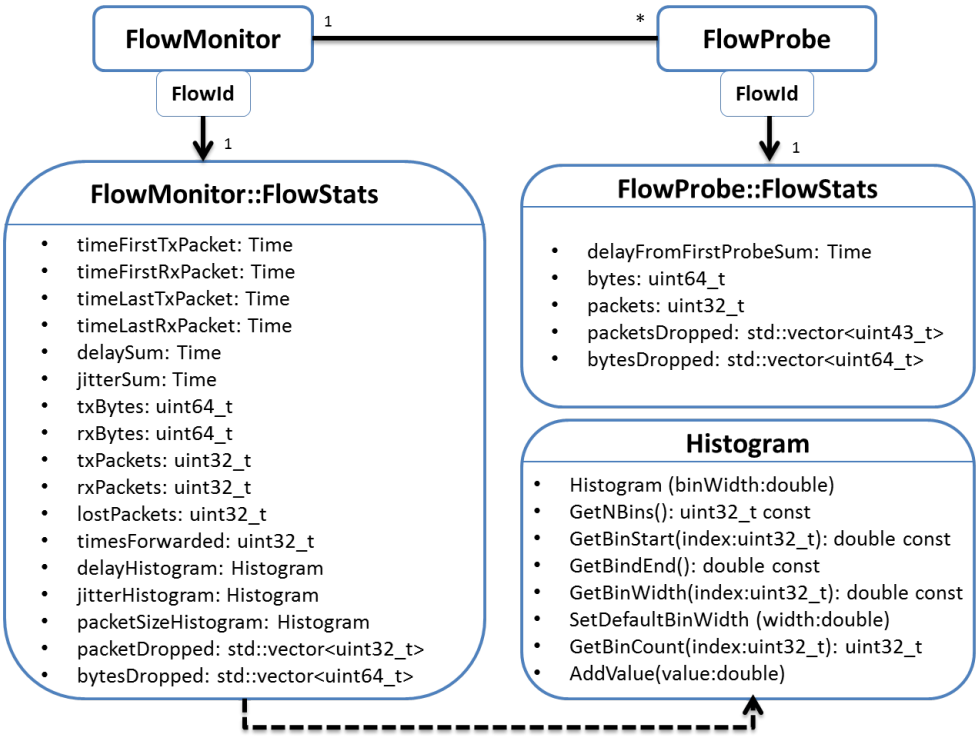


Figura 4.10: Método de coleta de dados do "FlowMonitor"

timeLastRxPacket contém o tempo absoluto, quando o último pacote de um fluxo foi recebido pelo nó destino, isto é, o momento em que a receção de pacotes termina;

delaySum contém a soma de todos os atrasos ponto-a-ponto de todos os pacotes recebidos por esse fluxo;

jitterSum contém a soma ponto-a-ponto da variação do atraso(delay jitter) de todos os pacotes recebidos por esse fluxo, isto é, uma variação do atraso na entrega dos pacotes desse fluxo. Neste caso, o jitter é definido como a variação do atraso relativamente ao último pacote dessa stream de dados, pela formula $Jitter(P_n) = [Delay(P_n) - Delay(P_{n-1})]$;

txBytes, txPackets contém o número total de bytes e pacotes transmitidos, respectivamente, para esse fluxo;

rxBytes, rxPackets contém o número total de bytes e pacotes recebidos, respectivamente, para esse fluxo;

lostPackets contém o número total de pacotes que são assumidos como perdidos, isto é, aqueles pacotes que foram transmitidos mas que por um período de tempo ainda não tenham sido recebidos. Por defeito, um pacote em falta por um período superior a 10 segundos é considerado perdido, embora este intervalo possa ser configurado;

timesForwarded contém o número de vezes que um pacote tenha sido encaminhado, assumindo para todos os pacotes desse fluxo;

delayHistogram, jitterHistogram, packetSizeHistogram correspondem a versões de histogramas para delay, jitter e tamanho de pacotes respectivamente;

packetsDropped, bytesDropped este campo contém o número de pacotes e bytes descartados devido a uma razão, do tipo, não encontra uma rota para encaminhar o pacote para esse destino, o pacote é descartado devido ao campo de TTL chegar a zero, então o seu tempo de vida na rede termina e por último devido a ocorrência de erro no cabeçalho ou checksum, sendo descartado;

A partir destes atributos são derivadas algumas das métricas importantes para a análise feita no final das simulações, através das seguintes formulas:

Média do valor de atraso, *delay*:

$$\overline{delay} = \frac{delaySum}{rxPackets}$$

Média do valor de *jitter*:

$$\overline{jitter} = \frac{jitterSum}{rxPackets-1}$$

Média do tamanho dos pacotes transmitidos(bytes):

$$\overline{S_{tx}} = \frac{txBytes}{txPackets}$$

Média do tamanho dos pacotes recebidos(bytes):

$$\overline{S_{rx}} = \frac{rxBytes}{rxPackets}$$

Média da taxa de transferência transmitida, *bitrate* (bit/s):

$$\overline{B_{tx}} = \frac{8 \times txBytes}{timeLastTxPacket - timeFirstTxPacket}$$

Média da taxa de transferência recebida, *bitrate* (bit/s):

$$\overline{B_{rx}} = \frac{8 \times rxBytes}{timeLastRxPacket - timeFirstRxPacket}$$

Média do número de saltos, *hop count*:

$$\overline{hopcount} = 1 + \frac{timesForwarded}{rxPackets}$$

Taxa de perda de pacotes, *PacketLossRatio*:

$$q = \frac{lostPackets}{rxPackets + lostPackets}$$

Desta forma implementamos na *script* um método para o cálculo de estatísticas do fluxo que pretendemos analisar, isto é, entre o nó fonte e o nó destino. Para tal desenvolveu-se o seguinte código:

```
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->
    GetFlowStats ();
 monitor->CheckForLostPackets ();
 for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i =
     stats.begin (); i != stats.end (); ++i){
  Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first
     );
  uint16_t destPort = 80;
  if(t.sourceAddress == addr1 && t.destinationAddress == addr && t
     .destinationPort== destPort){
   std::cout << Simulator::Now ().GetSeconds ()<<std::endl;
   std::cout << "->Flow_ID " << i->first <<  "(" << protocolName
       << " "<< " (" << t.sourceAddress << "/" << t.sourcePort<< "
       -> " << t.destinationAddress << "/" << t.destinationPort <<
       ")"<<std::endl;
    typePacket = "Data";
    print_stats(i->second, protocolName, typePacket);
  } else
   if(t.sourceAddress == addr1 && t.destinationAddress == group){
    typePacket = "Control";
```

```
    PrintDataControl(i->second, protocolName, typePacket);
   }
 else{}
 }
```

Em que fazemos a escuta do fluxo de dados entre o nó fonte e o nó destino através da passagem dos seus endereços IP, verificando sempre quando um pacote é enviado para o endereço destino ou quando o pacote é enviado em *broadcast*, isto é, diferenciando os pacotes de dados com os pacotes de controlo.

4.4.5 Simulação

Com a *script* desenvolvida e utilizando o simulador NS-3 versão 3.16, partimos para a simulação propriamente dita, correndo esta *script* utilizando as funcionalidades do *cluster search.di.uminho.pt* do departamento DI da Universidade do Minho. Assim a partir deste *cluster* foram utilizados 8 CPU's *quad core* devido à complexidade e ao processamento necessário que teríamos que obter para executar este processo em uma máquina pessoal, o que levaria muito tempo cada simulação. Para isso foram então desenvolvidas duas *scrips bash* ("job-template.sh"e "job-prepare.sh") de forma a colocar cada *job* neste *cluster* de forma a fazermos todas as simulações com cada combinação disponível, alternando o protocolo, o numero de nós, o modelo de mobilidade e a aplicação para cada simulação.

A primeira *script* consiste em definir os parâmetros que serão utilizados no *cluster*, sendo implementada da seguinte forma.

Listing 4.1: bash version

```
#!/bin/sh
### Job settings
#PBS -N AdHoc_End
#PBS -o AdHoc_End.out
#PBS -j oe
#PBS -M a50045(at)alunos.uminho.pt
#PBS -m abe
```

```
### Job resources HELP
#PBS -l nodes=1:ppn=8:quad
#PBS -l walltime=20:00:00

source /etc/profile.d/env-modules.sh
### module list

cd $PBS_O_WORKDIR
### ... SIMUL="/home/acosta/ns-allinone-3.16/ns-3.16/build/scratch
   /AdHoc_End"
### ... time $SIMUL --mobility=1 --application=3 --protocol=2 --
   nWifi=300
### ... echo "All done."
```

onde é definido o nome do *job*, o nome do ficheiro de *output*, o mail para o qual deve ser enviadas as notificações e a definição das notificações que se pretende receber. De seguida nas linhas "#PBS -l nodes=1:ppn=8:quad, #PBS -l walltime=20:00:00"define-se o número de cpu's que se pretende disponibilizar e os core's que compõe esse cpu para cada nó, seguido do tempo máximo que se pretende dar a esse *job*.

A segunda *script* (job-prepare.sh) consiste em preparar um *job*, identificando o url onde temos a nossa *script* de simulação da rede Ad Hoc, constituída ainda por todas as combinações que deverão ser executadas, descrita pelo seguinte código:

Listing 4.2: bash version

```
SIMUL="/home/ns-allinone-3.16/ns-3.16/build/scratch/AdHoc_End"
### User Arguments:
###  --protocol: 1=AODV; 2=OLSR; 3=DSDV
###  --nWifi: Number of wifi STA devices
###  --application: 2=SendPackets; 3=ON/OFF SendPackets
###  --mobility: Tipe of mobility model the nodes. 1, 2 or 3
###  --numSimulation: The number of simulations

NOS="50 100 150 200 250 300"
PROTOS="1 2 3"
```

```
PNAMES="AODV OLSR DSDV"
MOB="1 2"
MOBNAME="RandomWayPoint Gauss Walk3D"
APPS="2 3"
APPMAMES="CBR ONOFF"
sims=100

i=0
for n in $NOS; do
 for m in $MOB; do
  for a in $APPS; do
   for p in $PROTOS; do
    ## Ficheiro da script e respetivo output
    INFILE=aend-job-$n-$m-$a-$p.sh
    OUTFILE=aend-job-$n-$m-$a-$p.out
    ## inicia com o template e acrescenta-lhe o comando..
    cat $TEMPLATE > $INFILE
    echo "time $SIMUL --mobility=$m --application=$a --protocol=$p
        --nWifi=$n --numSimulation=$sims" >> $INFILE
    echo "Executing: qsub -N aend$i -o $OUTFILE $INFILE"
    qsub -N aend$i -o $OUTFILE $INFILE
    let i=i+1
   done
  done
 done
done
echo "A total of $i jobs where submited to the cluster..."
```

aqui é indicado o url onde está a *script* de simulação da rede Ad Hoc descrita nas secções anteriores, indicando os grupos de nós que serão utilizados em cada simulação, os protocolos, os modelos de mobilidade, as aplicações e o número de simulações que deverá executar para cada combinação, que neste caso tem o valor de 100. Contudo, para cada umas das combinações com as propriedades descritas executa o *qsub* que submete o *job* ao *cluster*.

4.5 Sumário

Neste capitulo fez-se a exposição de todo o processo de implementação e desenvolvimento, explorando as ferramentas utilizadas e os métodos para a simulação de encaminhamento de tráfego entre os nós em uma rede Ad Hoc. É feita assim a apresentação e descrição de toda a *script* desenvolvida e de todos os parâmetros que são avaliados, seguindo deste modo o caminho para atingir o objetivo principal desta dissertação. É apresentada nesta parte final a aplicação da *script* em simulação em um *cluster* e os parâmetros e configurações necessárias para a execução das simulações.

