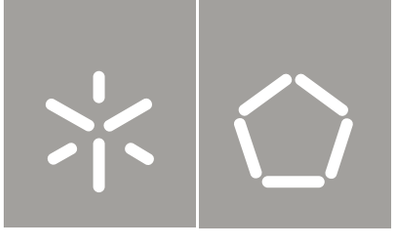
Universidade do Minho
Escola de Engenharia

Tiago Filipe Meneses Magalhães Coelho

Encaminhamento com QoS em Redes Móveis Ad Hoc

outubro de 2013

Encaminhamento com QoS em Redes Móveis Ad Hoc
Tiago Filipe Meneses Magalhães Coelho
UMinho | 2013



Universidade do Minho
Escola de Engenharia

Tiago Filipe Meneses Magalhães Coelho

Encaminhamento com QoS em Redes Móveis Ad Hoc

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

outubro de 2013

DECLARAÇÃO

Nome: Tiago Filipe Meneses Magalhães Coelho

Correio electrónico: tiagofmmc@gmail.com

TIm.: 938481553

Número do Bilhete de Identidade: 12296506

Título da dissertação:

Encaminhamento com QoS em Redes Móveis Ad Hoc

Ano de conclusão: 2013

Orientadores:

Maria João Nicolau

António Costa

Designação do Mestrado:

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia de Comunicações
Área de Especialização: Engenharia de Comunicações

Escola de Engenharia, Departamento/Centro: Informática/Algoritmi

Guimarães, 03/01/2014

Assinatura: _____

Resumo

Nos dias de hoje a grande diversidade e o aumento da capacidade dos dispositivos móveis sem fios e simultaneamente a evolução das aplicações multimédia, criou-se a necessidade de propor e avaliar formas de oferecer garantias de Qualidade de Serviço (QoS) ao tráfego fim a fim nas redes móveis ad hoc (Mobile Ad hoc Networks - MANET). Este tipo de redes tem a vantagem de possibilitar aos utilizadores de dispositivos móveis, estabelecerem rapidamente e sem assistência de um ponto de acesso uma rede entre eles, que potencie a utilização de diversos serviços. Devido às suas características, desde a tecnologia sem fios até à mobilidade dos nós, dotar este tipo de redes de garantias de qualidade de serviço no tráfego fim a fim torna-se um desafio.

Esta dissertação propõe um protocolo de encaminhamento com QoS para redes ad hoc, que se designa por Ad hoc QoS Multipath Routing with Route Stability (QMRS), que tem como objectivo suportar aplicações com requisitos de qualidade de serviço, nomeadamente requisitos no atraso fim a fim. Este protocolo tem a possibilidade de encontrar até três rotas de nós disjuntos que cumpram o requisito de QoS. Adicionalmente e com o objectivo de garantir a estabilidade do processo de encaminhamento, usa a potência de sinal das ligações entre nós vizinhos para eleger a rota mais estável, rota essa que passa a ser usada para o reenvio do tráfego. Quando se verifica a existência de rotas com uma estabilidade idêntica, dá-se preferência à rota com menor atraso fim a fim. O protocolo detém também um mecanismo de manutenção, recuperação e verificação de incumprimento do requisito de QoS nos caminhos encontrados. Este protocolo teve como base o protocolo Ad Hoc On-Demand Multipath Routing (AMR), também proposto e implementado no âmbito deste trabalho. Ambos os protocolos foram implementados e avaliados usando o simulador ns-3.

Os resultados obtidos através de várias simulações realizadas para cada um dos protocolos implementados, assim como para o protocolo Ad hoc On-Demand Distance Vector (AODV) existente no simulador, permitiram verificar que, o protocolo QMRS com os mecanismos existentes de descoberta, manutenção, recuperação rápida de rota e verificação de incumprimento do requisito de QoS nos caminhos encontrados, permite obter resultados significativamente melhores, comparativamente ao protocolo AODV e ao protocolo AMR, no que diz respeito ao atraso fim a fim, taxa de entrega de pacotes no destino e taxa de transferência efectiva.

Abstract

Nowadays with the increasing of the diversity and the capability of the mobile devices and simultaneously the evolution of multimedia applications, has created the need to propose and evaluate ways of offering guarantees of Quality of Service (QoS) for the end-to-end traffic in the Mobile Ad hoc Networks (MANET). Such networks have the advantage of enabling mobile users, establish quickly and without assistance of an access point a network between them, which make best use of various services. Due to its characteristics, from the fact that it uses a wireless technology up to the impact of node mobility, providing quality of service guarantees in this type of networks for the end-to-end traffic becomes a challenge.

This thesis proposes a QoS routing protocol for ad hoc networks, which is known as Ad hoc QoS Multipath Routing with Route Stability (QMRS), which aims to support applications with quality of service requirements, namely requirements for the end to end delay. This protocol is able to find up to three disjoint routes that complies with the requirement of QoS. Additionally, and for the purpose of guarantee the stability of the routing process, uses the signal strength of the links between neighboring nodes to elect the most stable route, such route which is now used for forwarding traffic. When it is noted routes with the same stability, preference is given to the route with the lowest end to end delay. The protocol also holds a mechanism for maintenance, recovery and verification of compliance of the QoS requirement in the discovered paths. This protocol was based on the protocol Ad Hoc On-Demand Multipath Routing (AMR), also proposed and implemented during this work. Both protocols have been implemented and evaluated using the simulator ns-3.