5

Resultados obtidos

Este capítulo apresenta uma descrição de todos os resultados obtidos de acordo com as simulações efetuadas. É aqui feita a apresentação e uma análise detalhada de todos os resultados obtidos das simulações descritas no capitulo anterior, através da representação gráfica do desempenho de cada protocolo para cada ambiente em que foi simulado, estabelecendo assim a sua comparação.

5.1 Cálculo do Intervalo de Confiança

Para a apresentação dos valores obtidos, das métrica avaliadas, e de forma a obtermos valores coerentes relativamente a um grupo de simulações efetuadas para cada cenário, utilizamos o cálculo de intervalos de confiança através de médias e desvios padrões obtidos. Os intervalos de confiança são normalmente utilizados na análise de resultados de forma a indicar a confiabilidade de uma estimativa.

Estes intervalos têm como base uma população, número de amostras, através do qual obtemos a média de valores que determinam uma estimativa afirmativa de confiança sobre os resultados obtidos e possivelmente corretos, obtendo um valor mais aproximado do valor real, ou seja, à medida que a amostragem é repetida inúmeras vezes e o intervalo de confiança é recalculado para cada amostra de acordo com o mesmo método, o parâmetro estatístico calculado está contido na proporção p dos intervalos de confiança.

No nosso caso, ao fazer mais de 30 simulações para cada cenário, aumentamos o nível de confiança dos resultados obtidos. Este nível de confiança, é normalmente descrito como sendo a probabilidade de que um intervalo estimado contenha o valor ou parâmetro populacional. Da teoria da estatística, sabemos que quando o número de amostras é igual ou superior a 30, $x \geq 30$, a distribuição amostral das médias amostrais é considerada uma distribuição normal. Assim, o nível de confiança **c** é a área sob qual a curva normal padrão está entre as valores críticos $-Z_c$ e Z_c. Como toda a área representa um valor total, valor 1, e a área remanescente é 1-c, conclui-se que a área em cada uma das extremidades é $\frac{1}{2}$(1-c).

Por exemplo, no caso do número de simulações for 70, c=90% então os 5% da área que estão à esquerda de $-Z_c$=-1,645 e os 5% que estão à direita de Z_c=1,645 (veja a Figura 5.1 onde α=1-c).

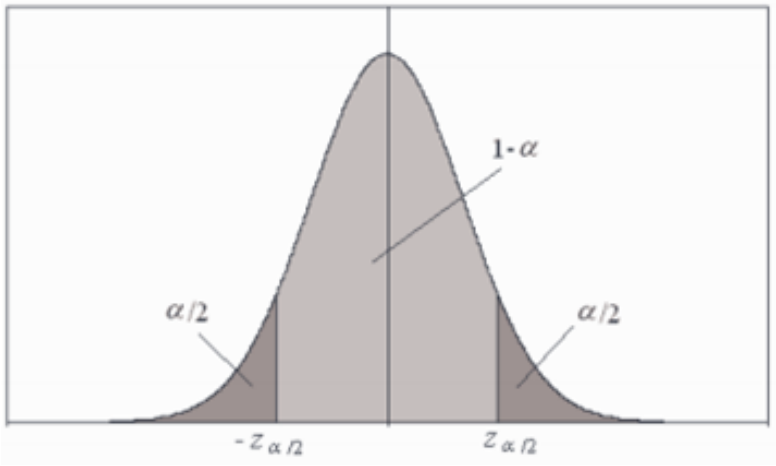


Figura 5.1: Intervalo de confiança

Em que α representa o nível de significância que determina o nível de confiança. O valor do nível de confiança é igual a $100\times(1 - \alpha)\%$, que consiste em dizer que, para um alfa de 0.05 indica um nível de confiança de 95%.

5.2 Análise dos resultados obtidos

Esta primeira série de resultados apresentam o resumo da realização de simulações que seguem as caraterísticas apresentadas na tabela 5.1, na qual utilizamos as duas aplicações como estratégica de comparação, com a utilização dos 3 protocolos de encaminhamento e utilizando o modelo de mobilidade 1 desenvolvido.

Aplicação 1 - SendPackets	Aplicação 2 – ON/OFF
Número de nós (grupos): 20-40-60-80-100-120	**Número de nós (grupos)**: 20-40-60-80-100-120
Número de simulações por cada grupo de nós: 100	**Número de simulações por cada grupo de nós**: 100
Tempo de simulação : 150 segundos	**Tempo de simulação** : 150 segundos
Número de nós fonte – destino: 2 (1 par)	**Número de nós fonte – destino**: 2 (1 par)
Modelo de mobilidade: RandomWaypointMobilityModel – Mobilidade 1	**Modelo de mobilidade**: RandomWaypointMobilityModel – Mobilidade 1
Protocolos de encaminhamento: AODV, OLSR e DSDV	**Protocolos de encaminhamento**: AODV, OLSR e DSDV
Tipo de canal: Wireless	**Tipo de canal**: Wireless
Tipo de tráfego: CBR, 360Kbps	**Tipo de tráfego**: ON_OFF (3s/5s), 12,2Kbps
Tamanho do pacote: 1500 bytes	**Tamanho do pacote**: 372 bytes
Intervalo entre pacotes: 0,033 segundos	**Intervalo entre pacotes**: 3 segundos a ON – 5 segundos a OFF

Tabela 5.1: Parâmetros de simulação 1

Através destas caraterísticas obtemos os seguintes resultados para cada uma das métricas estudadas de acordo com o protocolo utilizado e o número de nós (dispositivos) que compõe a estrutura de rede dentro do espaço de rede definido no modelo de mobilidade 1.

Através destes resultados podemos comparar os valores obtidos da métrica *PacketLoss* entre todos os protocolos AODV, OLSR e DSDV para cada grupo de nós tanto com a utilização da Aplicação 1 (SendPackets) como com a Aplicação 2 (ON/OFF). Em primeiro lugar, podemos diferenciar desde já a diferença do

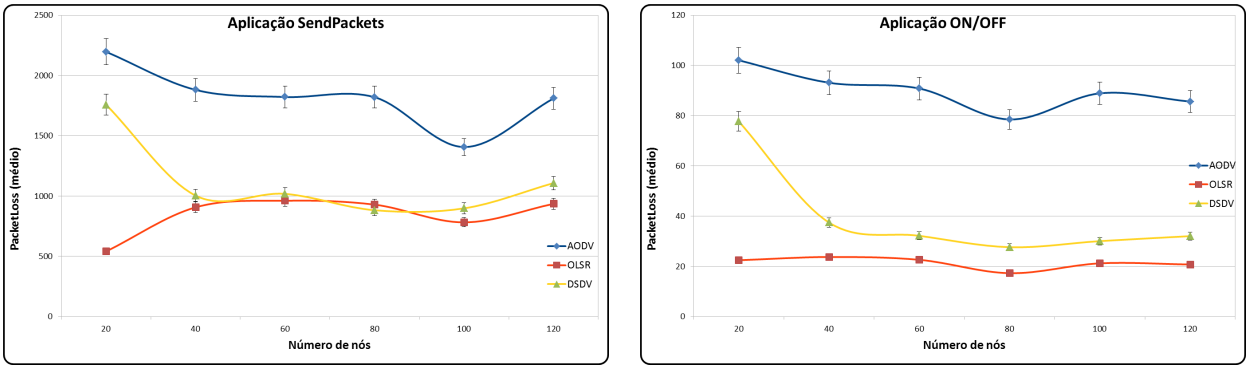


Figura 5.2: Valor médio de *PacketLoss* com mobilidade 1.

número de pacotes transmitidos entre as duas aplicações, sendo que com a aplicação SendPackets o intervalo entre pacotes é muito inferior (0.033 segundos) ao intervalo de envio de pacotes da aplicação ON/OFF, sendo que esta apenas envia pacotes durante 3 segundos de 5 em 5 segundos. De certa forma, verificamos que o protocolo AODV obtém valores superiores de pacotes perdidos, tanto com a utilização da aplicação SendPackets como com a aplicação ON/OFF, verificando que esta perda vai decrescendo à medida que existem mais dispositivos na rede. Isto pode entender-se, porque como o protocolo AODV só estabelece uma rota de encaminhamento apenas quando é necessária ou requisitada uma comunicação e não guarda tabelas de encaminhamento de toda a topologia de rede, este poderá obter mais perda de pacotes enquanto é estabelecida a ligação entre o nó fonte e o nó destino, sendo que a aplicação inicia sempre aos segundos 15 do tempo de simulação. Entretanto como existe ainda um número muito baixo de dispositivos na rede, podendo estes estarem bastante dispersos e não conseguirem estabelecer ligação, fica mais complicado com o uso deste protocolo (AODV) o estabelecimento da rota apenas quando necessita comunicar, podendo a topologia estar completamente diferente a cada vez que é necessária uma ligação.

Por outro lado, verifica-se que o protocolo DSDV tem inicialmente uma grande perda de pacotes, explicando-se este fenómeno pelo próprio funcionamento do protocolo DSDV, que inicialmente poderá não conter as tabelas de encaminhamento atualizadas e uma visão de toda a topologia devido a serem poucos dispositivos na rede e de poder não haver conectividade entre todos eles, fazendo com que ainda

possa faltar informação das suas atualizações das tabelas de encaminhamento. Como explicado no capitulo 3, com a utilização deste protocolo cada dispositivo contém uma tabela de encaminhamento com a lista de todos os caminhos possíveis. Por outro lado, como ainda são considerados um número reduzido de dispositivos numa área de $250000m^2$, era de prever que inicialmente tanto com a aplicação SendPackets como com a aplicação ON/OFF se verifica-se uma maior perda de pacotes. Isto deve-se relativamente à possibilidade de em alguns casos, os nós estarem muitos dispersos e o nível de cobertura entre os dispositivos não se verificar. Relativamente ao protocolo OLSR, que é dos protocolos mais indicados para situações de grande mobilidade dos dispositivos, verifica-se que este é o que obtém melhores resultados ao nível de perda de pacotes na comunicação, sendo que este protocolo guarda também uma ou mais tabelas de encaminhamento.

Um ponto imprevisto do protocolo OLSR é de inicialmente, sendo a rede composta ainda por poucos dispositivos dispersos numa grande área de cobertura, ele com a aplicação SendPackets obter um baixo valor de pacotes perdidos. Este protocolo tanto com a aplicação SendPackets como com a aplicação ON/OFF é o protocolo que obtém melhor resultado, explicado pelo uso dos nós MPR's que utiliza para transmitir e encaminhar o tráfego pela rede, fazendo com que os pacotes sejam encaminhados por menos nós e cheguem ao destino dentro do tempo de vida.

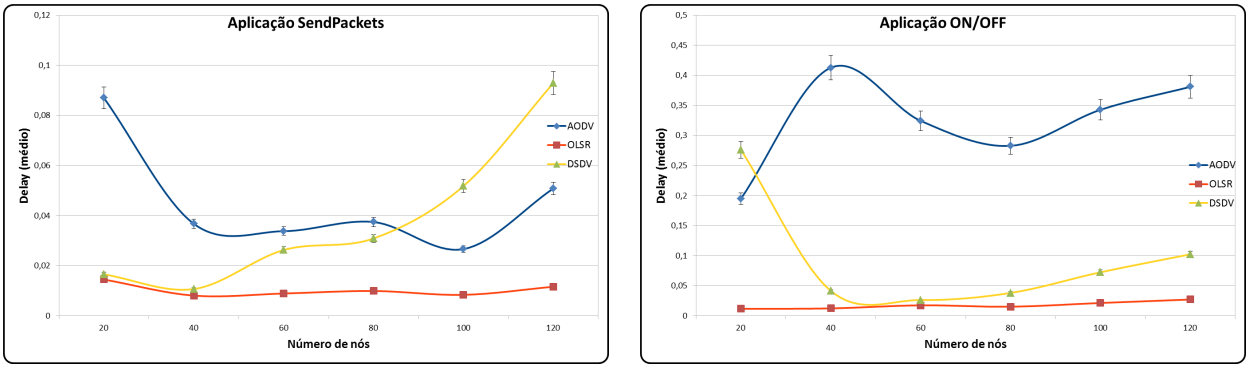


Figura 5.3: Valor médio de *Delay* com mobilidade 1.

Relativamente ao *delay* calculado entre as duas aplicações no mesmo modelo de mobilidade, verifica-se aqui diferenças entre as aplicações, isto porque a média da

soma dos atrasos de ponta a ponta para todos os pacotes recebidos se comporta de diferente forma. Isto verificando que com a aplicação SendPackets o AODV tende a melhorar, tendo inicialmente um registo elevado devido a, por exemplo, atrasos devido ainda existirem poucos nós e ser mais complicado a entrega de pacotes entre os diferentes dispositivos. Sendo que com a aplicação ON/OFF como o tráfego de dados é menor, e o intervalo entre pacotes é maior, a rede poderá já obter uma maior visão da rede e apesar de um aumento inicial do *delay*, este estabiliza. Por outro lado, verifica-se um subida na aplicação SendPackets do *delay* quando se utiliza o protocolo DSDV, isto explicado, porque à medida que aumenta o número de dispositivos, maior é a troca de informação entre os dispositivos, e assim com esta aplicação o intervalo entre pacotes é muito baixo, as rotas poderão ainda estar em atualização atrasando a entrega de pacotes. O que já não se verifica utilizando a aplicação ON/OFF, onde o protocolo OLSR e DSDV têm mais tempo para as trocas de informação e atualização das suas tabelas de encaminhamento baixando o *delay*. É importante referir, que para além destas diferenças, estes valores variam numa escala muito pequena e por isso se verifique uma maior diferença de atrasos entre a utilização de diferentes protocolos. Concluindo verifica-se que mesmo que o número de dispositivos aumente o protocolo OLSR mantém um *delay* sempre melhor, independente da aplicação utilizada, isto explicado por serem menos nós a transmitir e por cada um dos nós criarem inicialmente tabelas com a informação de toda a topologia e dos melhores caminhos, trocando periodicamente mensagens para verificar o estado de ligação da rede.

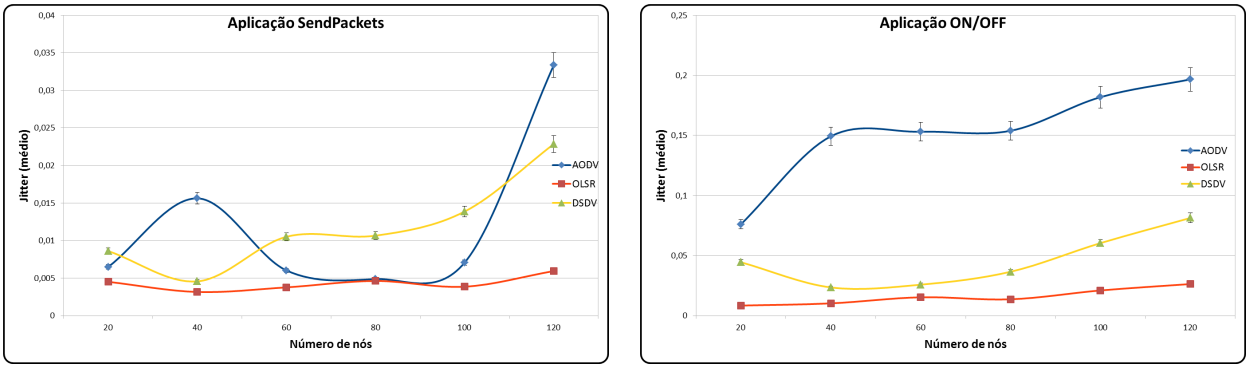


Figura 5.4: Valor médio de *Jitter* com mobilidade 1.

Na descrição dos valores de *jitter* obtidos, descrevendo a média da soma da variação dos atrasos de ponta a ponta para todos os pacotes recebidos, comparando com as figuras anteriores sobre os atrasos (*delay*), pode-se verificar uma subida dos valores de *jitter* à medida que aumenta o número de dispositivos dentro da mesma área, sendo que com a aplicação ON/OFF a subida é mais visível na utilização do protocolo AODV, podendo justificar-se como anteriormente, por ser um protocolo que apenas estabelece a rota apenas quando é necessário comunicar e não necessita de estar constantemente a trocar mensagens de controlo para atualização dos seus caminhos, obtendo um maior atraso na entrega de pacotes entre a fonte e o destino, isto devido a aplicação enviar menos pacotes e com maiores intervalos de tempo que a aplicação SendPackets. Contudo verifica-se utilizando a aplicação Sendpackets no intervalo de 60 aos 100 dispositivos uma estabilidade dos protocolos relativamente a esta métrica, sendo que a partir dos 100 dispositivos, com o protocolo AODV o valor de *jitter* sobe substancialmente aproximadamente 0.03 segundos. Esta subida poderá explicar-se devido a ser um número já elevado de dispositivos neste espaço de rede, fazendo com que à medida que o dispositivo fonte pretende enviar um pacote e o protocolo estabelece a rota até ao destino, obtenha-se um maior atraso, porque sempre que seja necessário enviar um pacote este protocolo faz uma procura da melhor rota. Mais uma vez verifica-se que o protocolo OLSR, tanto utilizando a aplicação SendPackets como a aplicação ON/OFF é o protocolo que obtém menos atrasos, aumentando ligeiramente à medida que aumenta o número de dispositivos, devido claro, a serem utilizados possivelmente mais dispositivos como encaminhadores.

É de notar que através destes gráficos se observem maiores discrepâncias com a diferença de protocolo utilizado, mas isto porque se utiliza um escala muito reduzida, de modo a se observar melhor as diferenças entre os resultados. Com a utilização de uma escala reduzida é possível verificar-se as principais diferenças, mas é importante dizer que entre uma escala com valores muito reduzidos como os que são aqui utilizados pouco diferenciam a utilização do protocolo no encaminhamento, verificando que são diferenças de 0.005 ou 0.05 segundos, respetivamente à aplicação utilizada.

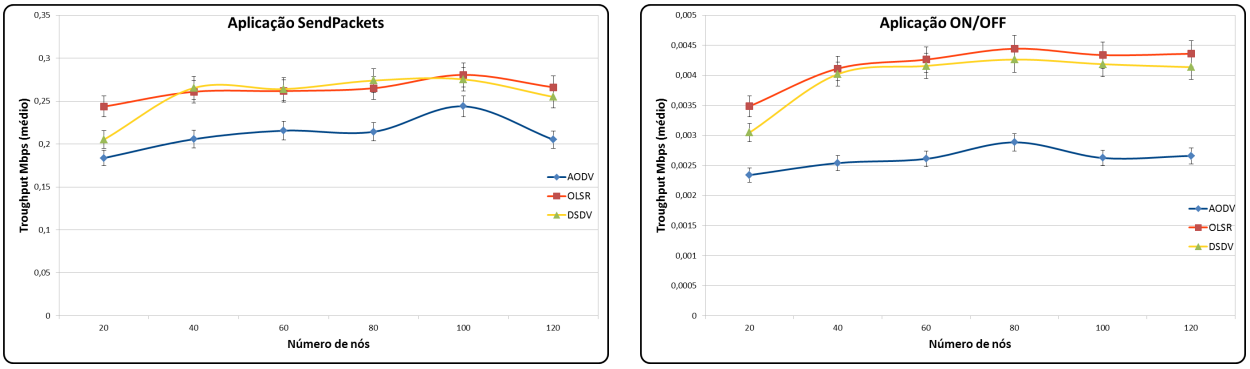


Figura 5.5: Valor médio de *Troughput* (Mbps) com mobilidade 1.

Relativamente ao *Troughput* entre as duas aplicações, é claramente notório que como a aplicação SendPackets se obtém um maior valor de débito, representando assim um maior valor de taxa de transferência desde a fonte para um destino. Neste caso, com a aplicação SendPackets verifica-se que os protocolos pro-ativos, que já obtêm um conhecimento da topologia de rede, resultam com melhores valores de *Troughput*, verificando-se do mesmo modo com a utilização da aplicação ON/OFF. Relativamente à utilização de cada protocolo, verifica-se que o protocolo OLSR para as duas aplicações obtém sempre valores mais estáveis. Este resultado deve-se a este protocolo utilizar os seus nós MPR's para encaminharem os pacotes, obtendo as rotas com menor custo, sendo menos nós a transmitir e utilizando um melhor caminho obtendo uma taxa de transferência superior, isto também como observado na figura 5.2, onde obtém um menor número de pacotes perdidos. Com a utilização da aplicação ON/OFF os protocolos pro-ativos obtêm um melhor *Troughput* porque já têm a informação sobre as melhores rotas até ao destino quando a aplicação inicia a transferência de dados. Verificando que com a utilização do protocolo AODV, o valor de *Troughput* é mais baixo por estar sempre entre intervalos de 5 em 5 segundos a estabelecer uma descoberta sobre a melhor rota, podendo perder-se mais dados entre a fonte e o destino, diminuindo a transferência de dados entre a fonte e o destino.

Mais uma vez, se identifica que para além destas diferenças, estes valores variam numa escala com uma variação muito baixa, observando-se de melhor forma algumas das diferenças entre a utilização dos diferentes protocolos.

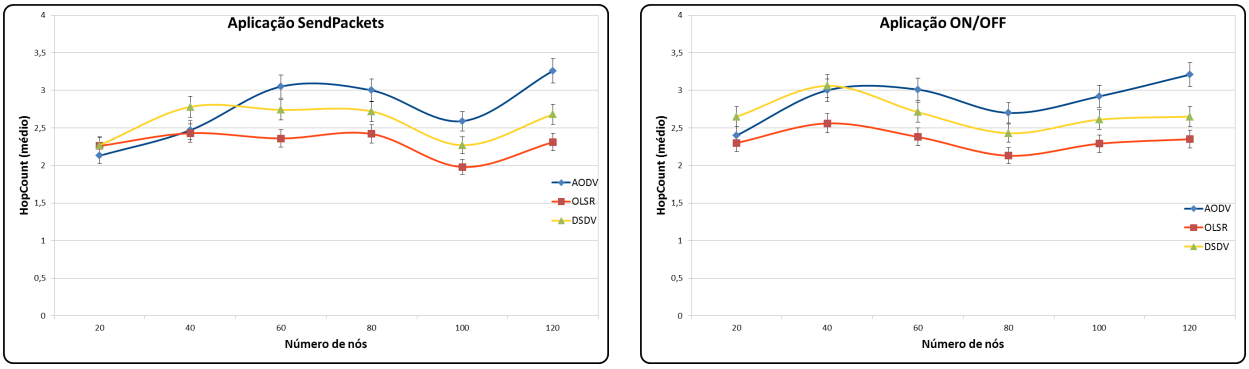


Figura 5.6: Número de Saltos com mobilidade 1.