The results obtained through various simulations for each of the protocols implemented, as well as the Ad hoc On-Demand Distance Vector (AODV) protocol existent in the network simulator, allowed to verify that the QMRS protocol with the existent mechanisms for discovery, maintenance, rapid recovery of route and verification of compliance of the QoS requirement in the discovered paths, allow to obtain results with significant improvements compared to AODV protocol and AMR protocol, with respect to the average end to end delay, packet delivery ratio and throughput.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

AODV	Ad hoc On-Demand Distance Vector
AMR	Ad hoc On-Demand Multipath Routing
AOMDV	Ad hoc On-Demand Multipath Distance Vector
AQOR	Ad Hoc QoS On-demand Routing
DiffServ	Differentiated services
DSCP	DiffServ CodePoint
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
DV	Distance Vector
IntServ	Integrated Services
IP	Internet Protocol
MAC	Media Access Control
MANET	Mobile Ad hoc Network
MPR	Multipoint Relay
MQRD	Multipath QoS Routing for supporting DiffServ
OLSR	Optimized Link State Routing
QMRS	Ad hoc QoS On-Demand Multipath Routing with Route Stability
QoS	Quality of Service
RREP	Route Reply
RREQ	Route Request
RSVP	Resource Reservation Protocol
SLA	Service Level Agreement
SMQR	Route Stability based QoS Routing
TTL	Time to Live
UDP	User Datagram Protocol
ZRP	Zone Routing Protocol

Capítulo 1

Introdução

Este capítulo servirá para introduzir o objecto de estudo desta dissertação, apresentando a motivação para a sua realização, os objectivos em que pretende focar-se e finalmente a descrição da estrutura e organização deste documento.

1.1. Enquadramento

As redes sem fios podem operar em dois modos distintos: infra-estrutura ou ad hoc. Tipicamente as mais utilizadas são as redes de infra-estrutura, onde existe um nó especial, normalmente designado por ponto de acesso. A transferência de pacotes neste tipo de redes nunca ocorre directamente entre dois nós, sendo o ponto de acesso o responsável pela recepção dos pacotes transmitidos pelos nós da rede e pelo encaminhamento desses mesmos pacotes para o nó destino.

O objecto de estudo desta dissertação são as redes móveis ad hoc (Mobile Ad hoc Network - MANET). Estas redes são compostas por nós móveis que têm a capacidade de autonomamente, criar uma rede de comunicações entre eles sem assistência de um ponto de acesso, contrariamente às redes de infra-estrutura mencionadas anteriormente.

Cada dispositivo móvel da rede comporta-se tanto como um nó que pode gerar e receber trafego, como também como um encaminhador, ou seja, pode receber tráfego destinado a outros nós e é responsável por o encaminhar para o respectivo destino.

Com o aumento da diversidade e capacidade dos dispositivos móveis sem fios, tais como portáteis, smartphones, tablets e PDAs, a forma como os utilizadores pretendem ter acesso ou utilizar alguns serviços através destes dispositivos tem evoluído muito nos últimos anos. As redes móveis ad hoc criam a possibilidade deste tipo de dispositivos estabelecerem facilmente e rapidamente uma rede entre eles: designada por Rede Ad Hoc, Essas redes potenciam a utilização de diversos serviços que envolvam a partilha de informação disponível nos vários dispositivos que estabelecem a rede.

Neste tipo de redes a capacidade de garantir o encaminhamento de tráfego com garantias de qualidade de serviço (QoS) de forma a cumprir certas exigências pretendidas pelo cliente ao utilizar certo serviço, constitui um desafio ainda maior do que fazê-lo nas redes fixas. É necessário ter em atenção vários factores, a própria topologia de rede, por ser dinâmica e imprevisível torna-se um obstáculo uma vez que está em constante alteração, devido à mobilidade dos nós, à entrada e saída de nós na rede e às quebras de ligações constantes, que provocam alterações inesperadas das rotas.

Parte da investigação actual nestas redes tem passado por analisar as aplicações de *streaming* de conteúdos multimédia, áudio ou vídeo, e verificar se é viável a utilização deste tipo de aplicações. Nas redes Ad hoc a transmissão deste género de conteúdos é sempre muito exigente, impondo certos requisitos que tornem viável a sua utilização. Normalmente são estabelecidas métricas mais apropriadas do que o número de saltos para o destino, que podem alterar a decisão da rota a utilizar quando se pretende transmitir. Exemplos desse tipo de métricas são o atraso entre a emissão e recepção de um pacote entre o nó fonte e o nó destino, a variação neste atraso, a largura de banda, a taxa de perda de pacotes máxima tolerada, a taxa de pacotes entregues com sucesso no destino, a estabilidade da rota, etc.

Cada nó móvel tem também limitações, características que variam de dispositivo para dispositivo. Estes nós da rede são dispositivos móveis normalmente alimentados a baterias, logo a energia disponível poderá ser um factor a ter em conta, uma vez que os nós gastam pouca energia quando estão activos e a escutar o meio, mas despendem mais energia sempre que fazem alguma transmissão de pacotes para a rede.

No contexto actual das redes móveis ad hoc existem vários protocolos de encaminhamento. Normalmente são classificados em duas categorias de acordo com a sua estratégia: os proactivos (table-driven) e os reactivos (on-demand).

Os protocolos de encaminhamento proactivos requerem que os nós da rede estejam periodicamente a actualizar as rotas para todos os destinos possíveis, para que quando um pacote precise de ser encaminhado, a rota já seja conhecida e possa ser utilizada imediatamente. Por qualquer alteração que ocorra na topologia é enviado para a rede uma notificação, para que todos os nós tenham o conhecimento dessa ocorrência. Alguns dos protocolos proactivos existentes são por exemplo o Optimized Link State Routing Protocol (OLSR) e o Destination-Sequenced Distance Vector

(DSDV).

Os protocolos de encaminhamento reactivos tentam construir as rotas apenas quando o nó fonte pretende transmitir. Desta forma a topologia de rede só é descoberta quando for necessário. Quando um nó pretende enviar pacotes para um determinado destino mas não tem conhecimento da rota, inicia o processo de descoberta da rota na rede. Uma vez encontrada, a rota é guardada na tabela de encaminhamento até o nó destino ficar inacessível ou até a rota não ser mais necessária. Alguns dos protocolos reactivos existentes são por exemplo o Ad-hoc On-demand Distance Vector (AODV) e o Dynamic Source Routing (DSR).

Os modelos de QoS existentes nas redes de infra-estrutura podem ser classificados em dois tipos, Serviços Integrados (IntServ) e Serviços Diferenciados (DiffServ). O modelo IntServ consiste em fornecer reservas explícitas por fluxo através do protocolo RSVP. No pedido de reserva são indicados os requisitos de QoS pretendidos. Caso seja possível cumprir as condições indicadas é efectuada a reserva, se não for possível, a reserva será negada. O modelo DiffServ oferece um tratamento diferenciado de pacotes de acordo com a classe de serviço a que pertencem, sendo necessário a marcação dos pacotes para ficarem associados a uma determinada classe de serviço. Cada classe possui os seus próprios requisitos de QoS e os pacotes recebem um tratamento diferenciado em cada nó, mediante a classe a que pertencem (por exemplo através da criação de diferentes filas de espera por cada classe de serviço).