De acordo com as os resultados obtidos das simulações, verifica-se que o número de saltos no encaminhamento entre a fonte e o destino, tanto utilizando a aplicação SendPackets como a aplicação ON/OFF, variam em média entre 2 a 3 saltos. Isto prevê-se devido à área de abrangência de cada dispositivo e do melhor caminho entre o dispositivo fonte e o dispositivo destino, visto que nem sempre o melhor caminho de encaminhamento é o caminho com menor número de saltos, isto é, dependendo do protocolo em utilização. Neste caso, verifica-se momentos em que o valor de número de saltos em qualquer um dos protocolos atinge um melhor valor, sendo que com a utilização da aplicação SendPackets, quando a rede é constituída por 100 dispositivos verifica-se o melhor cenário onde os três protocolos têm os melhores valores, relativamente ao número de nós que representam a rede. Enquanto que com a utilização da aplicação ON/OFF, o intervalo com 80 dispositivos observa-se o mesmo caso, subindo minimamente quando o número de dispositivos aumenta. Resumindo, verifica-se que o protocolo OLSR, utiliza sempre em média um menor número de saltos, isto presumivelmente devido à utilização dos seus encaminhadores MPR's. Por outro lado, depois de estudados os protocolos e o seu funcionamento já era previsível este funcionamento, sabendo que à medida que o número de nós aumenta, poderá haver mais caminhos desde a fonte até ao destino, mas nem sempre o melhor caminho seja o caminho com menos saltos, mas sim escolhe-se o caminho respeitando outras métricas para entrega de dados entre a fonte e o destino, utilizadas por cada um dos protocolos, já referidos na secção 4.3.

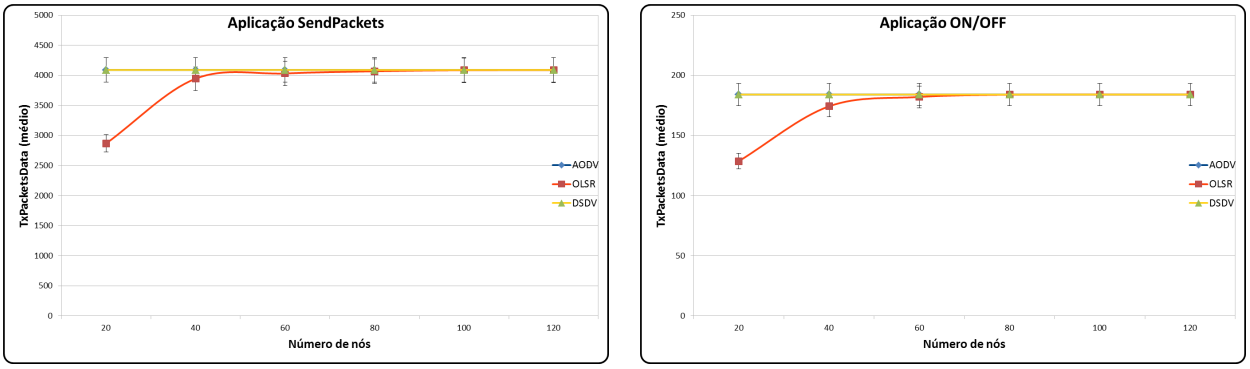


Figura 5.7: Número de pacotes de dados transmitidos com mobilidade 1.

Neste caso, tanto utilizando a aplicação SendPackets configurada para enviar no máximo do tempo de simulação 4090 pacotes e a aplicação ON/OFF enviar 184 pacotes, verifica-se que apenas a utilização do protocolo OLSR é o que inicialmente não consegue enviar a totalidade dos pacotes de dados, sendo isto representado devido a este protocolo inicialmente trocar informação para a escolha dos seus MPR's e depois calcularem as rotas com menor custo até ao destino. Com a utilização de apenas 20 dispositivos na área de rede da mobilidade 1, estes podem não serem todos atingíveis inicialmente, devido à sua disposição inicial e à sua mobilidade, sendo mais demorada a escolha dos nós MPR's e a troca de informação entre os diversos dispositivos, fazendo com que sejam enviados menos pacotes de dados dentro do tempo de simulação, desde que é iniciada a aplicação geradora de tráfego entre os dispositivos fonte e destino e o tempo que a rota é estabelecida.

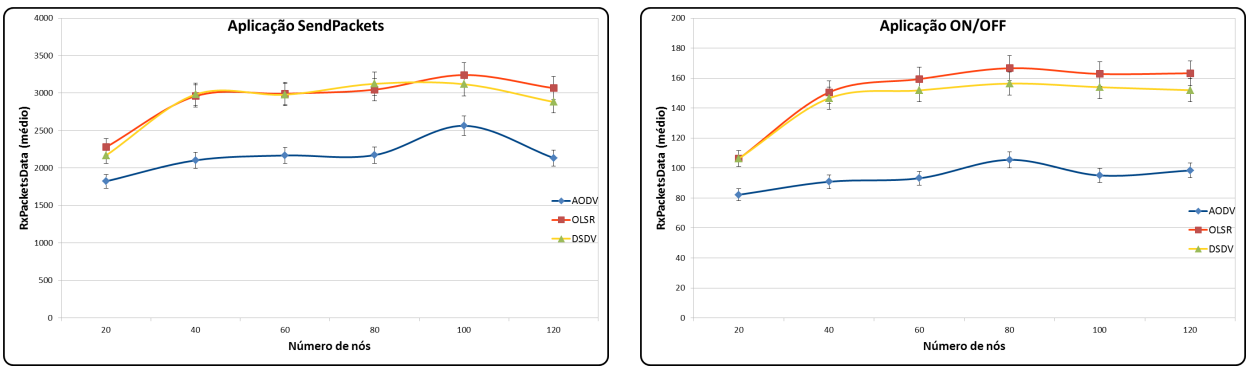


Figura 5.8: Número de pacotes de dados recebidos com mobilidade 1.

No que respeita aos pacotes recebidos, os protocolos pro-ativos têm uma perfor-

mance equivalente, sendo que inicialmente como já referido, quando a rede apenas contém 20 nós, é ainda difícil ter uma abrangência de toda a rede e que todos os pacotes enviados sejam recebidos pelo destino. Por outro lado, com mobilidade dos nós no espaço de rede faz com que se possam perder muitos pacotes de dados ao longo percurso em que estes são encaminhados, devido a falhas nos dispositivos ou mesmo falhas de ligações. Contudo, comparando as duas aplicações, verificamos que em média com o aumento do número de nós aumenta também o número de pacotes de dados entregues, devido ao serem encontrados mais caminhos entre a fonte e o destinos, apesar de que com a utilização do protocolo AODV, em média apenas metade dos pacotes transmitidos é que são entregues, podendo isto dever-se ao funcionamento do protocolo, sendo que este protocolo não necessita de guardar as rotas de encaminhamento e apenas estabelece a rota quando é necessário comunicar, podendo a aplicação de geração de tráfego já ter iniciado o envio de pacotes e ainda não estar estabelecida a rota entre a fonte e o destino.

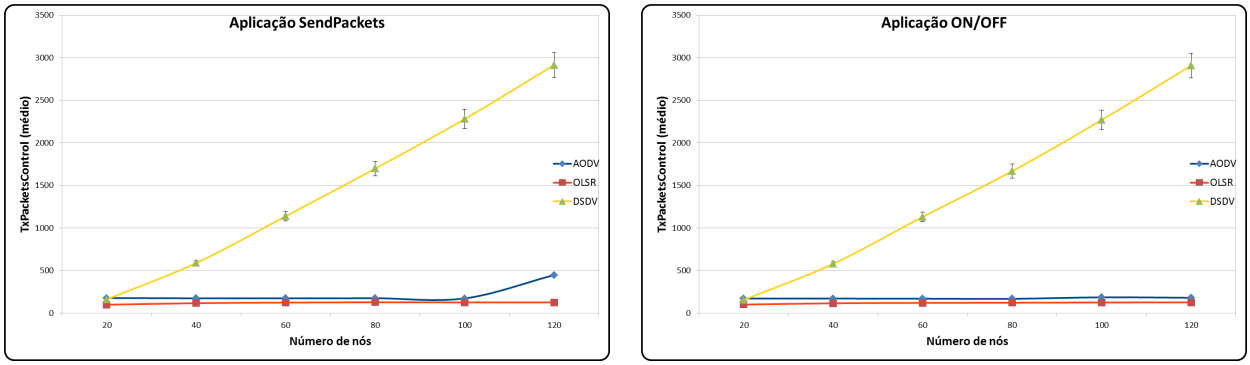


Figura 5.9: Número de pacotes de controlo transmitidos com mobilidade 1.

No que respeita aos pacotes de controlo, figura 5.9, trocados entre os dispositivos de rede, verifica-se que tanto com a aplicação SendPackets como com a aplicação ON/OFF apenas o protocolo DSDV obtêm um crescimento acentuado à medida que os número de dispositivos aumenta, isto porque com a utilização deste protocolo, os dispositivos trocam constantemente informação entre eles, e cada nó guarda uma tabela de encaminhamento com a lista de todos os nós da rede contendo os caminhos possíveis. Por outro lado, com a utilização deste protocolo, os dispositivos da rede, anunciam as suas informações para atualização das tabelas

de encaminhamento, mesmo quando não é necessária a transmissão de dados entre os dispositivos, obtendo uma maior *overhead*. Isto verifica-se no crescimento do valor da média dos pacotes transmitidos e recebidos (figura 5.10) à medida que aumenta o número de dispositivos na rede.

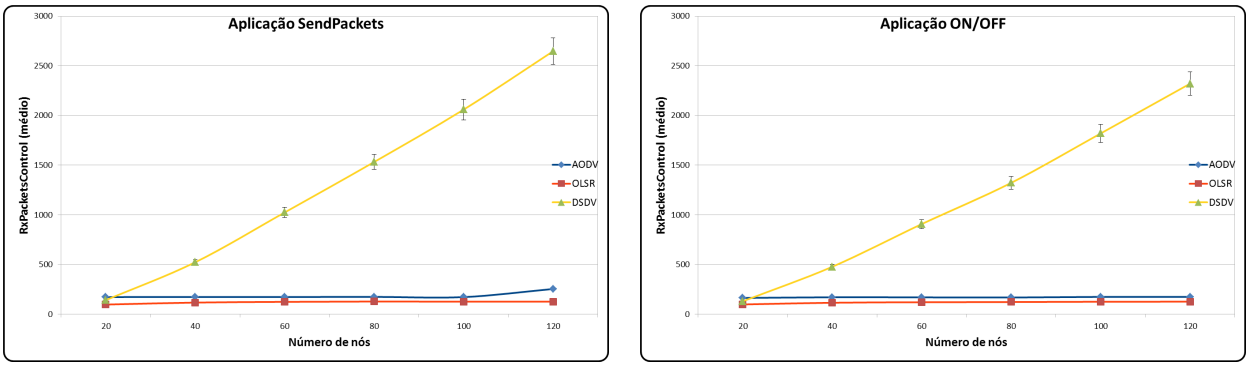


Figura 5.10: Número de pacotes de controlo recebidos com mobilidade 1.

Relativamente aos protocolos AODV e OLSR, com estes verificamos em média um valor constante na transmissão e receção de pacotes de controlo, visto que o AODV apenas troca informação de controlo quando necessita trocar dados entre os dispositivos e o protocolo OLSR apenas troca informação de controlo entre os dispositivos com o seu MPR associado, enviando pacotes de controlo apenas entre os seus dispositivos e os dispositivos designados MPR's para verificação dos estados de ligação entre os dispositivos, sendo menos dispositivos na rede a transmitir.

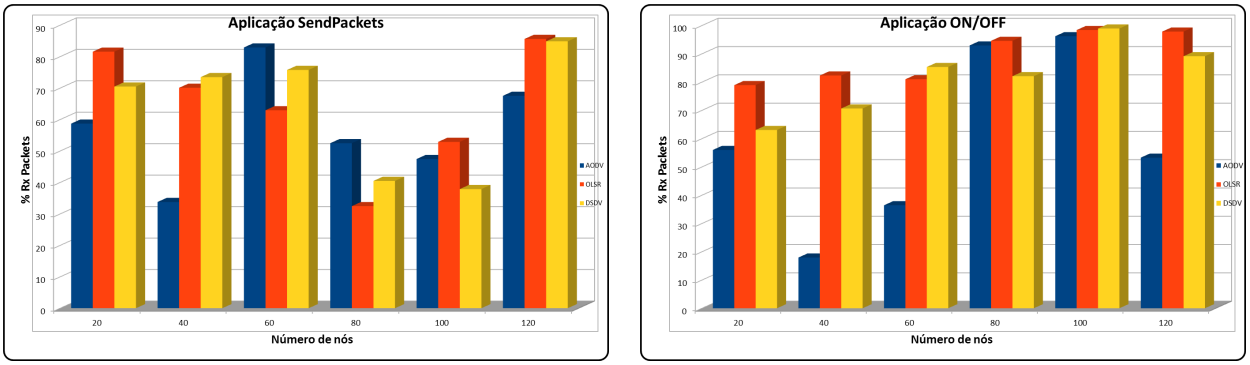


Figura 5.11: Percentagem de pacotes recebidos pelo nós destino com mobilidade 1.

Para finalizar mostramos as seguintes figuras 5.11 e 5.12, onde mostramos a

comparação entre a aplicação SendPackets e a aplicação ON/OFF correspondente à percentagem de pacotes recebidos e á percentagem de sucesso na comunicação, sendo que apenas é aceite como sucesso quando mais de metade do pacotes enviados pela fonte são recebidos pelo destino. Contudo, verificamos que à medida que aumenta o numero de dispositivos, a percentagem de pacotes recebidos também aumenta, apesar de com a utilização da aplicação SendPackets verificar-mos que nos intervalos que a rede tem 80 a 100 dispositivos, a percentagens de pacotes recebidos baixa. Enquanto que com a utilização da aplicação ON/OFF a percentagem de pacotes recebidos sobe, podendo isto dever-se a serem menos pacotes transmitidos na rede com a utilização desta aplicação. Mas por outro lado, a utilização do protocolo AODV, apenas obtêm um ótimo funcionamento, tal como os restantes protocolos quando a rede contém 80 a 100 dispositivos com a aplicação ON/OFF, devendo-se, como já referido, a ser um protocolo que demora mais tempo para estabelecer rota porque só pratica este estabelecimento quando o necessita, podendo já a aplicação geradora de tráfego estar a tentar enviar dados mesmo ainda não conhecendo um caminho da fonte até ao destino.

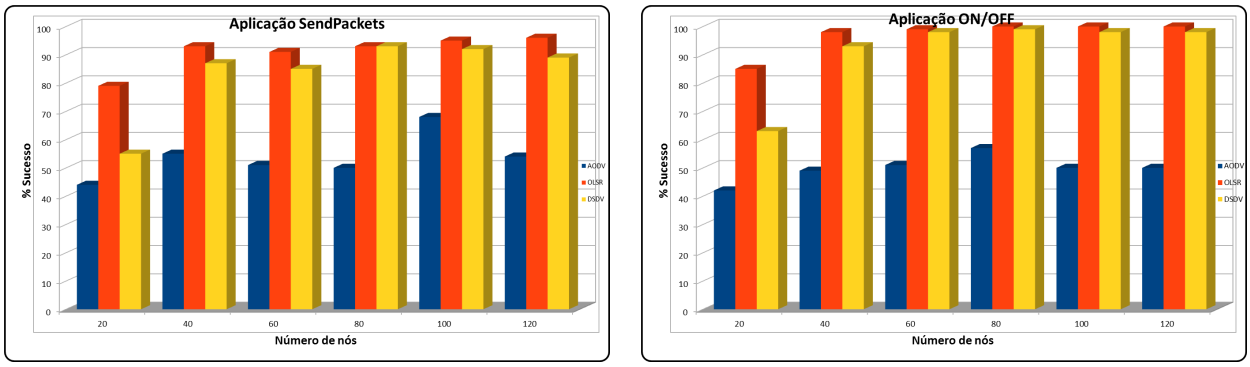


Figura 5.12: Percentagem de Sucesso na comunicação com mobilidade 1.

E continuando a apresentação acima descrita, verifica-se que os protocolos proativos, utilizando as aplicações desenvolvidas e dentro do tempo de simulação com as duas aplicações geradoras de tráfego a funcionar, obtêm uma melhor percentagem de sucesso na comunicação. Verifica-se o aumento do sucesso na comunicação à medida que aumenta o número de dispositivos na rede, sendo possível obter mais caminhos entre a fonte e o destino e uma maior abrangência dentro do espaço de

rede.

Concluindo, comparando o funcionamento das duas aplicações num mesmo espaço de rede e com o mesmo número de dispositivos, verifica-se que o protocolo OLSR obtém o melhor resultado.

Com a utilização do modelo de mobilidade 2, e utilizando as mesmas aplicações e os mesmos protocolos em funcionamento, obtemos um resumo da realização das simulações que seguem as características apresentadas na tabela 5.2.

Aplicação 1 - SendPackets	Aplicação 2 – ON/OFF
Número de nós (grupos): 20-40-60-80-100-120	**Número de nós (grupos)**: 20-40-60-80-100-120
Número de simulações por cada grupo de nós: 100	**Número de simulações por cada grupo de nós**: 100
Tempo de simulação : 150 segundos	**Tempo de simulação** : 150 segundos
Número de nós fonte – destino: 2 (1 par)	**Número de nós fonte – destino**: 2 (1 par)
Modelo de mobilidade: GaussMarkovMobilityModel – Mobilidade 2	**Modelo de mobilidade**: GaussMarkovMobilityModel – Mobilidade 2
Protocolos de encaminhamento: AODV, OLSR e DSDV	**Protocolos de encaminhamento**: AODV, OLSR e DSDV
Tipo de canal: Wireless	**Tipo de canal**: Wireless
Tipo de tráfego: CBR, 360Kbps	**Tipo de tráfego**: ON_OFF (3s/5s), 12,2Kbps
Tamanho do pacote: 1500 bytes	**Tamanho do pacote**: 372 bytes
Intervalo entre pacotes: 0,033 segundos	**Intervalo entre pacotes**: 3 segundos a ON – 5 segundos a OFF

Tabela 5.2: Parâmetros de simulação 2

Através da utilização destas características e utilizando um modelo de mobilidade dos dispositivos diferente, dentro de um espaço de rede menor, na qual pretendemos simular uma área de $120000m^2$ com a adição da altura, simulando um estrutura como um centro comercial, constituído por 3 andares com uma altura máxima de 15 metros.

Através destas caraterísticas obtemos os seguintes resultados para cada uma das métricas estudadas de acordo com o protocolo utilizado e o número de nós (dispositivos) que compõe a estrutura de rede dentro do espaço de rede definido no modelo de mobilidade 2.

Relativamente ao número de pacotes perdidos (*packet loss*), com este modelo de mobilidade verificamos que existe um número muito inferior de pacotes perdidos

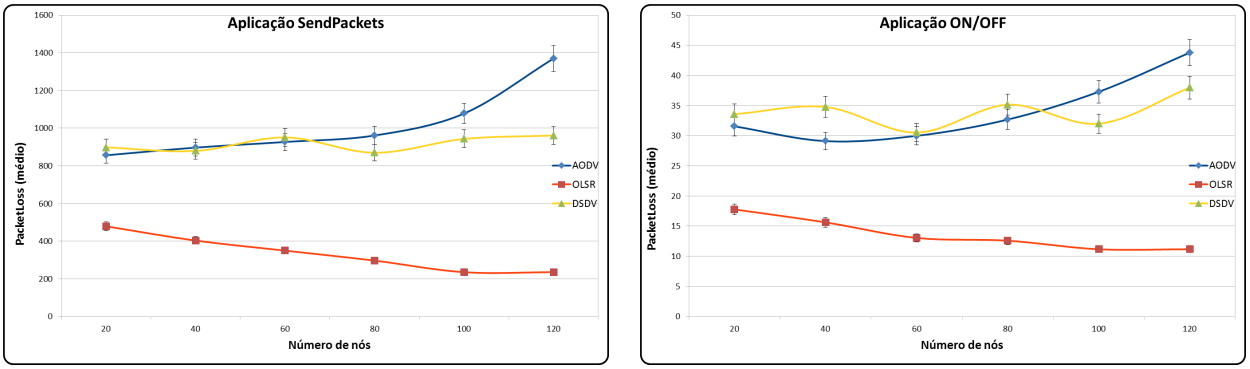


Figura 5.13: Valor médio de *PacketLoss com mobilidade 2*.

ao longo da comunicação, isto resultado de os dispositivos estarem dispostos numa área menor e a área de cobertura de cada dispositivo possivelmente abranger mais dispositivos dentro desse espaço de rede. Especificamente ao funcionamento dos protocolos, verificamos que existe uma crescente perda de pacotes à medida que aumenta o número de dispositivos na rede quando utilizamos o protocolo AODV, sendo que este à medida que existem mais nós na rede poderá demorar mais tempo até obter a melhor rota entre o dispositivo fonte e dispositivo destino, tal como alguns parâmetros indicados utilizando o modelo de mobilidade 1. Por outro lado, verifica-se que com a utilização do protocolo OLSR, este obtém sempre melhores resultados, obtendo um valor de pacotes perdidos menor ao longo que o número de dispositivos aumenta, oferecendo uma adaptação mais rápida às condições da rede, devido possivelmente à utilização dos dispositivos MPR's que oferecem uma visão da rede mais rápida obtendo os melhores caminhos e respondendo de melhor forma a problemas de conectividade na rede, através das tabelas de encaminhamento que estes dispositivos guardam.

Por outro lado, relativamente à média da soma dos atrasos de ponta a ponta para todos os pacotes recebidos, verifica-se que tanto para a aplicação SendPackets como com a utilização ON/OFF o protocolo AODV obtém maiores atrasos ao longo que se aumenta o número de dispositivos, devido a demorarem mais tempo entre o estabelecimento da rota sempre que é necessário comunicar quando existem mais dispositivos dentro da área de rede. Mesmo assim, estes valores variam muito pouco, na casa decimal, de modo que os atrasos aqui não se verificam muito

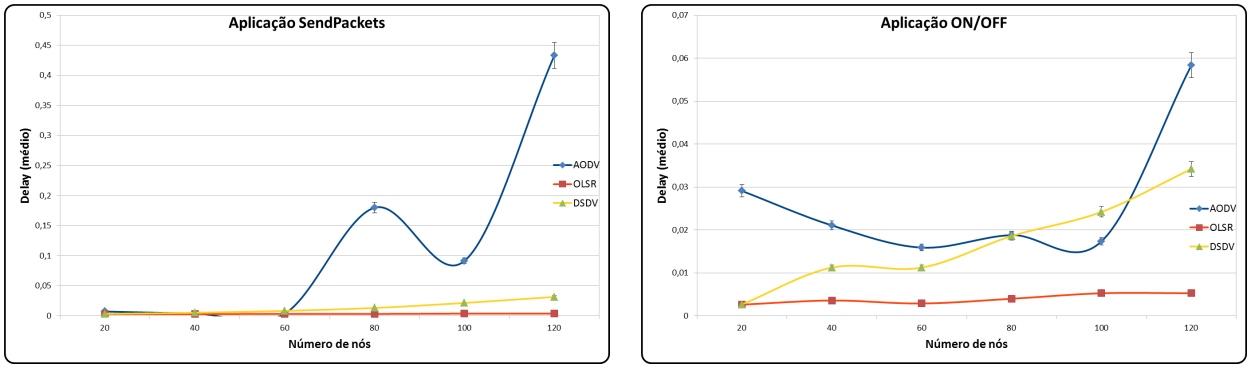


Figura 5.14: Valor médio de *Delay* com mobilidade 2.

significativos. O maior problema será na utilização da aplicação SendPackets e quando a rede é constituída por 120 dispositivos, que o protocolo AODV obtém um acréscimo acentuado no atraso de entrega dos pacotes entre a fonte e o destino, devido possivelmente ao estabelecimento da melhor de rota onde se verifica uma maior mobilidade dos dispositivos, que estando numa área mais pequena faz com que a melhor rota esteja sempre em alternância entre os dispositivos. Por fim, utilizando tanto a aplicação SendPackets, com o envio de no máximo 4090 pacotes, como utilizando a aplicação ON/OFF, que envia no máximo 184 pacotes dentro do tempo de simulação, verifica-se que o protocolo OLSR obtém sempre um valor de atraso menor. Isto é verificado do mesmo modo analisando os gráficos obtidos da média do *jitter*, figura 5.15, que à medida que o número de nós aumenta o valor desta métrica aumenta, mesmo que valores muito baixos.