Parte da investigação actual nas redes ad hoc consiste em analisar alguns protocolos de encaminhamento existentes, tentando se possível melhora-los de forma a conseguir dar suporte de QoS. As soluções propostas para dar garantias de QoS neste tipo de redes passam muitas vezes por soluções diferentes dos modelos IntServ e DiffServ descritos e muito utilizados em redes de infra-estrutura fixa. Há quem proponha soluções que utilizam alguns conceitos existentes nestes modelos, enquanto outras optam por tentam dar garantias de QoS com diferentes abordagens, devido à dificuldade de aplicar estes modelos directamente nas redes móveis ad hoc.

1.2. Objectivos e Contribuições

Devido às características das redes móveis ad hoc, dotar este tipo de redes de garantias de qualidade de serviço no tráfego fim a fim torna-se um desafio. Neste sentido, o projecto desenvolvido tem como

objectivo inicial, fazer um levantamento das vantagens e limitações das várias abordagens existentes para encaminhamento em redes Ad hoc. Investigar também as abordagens existentes para o encaminhamento com QoS em redes Ad hoc, tendo como meta apurar se é viável a sua utilização em aplicações exigentes, com determinados requisitos de qualidade de serviço.

Após as estratégias de encaminhamento estudadas, será criada uma plataforma de testes, que permita efectuar a implementação da solução proposta, de forma a conseguir avaliar se esta obteve um bom desempenho, e concluir se é viável a implementação de encaminhamento com QoS nas redes móveis Ad-hoc.

O projecto desenvolvido apresenta um protocolo de múltiplos caminhos de nós disjuntos para as redes ad hoc, que se designa por Ad Hoc On-Demand Multipath Routing (AMR) e um protocolo de encaminhamento com QoS, que se designa por Ad hoc QoS Multipath Routing with Route Stability (QMRS).

O protocolo AMR tem por objectivo possibilitar durante a descoberta de rota, encontrar múltiplos caminhos de nós disjuntos entre uma fonte e um destino. Desta forma, quando ocorre uma quebra de ligação entre dois nós pertencentes ao caminho principal, é possível efectuar a comutação para uma rota alternativa para que a fonte possa dar continuidade à transmissão dos dados, sem a necessidade de iniciar um novo processo de descoberta de rota. Neste sentido, este protocolo, permite reduzir o número necessário de pacotes de controlo na rede e diminuir o atraso durante a transmissão do fluxo de dados entre as fontes e os destinos.

O protocolo QMRS teve como base o protocolo ARM, tendo sido acrescentadas outras funcionalidades para que este cumpra os objectivos pretendidos. O protocolo QMRS tem como objectivo suportar aplicações com requisitos de qualidade de serviço, nomeadamente requisitos no atraso fim a fim. Para isso começa por tentar descobrir até três rotas de nós disjuntos que cumpram o requisito de QoS. Adicionalmente e com o objectivo de garantir a estabilidade do processo de encaminhamento, usa a potência do sinal recebido das ligações entre nós vizinhos para eleger a rota mais estável, rota essa que passa a ser usada para o reenvio do tráfego. Quando se verifica a existência de rotas com uma estabilidade idêntica, dá-se preferência à rota com menor atraso fim a fim. O protocolo detém também um mecanismo de manutenção, recuperação e verificação de incumprimento do requisito de QoS nos caminhos encontrados.

1.3. Estrutura da dissertação

No capítulo 1 é feita uma breve introdução ao tema da dissertação. É apresentada a motivação para a realização deste trabalho e os principais objectivos e contribuições. Este capítulo termina com a descrição da estrutura da dissertação.

No capítulo 2 é feita uma revisão sobre o estado da arte, que inclui uma introdução às redes móveis ad hoc e às características e dificuldades no processo de encaminhamento usado neste tipo de redes. Inclui uma descrição das diferentes abordagens utilizadas nas várias categorias de protocolos de encaminhamento existentes nas redes ad hoc, descrevendo os protocolos mais referenciados para cada uma das categorias mencionadas. Inclui a descrição de encaminhamento com QoS dos modelos de Qos típicos das redes de infra-estrutura e apresenta uma discussão da possibilidade destes serem aplicados às redes móveis ad hoc. Descreve ainda a dificuldade na concepção de protocolos de encaminhamento, sendo descritos alguns mecanismos e abordagens possíveis. E termina com a descrição de vários modelos e protocolos de encaminhamento com QoS existentes nas redes móveis ad hoc.

O capítulo 3 descreve os dois protocolos de encaminhamento concebidos. Inicialmente descreve o protocolo de encaminhamento de múltiplos caminhos de nós disjuntos e os seus mecanismos: o protocolo ARM. Por último, descreve o protocolo de encaminhamento com QoS concebido (protocolo QMRS), tendo como base o protocolo AMR, uma vez que resulta de uma extensão a este.

No capítulo 4 é feita uma comparação entre alguns dos simuladores de redes, destacando-se o simulador ns-3 usado para a implementação dos protocolos de encaminhamento propostos.

No capítulo 5 apresentam-se os resultados experimentais, ou seja, é feita uma descrição, análise e discussão dos resultados obtidos.

Por fim, o capítulo 6 apresenta as conclusões desta dissertação, resumindo os objectivos cumpridos e apontando algumas propostas de trabalho futuro.

Capítulo 2

Encaminhamento em Redes Ad Hoc

As redes móveis ad hoc (MANET) são compostas por nós móveis, que autonomamente decidem criar uma rede para comunicarem entre si. Utilizam o meio sem fios para transmissão, sendo que podem utilizar diversas tecnologias neste tipo de comunicação.

Estas redes não têm a necessidade da existência de qualquer infra-estrutura. Os nós que integram este tipo de redes conseguem estabelecer facilmente e rapidamente uma rede entre si.