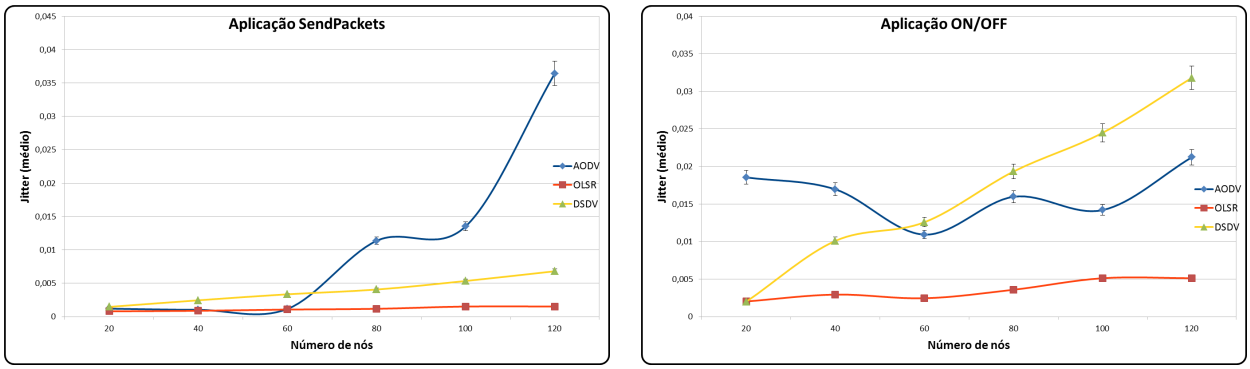


Figura 5.15: Valor médio de *Jitter* com mobilidade 2.

Neste caso, verifica-se uma diferença na utilização da aplicação ON/OFF, com a

utilização do protocolo DSDV, que verifica um ligeiro crescimento do valor de *jitter* à medida que aumentam o número de dispositivos. Este caso pode ser resultado, devido a haverem mais alterações na disposição dos dispositivos entre o tempo de envio de pacotes e o tempo em que a aplicação está sem enviar pacotes de dados, fazendo com que este protocolo perca mais tempo na atualização das suas tabelas de encaminhamento e aumente a variação dos atrasos de ponta a ponta para todos os pacotes recebidos. Isto porque esta aplicação apenas envia pacotes de 5 em 5 segundos durante 3 segundos, fazendo com que nesses intervalos possam haver várias atualizações nas suas tabelas de encaminhamento.

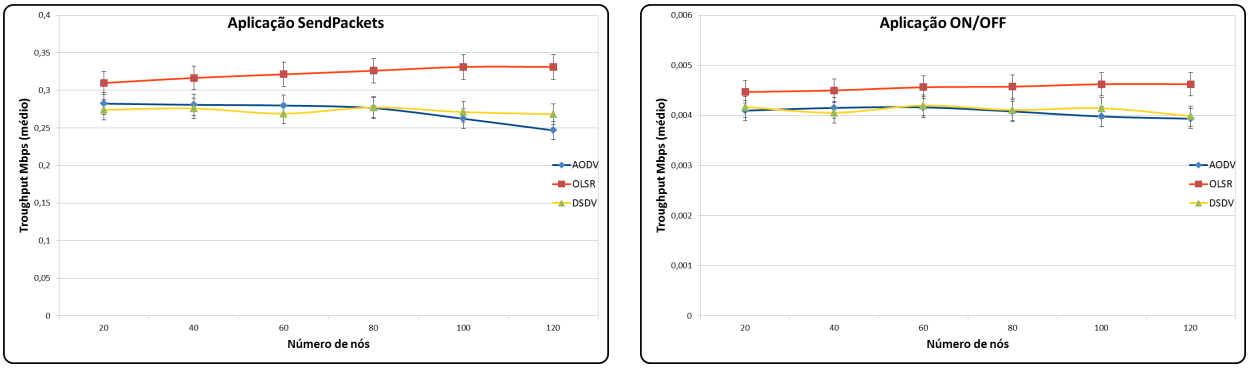


Figura 5.16: Valor médio de *Troughput* (Mbps) com mobilidade 2.

Comparando neste momento os valores de *Troughput* entre as duas aplicações, utilizando o modelo de mobilidade 2, verifica-se que com a utilização de qualquer um dos protocolos o valor de *Troughput* vai-se mantendo quase constante ao longo que o número de dispositivos aumenta. Verifica-se apenas um pequeno decréscimo utilizando a aplicação SendPackets com o protocolo AODV à medida que o número de dispositivos aumenta, possivelmente provocado por esta aplicação ter um débito superior e quando o número de dispositivos aumenta obter uma maior perda de pacotes, figura 5.13, sendo que a taxa de transferência desde a fonte para um destino diminui devido ao atraso no estabelecimento dos caminhos entre os dispositivos. Concluindo, verifica-se tal como com a utilização da mobilidade 1, que o protocolo OLSR obtém melhores resultados, podendo este resultado ser definido devido ao modo como o protocolo calcula sempre as rotas com menor custo e cada dispositivo conter a informação da topologia de rede devido à constante troca de mensagens

de ligação entre os dispositivos, utilizando apenas os dispositivos designados MPR para encaminhar os pacotes na rede.

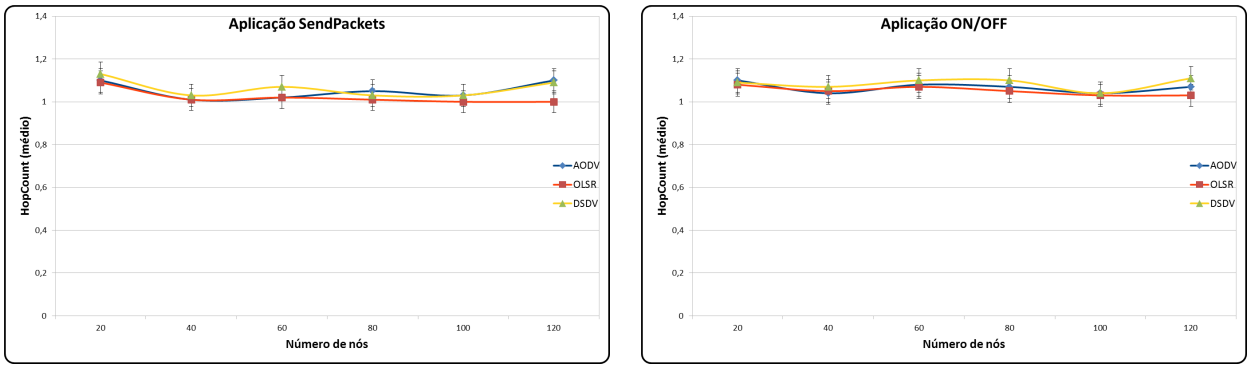


Figura 5.17: Número de Saltos com mobilidade 2.

Relativamente ao número de saltos, em comparação com a mobilidade 1, neste caso como os dispositivos estão dentro de uma área menor, verifica-se que o número de saltos tanto utilizando a aplicação SendPackets como a aplicação ON/OFF o valor rodeia em média o valor de 1 salto, estando na maioria das simulações o dispositivo destino dentro da área de abrangência do dispositivo fonte, sendo a comunicação direta na maioria dos casos para cada simulação.

Com isto, como era de esperar o número total de pacotes de dados transmitidos da fonte para o destino têm sempre o resultado máximo utilizado por cada uma das aplicações, verificado na figura 5.18.

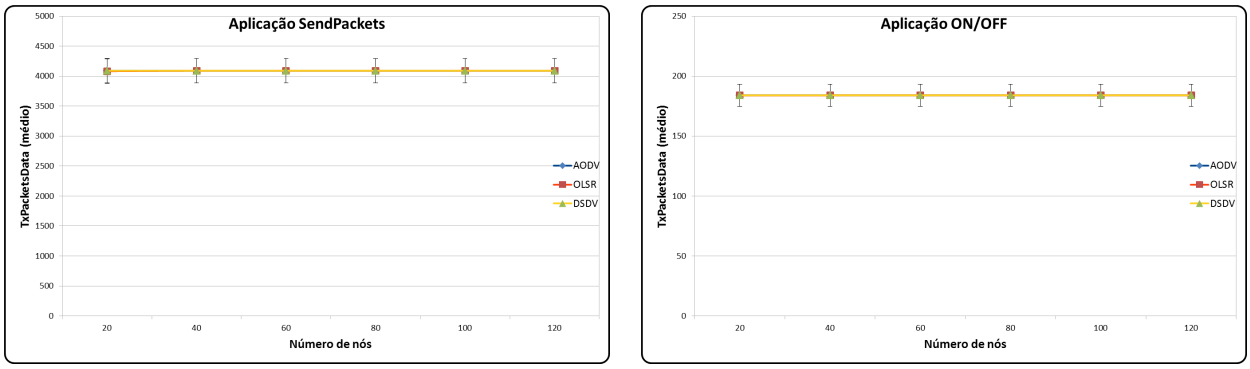


Figura 5.18: Número de pacotes de dados transmitidos com mobilidade 2.

Por outro lado, não se verifica o número total de pacotes de dados recebidos pelo dispositivo destino, que foram enviados pela fonte, sejam os valores máximos.

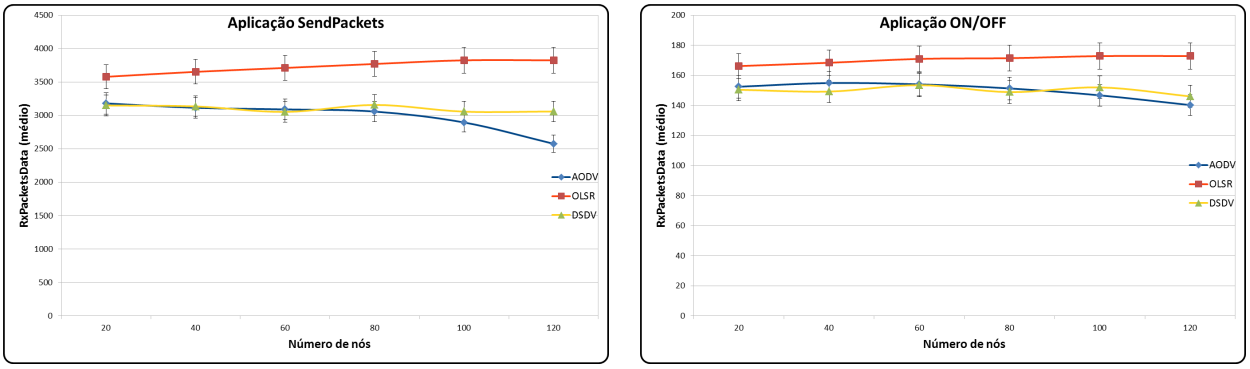


Figura 5.19: Número de pacotes de dados recebidos com mobilidade 2.

Concluindo que à medida que o número de dispositivos aumenta na rede mais pacotes de dados são perdidos na rede ou durante a comunicação, devido aos atrasos no envio dos pacotes de dados, como referenciado na figura 5.14, ou na possibilidade de pacotes serem descartados ao longo da comunicação. Contudo verifica-se um valor baixo de perda de pacotes, verificando-se a principal diferença na utilização do protocolo AODV com a utilização da aplicação SendPackets, sendo que à medida que aumenta o número de dispositivos, com a utilização deste protocolo se vão perdendo mais pacotes de dados.

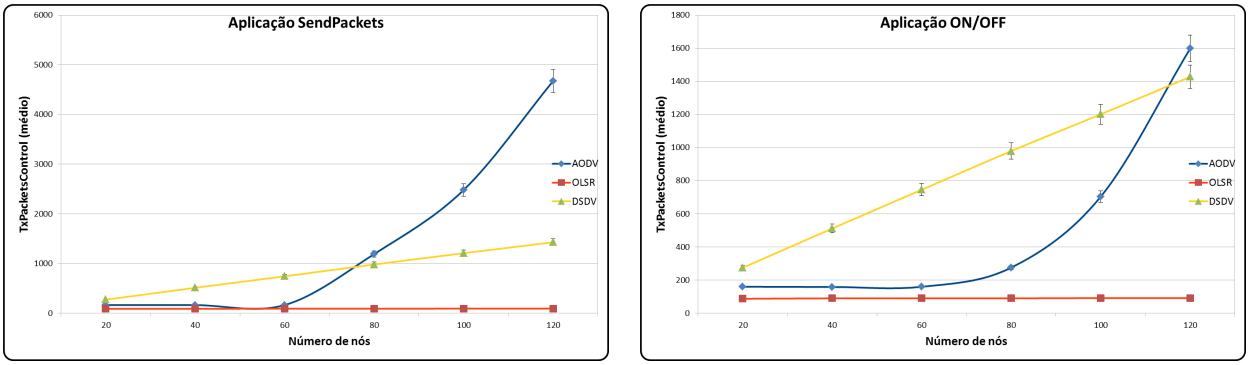


Figura 5.20: Número de pacotes de controlo transmitidos com mobilidade 2.

Comparando os resultados que representam o número de pacotes de controlo enviados de um dispositivo para os restantes dispositivos da rede, verifica-se que o protocolo AODV utilizando a aplicação SendPackets como a aplicação ON/OFF obtém um acréscimo muito acentuado à medida que o número de dispositivos aumenta, notando-se principalmente a partir dos 80 dispositivos. Este facto deve-se

ao modo como funciona este protocolo, fazendo sempre uma troca de pacotes de controlo entre todos os dispositivos sempre que é necessária comunicação entre dispositivos, logo à medida que o número de dispositivos aumenta, maior será o número de pacotes que serão trocados entres os dispositivos, e como estes obtêm diferentes posições mais rapidamente dentro do espaço de rede, mais informação será trocada de modo a obterem uma rota atualizada. Comparativamente os outros dois protocolos, tanto utilizando a aplicação SendPacktes como a aplicação ON/OFF obtêm o mesmo resultado, sendo que o DSDV como explicado com a utilização da mobilidade 1, é um protocolo que está constantemente a trocar pacotes de controlo entre todos os dispositivos da rede, e obviamente que com o aumento do número de dispositivos aumenta também gradualmente o número de pacotes de controlo enviados.

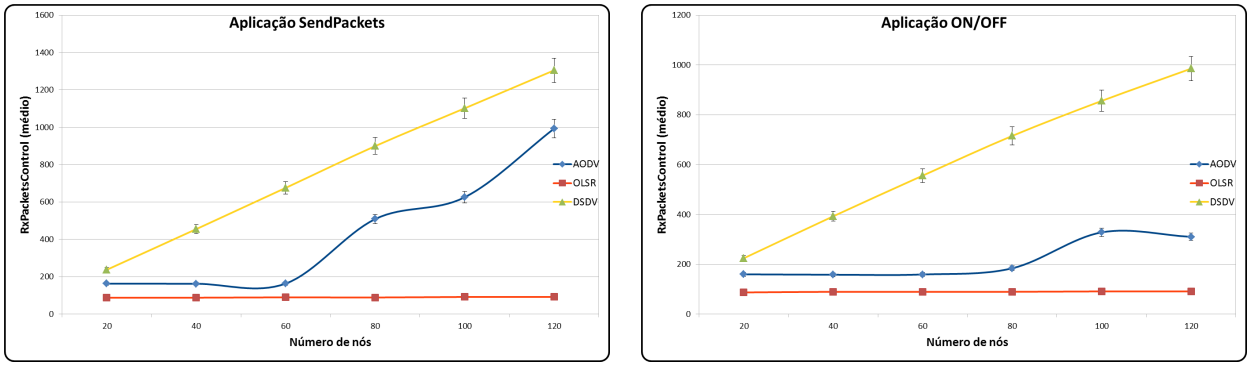


Figura 5.21: Número de pacotes de controlo recebidos com mobilidade 2.

Depois de comparados o número de pacotes de controlo enviados entre os dispositivos da rede, obtemos a comparação que representa o número de pacotes de controlo recebidos por um dispositivo a partir dos restantes dispositivos da rede. Neste caso, verificamos que como descrito com a mobilidade 1, o protocolo DSDV obtém uma constante subida de pacotes de controlo recebidos referente à constante subida de pacotes de controlo enviados consoante o aumento do número de dispositivos na rede. Por conseguinte, verifica-se um aumento gradual dos pacotes recebidos com a utilização do protocolo AODV utilizando a aplicação Sendpackets com o aumento do número de dispositivos na rede, isto correspondendo ao aumento do número de pacotes de controlo que são enviado com a utilização deste protocolo,

verificando-se que muitos dos pacotes de controlo enviados com a utilização deste protocolo não chegam a ser recebidos pelos respetivos dispositivos, possivelmente pela movimentação dos nós neste espaço de rede e pela constante alteração da posição dos nós, fazendo com que muitos dos pacotes de controlo sejam perdidos na rede ou descartados devido ao tempo que circulam na rede.

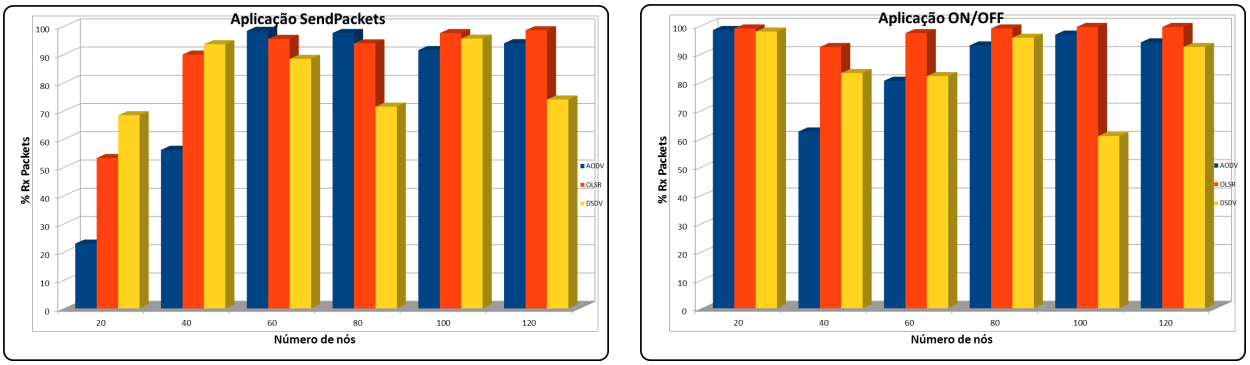


Figura 5.22: Percentagem de pacotes recebidos pelo nós destino com mobilidade 2.

Para finalizar, comparando a percentagem de pacotes recebidos, figura 5.22, pelo dispositivo destino e a percentagem de sucesso na comunicação, figura 5.23, sendo este sucesso contabilizado quando mais de 50% do pacotes enviados pelo dispositivo fonte são recebidos pelo dispositivo destino, verifica-se que com este modelo de mobilidade os valores são mais constantes utilizando a aplicação ON/OFF, sendo mais pacotes recebido no destino e o sucesso na comunicação estar sempre em volta dos 100% com esta aplicação. Contudo, verifica-se uma baixa percentagem de pacotes recebidos com a aplicação SendPackets, inicialmente quando a rede ainda é constituída por apenas 20 dispositivos, devido provavelmente à mobilidade dos nós neste espaço de rede e por ainda ser difícil haver ligação entre todos os dispositivos da rede.

Continuando a apresentação acima descrita, verifica-se que os protocolos proativos, utilizando as aplicações desenvolvidas e dentro do tempo de simulação com as duas aplicações geradoras de tráfego a funcionar, obtêm uma melhor percentagem de sucesso na comunicação, sendo que o protocolo OLSR leva sempre vantagem comparativamente com os restantes protocolos.

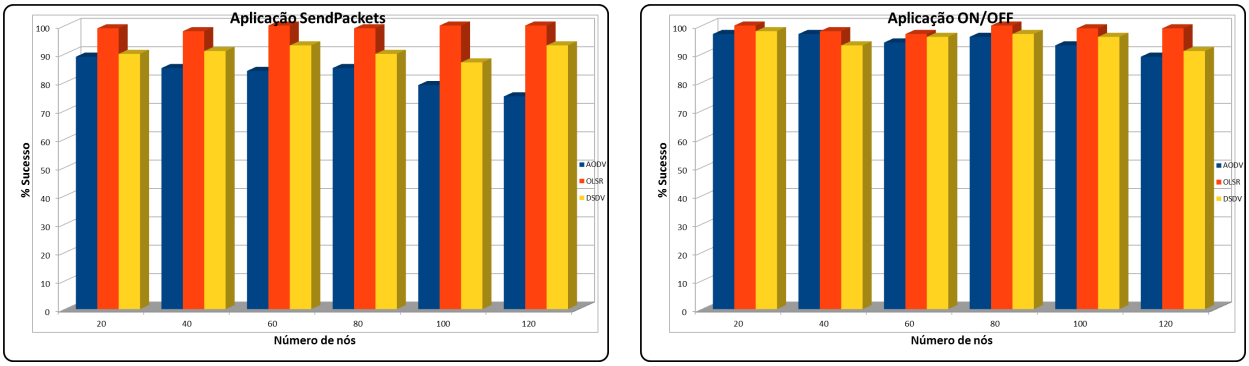


Figura 5.23: Percentagem de Sucesso na comunicação com mobilidade 2.

5.3 Sumário

Com a apresentação deste capitulo, realizou-se a exposição graficamente de todos os resultados originados pelas simulações realizadas. Foi ainda feita a tentativa da utilização de mais nós na rede, o que necessitava de um elevado processamento a partir do simulador e que demoraria demasiado tempo a finalizar cada simulação. Deste modo, expomos todos os resultados obtidos, do cálculo das métricas descritas, ao longo da comunicação entre dois dispositivos na rede. Verificando o desempenho de cada protocolo em cenários de rede diferentes, variando o modelo de mobilidade dos dispositivos e as aplicações geradores de tráfego para o mesmo cenário.

6

Conclusões

Este capitulo apresenta uma síntese dos resultados obtidos relativamente ao desempenho do encaminhamento de tráfego em redes Ad Hoc, fazendo a avaliação do desempenho através da utilização de alguma métricas importantes na área das redes de comunicações, precisamente as que são ainda mais influenciadas pela mobilidades dos dispositivos. Desta forma podemos verificar o impacto do aumento do número de dispositivos numa área definida, o impacto da sua mobilidade dentro dessa área e a diferença do desempenho do encaminhamento de dados consoante o protocolo de encaminhamento aplicado.