A topologia da rede não é pré-determinada, mas sim dinâmica e imprevisível, com alterações constantes devido à mobilidade dos nós.

Pelo facto de que os nós pertencentes a este tipo de redes poderem-se movimentar por decisão própria em qualquer direcção e a diferentes velocidades, as ligações entre os nós que se encontram no alcance de transmissão alteram-se constantemente. O alcance de transmissão de cada nó é limitado, sendo por isso muitas vezes necessário recorrer a nós intermédios para encaminhar os pacotes até ao nó destino. Por este factor, estas redes são designadas redes de múltiplos saltos, em que cada nó da rede se comporta tanto como um nó que pode gerar e receber tráfego, assim como ter a funcionalidade de encaminhador, ou seja, receber pacotes destinados a outros nós e terá de tomar decisões de encaminhamento, retransmitindo o pacote para que este possa chegar ao respectivo destino.

Este tipo de redes devido à inexistência de uma infra-estrutura, às características do meio de comunicação utilizado e à mobilidade dos nós, colocam vários desafios aos protocolos de encaminhamento, para que este obtenha um bom desempenho. Neste sentido, verificam-se várias investigações nesta área para tentar melhorar os protocolos existentes, desde acrescentar outras características próprias neste tipo de redes, como a energia existente dos nós [1] [2] [3], a potência do sinal [4] [5], a informação geográfica e tentar prever a movimentação dos nós [6] [7], diferentes interacções com a camada MAC [8] [9] [10] [11], etc.

Como referido, a topologia destas redes não é pré-conhecida, os nós formam e abandonam a

rede quando pretendem, assim como se movimentam arbitrariamente provocando alterações constantes na topologia, a não existência de um controlo centralizado e o meio de comunicação sem fios, são a grande dificuldade para a realização do encaminhamento nas redes móveis ad hoc. Neste sentido, o objectivo principal de um protocolo de encaminhamento neste tipo de redes será a descoberta e manutenção de rotas tendo em conta estas dificuldades, para que os nós possam comunicar entre si de forma eficiente, transmitindo pacotes e que esses possam chegar ao destino pretendido.

2.1. Classificação dos protocolos de encaminhamento

Neste capítulo são apresentados alguns dos protocolos de encaminhamento existentes nas redes móveis ad hoc. Estes protocolos são normalmente classificados em três categorias: proactivos (table-driven), reactivos (on-demand) ou híbridos.

Para cada uma das categorias mencionadas, será feita uma descrição do seu funcionamento e serão descritos alguns dos protocolos mais referenciados.

2.1.1. Protocolos Reactivos

Os protocolos de encaminhamento reactivos tentam obter a informação de uma rota para determinado destino, apenas quando o nó fonte pretende transmitir. Não fazem uma actualização periódica das tabelas de encaminhamento, as rotas neste tipo de protocolos são obtidas a pedido.

Neste tipo de protocolos, um nó da rede que pretende enviar pacotes para um determinado nó destino, mas não tem conhecimento de uma rota, terá de iniciar o processo de descoberta da rota na rede. Este processo é realizado com o envio em broadcast de um pacote com o pedido de rota. Os nós vizinhos ao receber o pacote retransmitem-no, e vai sendo difundido pelos restantes nós até que possivelmente o pacote chega ao nó destino, ou a um nó da rede com o conhecimento de uma rota para o destino. Esse nó ao receber o pedido, envia uma resposta para o nó fonte, sendo a mensagem encaminhada segundo o percurso inverso ao que o pedido de rota percorreu até chegar a este nó.

Após a descoberta de uma rota para o destino, a informação da rota é guardada na tabela de encaminhamento até o nó destino ficar inacessível ou até a rota não ser mais necessária.

Este tipo de protocolos de encaminhamento têm a vantagem de não inundarem constantemente

a rede com pacotes de controlo para manter as tabelas de encaminhamento actualizadas, como se verifica nos protocolos de encaminhamento proactivos.

Nas redes móveis ad hoc, a partilha da largura de banda entre os nós vizinhos, e as limitações que estes dispositivos normalmente apresentam, quer seja ao nível de capacidade computacional ou por serem alimentados a bateria, é por vezes vantajoso manter os nós da rede inactivos quando não há necessidade de encaminhar pacotes.

Os protocolos reactivos pelas vantagens descritas, têm-se tornado os protocolos de encaminhamento mais utilizados nas redes móveis ad hoc [12], mas nestes protocolos também se verificam algumas desvantagens em relação aos protocolos de encaminhamento proactivos. Pelo facto de as rotas serem obtidas a pedido, é acrescentado um atraso neste processo de descoberta de rota, enquanto nos protocolos proactivos as rotas são pré-calculadas e podem ser utilizadas de imediato.

Na Secção 2.3, vão ser descritos alguns dos protocolos de encaminhamento reactivos mais referenciados, o protocolo Ad-hoc On-demand Distance Vector (AODV) [13] e o protocolo Dynamic Source Routing (DSR) [14].

2.1.2. Protocolos Proactivos

Os protocolos de encaminhamento proactivos têm como objectivo, manter a informação actualizada das tabelas de encaminhamento. Para isso, é necessário uma troca periódica de mensagens entre os nós, de forma a obter informação de rotas para cada nó destino pertencente à rede.

Desta forma, quando um nó necessita de encaminhar um pacote para determinado destino, pode consultar a sua tabela de encaminhamento, visto já conter uma rota actualizada que pode ser utilizada de imediato.

Estes protocolos, com a evolução dos protocolos reactivos, têm vindo a ser menos utilizados devido aos problemas que apresentam nas redes móveis ad hoc. Com a mobilidade dos nós e a imprevisibilidade existente neste tipo de redes, com as alterações constantes na topologia da rede, manter as tabelas de encaminhamento actualizadas torna-se uma tarefa difícil, sendo requerido por isso o envio constante de pacotes de controlo na rede.

Devido ao facto de que os nós que constituem as redes móveis ad hoc, serem normalmente dispositivos que possuem características limitadas, alimentados a bateria e com limitações a nível computacional, a exigência imposta neste tipo de protocolos pode ser um problema.