6.1 Síntese dos principais resultados e conclusão

Os resultados apresentados no capitulo 5 mostram a aplicação do funcionamento das redes Ad Hoc em cenários onde são modificados alguns parâmetros, tais como,

o modelo de mobilidade que os dispositivos têm dentro da área definida, a aplicação geradora de tráfego que é utilizada entre os dispositivos que estabelecem ligação, o protocolo de encaminhamento utilizado durante esse cenário de comunicação e o número de dispositivos dispersos dentro da mesma área descrita.

Este trabalho pretende fazer uma avaliação abrangente ao desempenho de três dos principais protocolos de encaminhamento em redes Ad Hoc (AODV, OLSR e DSDV). Assim, utilizando o ambiente de simulação NS-3, avaliou-se o desempenho no encaminhamento de pacotes entre dois dispositivos aleatórios na rede, através de uma análise minuciosa que inclui parâmetros e métricas importantes para o desempenho do encaminhamento em redes sem fios.

Os resultados mostram que, mesmo em redes em cenários hostis, em que a rede é carregada de tráfego e os nós estão em constante mobilidade, a transmissão de dados (por exemplo voz e vídeo) é possível sem se perder a interatividade. No entanto, com o aumento da mobilidade e número de dispositivos na rede, consoante o protocolo encaminhamento utilizado, verifica-se uma degradação na capacidade da rede de maneiras diferentes. A carga da rede afeta diretamente o tempo de acesso médio causando perdas de pacotes devido à expiração do tempo de vida dos pacotes transmitidos, enquanto que a mobilidade dos dispositivos afeta outros parâmetros relacionados com o encaminhamento, o que aumenta a perda de pacotes e os atrasos no encaminhamento. Por outro lado, um número muito baixo de dispositivos no mesmo espaço de rede, pode também causar este mesmo problema, de forma a que os dispositivos possam estar muito dispersos pelo espaço de rede e não terem alcance suficiente para comunicarem com os restantes dispositivos, sendo difícil o estabelecimento de ligação entre eles. Os resultados revelam que as mudanças na topologia e a constante movimentação dos dispositivos na rede têm um grande impacto sobre as perdas, devido principalmente a quebras de ligação e falhas na comunicação entre dispositivos intermédios na comunicação, uma vez que a densidade dos dispositivos está fortemente relacionada com a conectividade da rede. Assim, a alta conectividade e um pequeno número de saltos reduz as perdas de ligação.

Por outro lado, muitas vezes o atraso na descoberta devido a falhas de ligação

provocam também grande perda de pacotes transmitidos na rede.

Concluindo, comparando o funcionamento das duas aplicações num mesmo espaço de rede, entre dois modelos de mobilidade diferentes e com o mesmo número de dispositivos, verifica-se que o protocolo OLSR obtém sempre o melhor resultado, não adicionando grande valor de *delay* e perdas de pacotes (*packet loss*), sendo este o protocolo que melhor se adequa independente da aplicação ou do modelo de mobilidade escolhidos, utilizando conjuntos de nós entre os 20 até 120 nós na mesma área da rede.

6.2 Trabalho Futuro

Relativamente a uma possibilidade de continuação ou desenvolvimento ao que até aqui foi apresentado, concluídos os objetivos inicialmente propostos, existe um imenso conjunto de outras características e funcionalidades que poderão ser abordadas no futuro. Para outras simulações relacionadas com este tema, poderemos indicar alguns aspetos que podem ser abordados de forma a apresentar novas funcionalidades. Devido a algumas limitações, mais propriamente do simulador, sobre o número de carga (número de nós por simulação) não nos foi possível, dentro do tempo estimado, elaborar simulações com um maior número de nós. Este problema acontece devido ao elevado processamento necessário e o tempo que cada simulação demora a finalizar quando o número de nós ultrapassava um nível elevado, que no nosso caso acima de 250 nós cada simulação demorava acima de 50 horas a finalizar. Desta forma apresentamos alguns pontos que poderão ser abordados no seguimento ou num trabalho futuro nesta área.

- Elaboração de testes com um maior número de dispositivos dentro de um espaço de rede;
- Utilização de mais do que um par de dispositivos a comunicarem dentro do mesmo cenário;
- Estabelecer a criação de uma rede de dispositivos que utilizem protocolos de encaminhamento Ad Hoc diferentes;

- Estabelecer simulações com a utilização da aplicação ON/OFF em que os tempos a ON e a OFF sejam aleatórios;
- Definir diferentes modelos de mobilidade de modo a simular diferentes locais e estruturas;
- Em estudos futuros poderão ser avaliados outros parâmetros de desempenho com base em aplicações com outras configurações;
- Elaboração de um estudo a nível de questões de segurança de forma a ser aplicável como critério na escolha do protocolo confiável e seguro;
- Adicionar critérios de QoS no encaminhamento de pacotes;
- Aplicação de um estudo de simulação em um cenário real;

Bibliografia

[1] Glomosim global mobile information systems simulation library. http://pcl.cs.ucla.edu/projects/glomosim/. Accessed: 2013-04-08.

[2] Jist/swans java in simulation time / scalable wireless ad hoc network simulator. http://jist.ece.cornell.edu/index.html. Accessed: 2013-04-08.

[3] Ns-2 network simulator ns-2.30 released on sept 26, 2006. http://www.isi.edu/nsnam/ns/. Accessed: 2013-04-08.

[4] Ns-3 network simulator ns-3.16 was released on 21 december 2012. http://www.nsnam.org/. Accessed: 2013-04-08.

[5] Omnet++ simulation environment 4.3 released on april 15,2013. http://www.omnetpp.org/. Accessed: 2013-04-08.

[6] Opnet technologies, inc.(nasdaq:opnt). http://www.opnet.com/. Accessed: 2013-04-08.

[7] Tracemetrics a trace file analyzer for network simulator 3.

[8] *Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network*, volume 3, August 2005.

[9] M. Abolhasan, B. Hagelstein, and J. C.-P. Wang. Real-world performance of current proactive multi-hop mesh protocols. In *Proceedings of the 15th Asia-*

Pacific conference on Communications, APCC'09, pages 42–45, Piscataway, NJ, USA, 2009. IEEE Press.

[10] Giuseppe Anastasi, Eleonora Borgia, Marco Conti, and Enrico Gregori. Ieee 802.11 ad hoc networks: Performance measurements. *2012 32nd International Conference on Distributed Computing Systems Workshops*, 0:758, 2003.

[11] E. Baccelli and M. Townsley. IP Addressing Model in Ad Hoc Networks. RFC 5889 (Informational), September 2010.

[12] Hakim Badis and Khaldoun Al Agha. Qos routing in ad hoc networks using qolsr with no need of explicit reservation. In *In Proc. of VTC*, 2004.

[13] Azzedine Boukerche. Performance evaluation of routing protocols for ad hoc wireless networks. *Mob. Netw. Appl.*, 9(4):333–342, August 2004.

[14] I. Bradaric, R. Dattani, A. P. Petropulu, F. L. Schurgot, Jr., and J. Inserra. Analysis of physical layer performance of ieee 802.11a in an ad hoc network environment. In *Proceedings of the 2003 IEEE conference on Military communications - Volume II*, MILCOM'03, pages 1231–1236, Washington, DC, USA, 2003. IEEE Computer Society.

[15] Ch.Divakar B.Soujanya, T.Sitamahalakshmi. Study of routing protocols in mobile ad-hoc networks. *International Journal of Engineering Science and Technology*, 3(4):2622–2631, 2011.

[16] E. Belding-Royer C. Perkins and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[17] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.

[18] Gustavo Carneiro, Pedro Fortuna, and Manuel Ricardo. Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3). In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, pages 1:1–1:10, ICST,

Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[19] C. Chaudet, D. Dhoutaut, and I.G. Lassous. Performance issues with ieee 802.11 in ad hoc networking. *Communications Magazine, IEEE*, 43(7):110–116, 2005.

[20] P. Chenna Reddy and P. ChandraSekhar Reddy. Performance analysis of adhoc network routing protocols. In *Ad Hoc and Ubiquitous Computing, 2006. ISAUHC '06. International Symposium on*, pages 186–187, 2006.

[21] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[22] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Peformance Issues and Evaluation Considérations. RFC 2501, IETF, January 1999.

[23] C. Demichelis and P. Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393 (Proposed Standard), November 2002.

[24] Internet Engineering Task Force. Mobile ad-hoc networks (manet) - work-group.

[25] Mounir FRIKHA and Manel MAAMER. Implementation and simulation of olsr protocol with qos in ad hoc networks.

[26] Guoyou He. Destination-sequenced distance vector (dsdv) protocol.

[27] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. Project description. nsnam.

[28] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. ns-3 project goals. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, WNS2 '06, New York, NY, USA, 2006. ACM.

[29] Harri Holma, Juan Melero, Janne Vainio, Timo Halonen, and J Makinen. Performance of adaptive multirate (amr) voice in gsm and wcdma. In *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, volume 4, pages 2177–2181. IEEE, 2003.

[30] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard 802.11, June 1999.

[31] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pages 62–68. IEEE, August 2001.

[32] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), February 2007.

[33] Masood Khosroshahy. Study and Implementation of IEEE 802.11 Physical Layer Model in YANS (Future NS-3) Network Simulator. Master's thesis, Télécom Paris (Ecole Nationale Supérieure des Télécommunications), France, 2006.

[34] Mathieu Lacage and Thomas R. Henderson. Yet another network simulator. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, WNS2 '06, New York, NY, USA, 2006. ACM.

[35] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. Ieee 802.11 rate adaptation: a practical approach. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '04, pages 126–134, New York, NY, USA, 2004. ACM.

[36] L.A. Latiff and N. Fisal.

[37] C. Maihofer. A survey of geocast routing protocols. *Communications Surveys Tutorials, IEEE*, 6(2):32 –42, quarter 2004.

[38] J. Malek and K. Nowak, editors. *Trace graph-data presentation system for network simulator ns.*, Poland.

[39] S. Mccanne, S. Floyd, and K. Fall. http://www-nrg.ee.lbl.gov/ns/.

[40] C. Siva Ram Murthy and B.S. Manoj. *Ad Hoc Wireless Networks: Architectures and Protocols.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[41] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684 (Experimental), February 2004.

[42] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pages 90 –100, feb 1999.

[43] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the conference on Communications architectures, protocols and applications*, SIGCOMM '94, pages 234–244, New York, NY, USA, 1994. ACM.

[44] Charles E Perkins et al. *Ad hoc networking*, volume 1. Addison-wesley Reading, 2001.

[45] Das S. Perkins C., Belding-Royer E. and Chakeres I. Ad hoc on-demand distance vector routing. `http://moment.cs.ucsb.edu/AODV/aodv.html`, 2007.

[46] E.M. Royer and Chai-Keong Toh. A review of current routing protocols for ad hoc mobile wireless networks. *Personal Communications, IEEE*, 6(2):46 –55, apr 1999.

[47] Thomas Schierl, Thomas Stockhammer, and Thomas Wiegand. Mobile video transmission using scalable video coding. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 17(9), 2007.

[48] Mineo Takai, Jay Martin, and Rajive Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. In *in MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 87–94. ACM Press, 2001.

[49] Vahid Nazari Talooki and Jonathan Rodriguez. Quality of service for flat routing protocols in mobile ad hoc networks. In *Proceedings of the 5th International ICST Mobile Multimedia Communications Conference*, Mobimedia '09, pages 60:1–60:5, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[50] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[51] E. Weingartner, H. vom Lehn, and K. Wehrle. A performance comparison of recent network simulators. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–5, 2009.