Na Secção 2.2, vão ser descritos alguns dos protocolos de encaminhamento proactivos mais referenciados, o protocolo Destination-Sequenced Distance Vector (DSDV) [15] e o Optimized Link State Routing (OLSR) [16].

2.1.3. Protocolos Híbridos

Os protocolos de encaminhamento híbridos procuram combinar as abordagens dos protocolos proactivos e reactivos. Tentam aproveitar as vantagens existentes em cada uma das abordagens, a baixa latência existente nos protocolos proactivos e a menor sobrecarga de mensagens de controlo na rede da abordagem dos protocolos reactivos. Os protocolos de encaminhamento híbridos podem apresentar um comportamento proactivo ou reactivo, dependendo da situação, a sua vantagem será permitir essa flexibilidade.

A Secção 2.4 descreve um dos protocolos de encaminhamento híbridos mais referenciados, o protocolo Zone Routing Protocol (ZRP) [17].

2.2. Descrição de Protocolos de Encaminhamento Proactivos

2.2.1. Destination-Sequenced Distance Vector (DSDV)

O protocolo Destination-Sequenced Distance Vector [15] é baseado no algoritmo de Bellman Ford. É um protocolo proactivo, isto significa que cada nó da rede tenta manter actualizadas as entradas na tabela de encaminhamento para todos os nós da rede. As tabelas são actualizadas periodicamente ou quando existe uma mudança significativa na topologia da rede. Deste modo há sempre uma rota para qualquer destino na rede se a topologia não se alterar muito. As tabelas de encaminhamento para além do destino, próximo salto e o custo do caminho, contêm também o último *sequence number* conhecido do destino.

Este *sequence number* do destino permite evitar ciclos no encaminhamento, durante as actualizações periódicas entre vizinhos. Um nó da rede ao receber a mensagem de actualização compara o *sequence number* do destino da mensagem com o *sequence number* que conhece (contido na tabela), sendo o maior *sequence number* o que corresponde à informação da rota mais recente e será essa a utilizada. No entanto, se há informação de duas rotas com o mesmo *sequence number* do destino, dá-se preferência à que detiver uma melhor métrica. A métrica de custo utilizada neste

protocolo é o número de saltos, logo para este caso, seria escolhida a rota com um menor número de saltos.

As entradas que não forem actualizadas por um determinado período de tempo são consideradas obsoletas, e portanto são removidas.

As mensagens de actualização contêm também um *sequence number* da mensagem, quando se pretende enviar a informação de rotas válidas, este campo é incrementado por dois, o que significa que corresponderá sempre a um número par. Se um nó quer notificar os outros nós da rede de uma rota inválida, a mensagem de actualização irá com um *sequence number* impar, para que os restantes nós possam identificar que a informação diz respeito a uma rota inválida e procederem à remoção da entrada na tabela de encaminhamento correspondente.

Este protocolo tem como desvantagem as actualizações periódicas, uma vez que consome muita bateria dos dispositivos móveis que constituem a rede, sendo este um recurso limitado, mesmo que essas rotas não sejam necessárias. No entanto, para pequenas topologias e não sofrerem muitas alterações, ou seja, não ocorrer muita movimentação dos nós da rede, este protocolo pode ser eficiente, pois contém uma rota para qualquer destino sempre que for necessário.

2.2.2. Optimized Link State Routing (OLSR)

O protocolo Optimized Link State Routing [16], é uma optimização do algoritmo Link State (LS). Este protocolo introduz o conceito de Multipoint Relays (MPR). O conjunto MPR é seleccionado de modo a que cubra todos os nós a dois saltos de distância. O nó N, que é seleccionado como um multipoint relay pelos vizinhos, anuncia periodicamente a informação de quem foi seleccionado como um MPR. É através das mensagens Hello que se ficam a conhecer os vizinhos de dois saltos, bem como o anúncio das escolhas do MPR.

As mensagens de controlo são recebidas e processada por todos os vizinhos de N, mas apenas os vizinhos que estão no conjunto MPR do nó N retransmitem a mensagem. Os nós MPR enviam a informação do estado das suas ligações a todos os nós vizinhos que o elegeram como MPR. Este mecanismo permite que todos os nós sejam informados de um subconjunto de ligações. Deste modo, consegue-se minimizar a informação de controlo, ao evitar a necessidade de que todos os nós teriam em enviar mensagens de controlo para todos os nós da rede.

2.3. Descrição de Protocolos de encaminhamento Reactivos

2.3.1. Dynamic Source Routing (DSR)

O protocolo Dynamic Source Routing (DSR) [14] é um protocolo reactivo que utiliza uma abordagem de encaminhamento source routing. Este protocolo durante a descoberta de rota inunda a rede, com a transmissão em broadcast de uma mensagem Route Request (RREQ) até que possivelmente esta chegue ao destino. Durante a retransmissão da mensagem RREQ entre os nós intermédios até ao destino, o caminho percorrido é colocado em cache, isto é, cada nó pertencente a esse caminho antes de retransmitir a mensagem insere no cabeçalho da mensagem a identificação do nó. Desta forma, quando a mensagem chega ao destino, este tem conhecimento exacto do caminho percorrido pela mensagem, e irá proceder ao envio da resposta ao pedido de rota através da transmissão em unicast de uma mensagem Route Reply (RREP) para a fonte pelo mesmo caminho (sentido inverso) ao percorrido pela mensagem RREQ. A fonte ao receber a resposta ao pedido de rota, antes de dar início à transmissão dos dados, incluiu o caminho encontrado no cabeçalho dos pacotes, para que, durante a sua transmissão, estes possam ser encaminhados de acordo com essa informação.

A Figura 2.1 ilustra o conceito de source routing, sendo o caminho entre a fonte e o destino encontrado durante o processo de descoberta de rota que será inserido no cabeçalho dos pacotes, que neste caso corresponde ao caminho B – C – destino. Desta forma os nós pertencentes ao caminho ao receberem o pacote retiram do cabeçalho a sua identificação, e o próximo nó desse caminho, corresponderá ao próximo salto para que o pacote possa alcançar o destino.

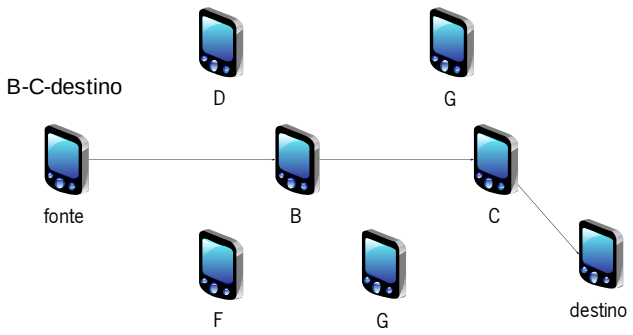


Figura 2.1: Exemplo do protocolo DSR com a abordagem de encaminhamento source routing

2.3.2. Ad hoc On-Demand Distance Vector (AODV)

O Protocolo de encaminhamento reactivo Ad hoc on-demand distance vector (AODV) é definido no RFC 3561, concebido por Charles Perkins e Elizabeth Belding-Royer [13]. Este protocolo aproveita alguns dos conceitos utilizados na descoberta e manutenção de rotas no protocolo Dynamic Source Routing (DSR) [14] e o conceito de número de sequência utilizado no protocolo Destination-Sequenced Distance Vector (DSDV) [15].

Neste protocolo estão definidos três tipos de mensagens, Route Requests (RREQ), Route Replies (RREP) e Route Errors (RERR). Um nó fonte quando pretende transmitir para um determinado destino, vai consultar a sua tabela de encaminhamento e verificar se contém uma rota válida para o destino pretendido, caso não possua, terá de iniciar o processo de descoberta de rota.

Este processo de descoberta de rota consiste no envio em *broadcast* de uma mensagem RREQ para os seus nós vizinhos, de forma a encontrar uma rota para o destino. Os seus vizinhos retransmitem o pedido e este vai sendo retransmitido nos restantes nós, para que desta forma a mensagem seja difundida pela rede até que possa chegar ao nó destino, ou então a um nó intermédio que possua uma rota válida para esse destino. A rota só é considerada válida, no caso de o nó intermédio conter na sua tabela de encaminhamento uma rota com um *destination sequence number* igual ou superior ao contido na mensagem RREQ recebida.

Cada nó da rede mantém o seu próprio sequence number e broadcastID. Antes do envio do pacote RREQ por parte do nó fonte o campo broadcastID é incrementado. Através desta informação e do endereço do nó fonte (Originator IP Address), é possível identificar unicamente cada pacote RREQ. Para além destes dois campos, a mensagem RREQ contém ainda os campos destination ip address, e os campos originator sequence number e destination sequence number, essenciais para evitar ciclos no encaminhamento, permitem distinguir pedidos sucessivos e verificar qual a mensagem mais recente.

Após um nó emitir em *broadcast* uma mensagem RREQ, este fica à espera de receber um pacote RREP, caso não receba uma resposta durante um período de tempo (NET_TRAVERSAL_TIME), o nó pode iniciar o pedido de descoberta de rota, enviando novamente em broadcast um RREQ, estando definido um número máximo de tentativas (RREQ_RETRIES).

Na Figura 2.2 pode-se observar o início da descoberta de rota por parte do nó fonte. E na Figura

2.3 verifica-se que os nós A, B e C não contêm uma rota para o destino, logo continuam a retransmitir os pacotes RREQ, e o mesmo se verifica por parte dos nós D e E.

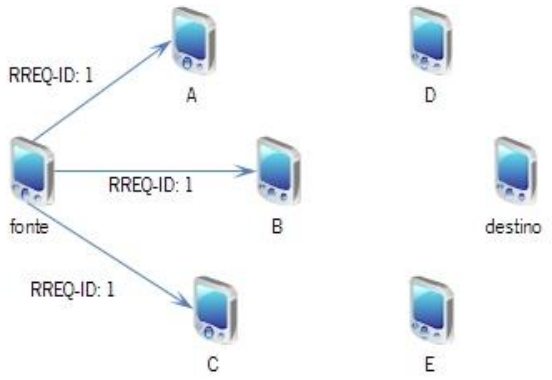


Figura 2.2: Transmissão em broadcast do pacote RREQ no nó fonte

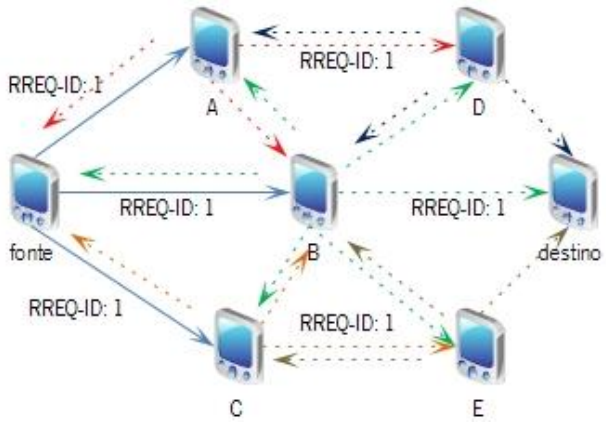


Figura 2.3: Envio do pacote RREQ em broadcast nos nós A,B e C, e retransmitido nos nós D e E.

Após a recepção do RREQ no próprio nó destino, ou num nó intermédio que contenha uma rota válida para o destino (o destination sequence number contido na sua tabela de encaminhamento terá de ser igual ou superior ao que consta no pacote RREQ recebido) será enviada uma resposta ao pedido de descoberta de rota através de uma mensagem RREP. Esta mensagem percorre o caminho inverso, de volta até ao nó que originou o processo de descoberta de rota, demonstrado na Figura 2.4. Isto só é possível realizar porque os nós intermédios ao receberem o primeiro RREQ (sendo os restantes descartados, visto conterem o mesmo numero de sequência), antes de o retransmitir guardam o endereço do seu vizinho do qual receberam a mensagem (Reverse Path Setup), para que após a descoberta da rota, a reposta (RREP) percorra o caminho inverso até ao nó fonte.

Cada nó intermédio ao receber o primeiro pacote RREP guarda a informação do nó que enviou a

mensagem e o sequence number do destino (Forward Path Setup) e retransmite o pacote. Após a sua retransmissão, caso receba mais pacotes RREP, verifica o campo destination sequence number e só o retransmite no caso de este pacote contiver um destination sequence number superior, ou se este for igual mas o número de saltos seja inferior.

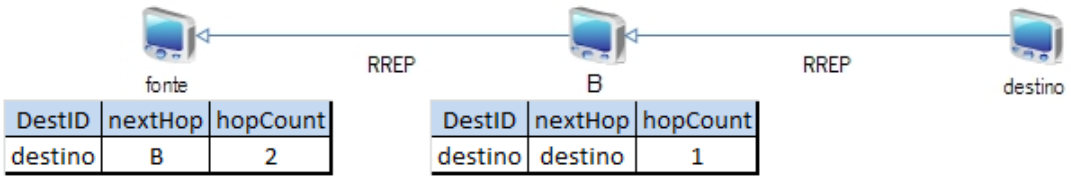


Figura 2.4: Envio do pacote RREP e tabelas de encaminhamento.

Os nós que contêm rotas activas, efectuam uma monitorização do estado das suas ligações com os nós que sejam o próximo salto a efectuar nessas rotas.

Quando é detectada uma falha na ligação (p.e. um nó vizinho deixou de estar no seu alcance de transmissão) é invocado o processo de manutenção de rotas. O nó que se apercebe da falha envia uma mensagem RERR para o seu nó vizinho *upstream*, de forma a indicar qual o nó pertencente à rota activa deixou de estar acessível. Os seus vizinhos ao receberem a mensagem RERR processam-na e retransmitem-na para os seus vizinhos no sentido *upstream,* para que a notificação da falha possa ser recebida no nó fonte. O nó fonte ao receber a mensagem RERR, caso ainda seja necessário, poderá iniciar um novo processo de descoberta de rota para o destino.

2.4. Descrição de Protocolos de encaminhamento Híbridos

2.4.1. Zone Routing Protocol (ZRP)

O protocolo de encaminhamento Zone Routing Protocol (ZRP) [17], é um protocolo hibrido que combina as abordagens dos protocolos proactivos e dos reactivos, para aproveitar as vantagens existentes em cada uma destas abordagens.

Este protocolo divide a topologia em zonas e procurar utilizar diferentes protocolos de encaminhamento dentro e fora das zonas com base nos pontos fortes e fracos destes protocolos. O protocolo ZRP é modular, ou seja, poderá ser usado qualquer protocolo de encaminhamento dentro (Intra-zone routing protocol - IARP) e fora (inter-zone routing protocol - IERP) das zonas. O seu

funcionamento consiste em definir para cada nó uma zona. Dentro da zona o encaminhamento é realizado através de um protocolo proactivo, esta abordagem mantém as rotas actualizadas para qualquer nó dentro da zona, deste modo não ocorrerá um atraso inicial ao transmitir para um desses nós. Fora da zona utiliza-se um protocolo reactivo. A vantagem deste protocolo é que elimina o problema dos protocolos proactivos, que causam uma sobrecarga na rede com a necessidade de transmitir periodicamente mensagens de controlo em toda a rede, para as actualizações das rotas.

2.5. Protocolos de encaminhamento com múltiplos caminhos

2.5.1. Ad hoc On-Demand Multipath Distance Vector (AOMDV)

O protocolo Ad hoc On-Demand Multipath Distance Vector (AOMDV) [18], é um protocolo de encaminhamento reactivo de múltiplos caminhos. Este protocolo é baseado no protocolo Ad hoc On-Demand Distance Vector (AODV) [13], com o acréscimo de funcionalidades que permitem a descoberta e manutenção de múltiplos caminhos entre a fonte e o destino.

O processo de descoberta de rota do protocolo AOMDV permite descobrir vários caminhos de ligações disjuntas (link disjoint) entre a fonte e o destino. Mas também apresenta possíveis alterações ao protocolo para encontrar múltiplos caminhos de nós disjuntos (node disjoint).

Este protocolo apresenta soluções para um encaminhamento sem ciclos e com uma baixa sobrecarga na rede. Assim como encontrar múltiplos caminhos de ligações disjuntas apenas com a informação contida nas tabelas de encaminhamento, ou seja, não necessita de utilizar um encaminhamento source routing.

Para evitar ciclos no encaminhamento, o protocolo usa os campos sequence number do destino e o advertised hop count contidos na tabela de encaminhamento. O sequence number do destino vai conter sempre o maior sequence number que o nó conhece do destino. No caso de o nó receber uma mensagem de actualização de uma rota mais recente para o destino, este campo é actualizado com o sequence number do destino contido na mensagem recebida. O campo advertised hop count contido na tabela é definido com o maior número de saltos verificado entre os caminhos encontrados para determinado destino, apenas será atribuído um valor a esse campo quando o nó responde a um pedido de rota. As múltiplas rotas armazenadas na tabela vão conter sempre o mesmo sequence number do

destino, para isto, sempre que o sequence number do destino é actualizado, as entradas obsoletas são removidas, assim como o campo advertised hop count é inicializado.

O protocolo AOMDV para possibilitar encontrar múltiplos caminhos de ligações disjuntas, durante o processo de descoberta de rota verifica se o próximo salto (next hop) e o último salto (last hop) para o destino em cada um dos caminhos encontrados são distintos. A Figura 2.5 ilustra um exemplo dos múltiplos caminhos de ligações disjuntas (fonte-A-B-D-destino e fonte-C-B-E-destino), após o processo de descoberta de rota, entre a fonte e o destino.

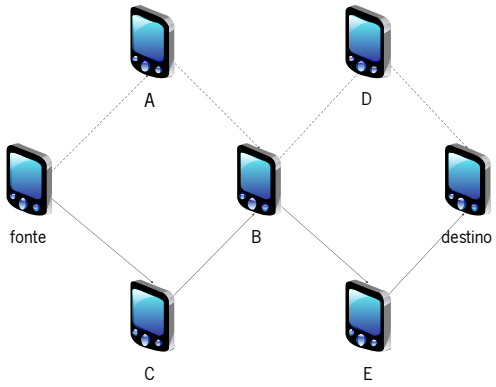


Figura 2.5: Exemplo de múltiplos caminhos de ligações disjuntas

2.6. Encaminhamento com QoS

Os protocolos de encaminhamento normalmente seleccionam o caminho de custo mínimo desde o nó fonte até ao nó destino, tendo como métrica utilizada simplesmente o número de saltos. Este tipo de protocolos são adequados ao paradigma de best-effort, mas quando se pretende utilizar aplicações mais exigentes (streaming de conteúdos multimédia, áudio ou vídeo; aplicações tempo-real, etc), que impõe determinados requisitos (largura de banda, atraso fim-a-fim, jitter, taxa de perda de pacotes, etc), esta abordagem não é adequada, visto que, o caminho encontrado por ser o mais curto poderá não satisfazer os requisitos de QoS solicitados, e podem existir outros caminhos com um número de saltos superior que possam satisfazer esses requisitos. É necessário assim dotar a rede com a capacidade de fornecer qualidade de serviço que cumpra os requisitos exigidos por este tipo de aplicações.

Nas redes móveis ad hoc, garantir um encaminhamento com QoS é sempre um desafio, ao contrário das redes fixas, neste tipo de redes a topologia de rede por ser dinâmica, com a constante

movimentação dos nós, entrada e saída de nós na rede, provoca quebras constantes nas ligações, o que conduz a alterações inesperadas das rotas. Por estes factores descritos, neste tipo de redes, verifica-se uma enorme dificuldade de obter a informação precisa do estado actual das ligações.

Neste sentido, neste tipo de redes, o conceito de QoS no encaminhamento é muitas vezes referido como Soft QoS Routing, em que as garantias fornecidas ao requisito de QoS solicitado por determinada aplicação para determinado fluxo, poderá não ser cumprido durante a transmissão total do fluxo. O protocolo de encaminhamento com QoS é responsável por encontrar um caminho exequível para as características requisitadas pela aplicação, tendo como principal objectivo influenciar nas decisões de encaminhamento tendo em consideração esses requisitos, mas durante a transmissão de um fluxo, em um determinado intervalo de tempo poderá não conseguir dar essas garantias. Isto porque, como mencionado anteriormente, a informação da rota utilizada, poderá não estar actualizada naquele instante de tempo, devido à mobilidade dos nós existente neste tipo de redes, provocando alterações contantes nas rotas, desta forma poderá ser impraticável obter a informação precisa do estado actual das ligações. Na tentativa de obter o estado das ligações o mais actual preciso, poderá conduzir a um aumento do custo computacional nos nós e de um aumento tráfego de pacotes de controlo na rede. Por isso, é conveniente ter sempre em consciência da frequência com que é realizada a actualização, sendo que uma má opção, poderá provocar uma sobrecarga na rede, degradando muito a performance do encaminhamento ou obter valores muito desactualizados.

A complexidade existente nos protocolos de encaminhamento com QoS difere conforme o tipo de métricas utilizadas para obter caminhos exequíveis para determinado destino ou para tomar decisões que influenciem no encaminhamento. Os protocolos de encaminhamento tradicionais seleccionam o caminho de custo mínimo desde uma fonte até ao destino, sendo o custo uma função e um número bastante limitado de parâmetros muitas vezes tão simples quanto o número de saltos. O caminho encontrado, embora possa ser o que tem um menor número de saltos para o destino, poderá não satisfazer os requisitos de QoS, e existirem caminhos alternativos que por sua vez os satisfazem. Este modelo adequa-se ao paradigma best-effort, mas é desadequado quando se pretende dotar a rede da capacidade de fornecer QoS. Nos protocolos de encaminhamento com QoS, normalmente são utilizadas métricas como: a largura de banda disponível, o atraso fim a fim, as perdas e o jitter. Estas métricas podem ser classificadas em três categorias: aditivas, multiplicativas e

côncavas.

Se considerarmos m(a,b) como a métrica relativa à ligação entre o nó a e o nó b, e o caminho C utilizado será: a-b-c-d-e-f.

Uma métrica será aditiva na seguinte condição: $m(C) = m(a,b) + m(b,c) + \cdots + m(e,f)$; como exemplo de uma métrica aditiva, pode-se destacar o atraso fim a fim, que corresponde à soma dos vários atrasos verificados durante o caminho realizado entre o nó fonte e o destino.

Uma métrica será multiplicativa na seguinte condição: $m(C) = m(a,b) \times m(b,c) \times \ldots \times m(e,f)$;

Uma métrica será côncava na seguinte condição: $m(C) = \min\{m(a,b), m(b,c), \ldots, m(e,f)\}$; como exemplo de uma métrica côncava, pode-se destacar a largura de banda, que para determinado caminho o calculo realizado para se obter a largura de banda disponível, corresponderá ao valor mínimo das larguras de banda disponíveis em cada uma das ligações pertencentes a esse caminho.

Na utilização de uma combinação de métricas no protocolo de encaminhamento, deve-se ter em conta que sejam ortogonais entre si, para que não haja informação redundante entre as métricas.

Wang e Crowcroft provam que o problema de encontrar um caminho que tenha em conta duas ou mais métricas aditivas ou multiplicavas independentes entre si, qualquer tipo de combinação utilizada, será um problema NP-Completo [19].

Os requisitos de QoS variam de acordo com o tipo de aplicação, poderá ser tráfego mais ou menos exigente, por exemplo, Voz sobre IP (VOIP) será mais exigente que o *streaming* de conteúdos multimédia, neste sentido, o protocolo de encaminhamento com QoS tem como principal objectivo, influenciar nas decisões de encaminhamento, conforme os requisitos de QoS da aplicação. Na secção 2.6.1 e na 2.6.2, serão descritas algumas ideias para a concepção de protocolos de encaminhamento, que possibilitam influenciar de forma diferente no encaminhamento.

2.6.1. Modelos de QoS

Nos últimos anos, nas redes de computadores quando se pretende dotar a rede de garantias de qualidade de serviço, costumam ser utilizados principalmente dois modelos, classificados em dois tipos, Serviços Integrados (IntServ) [20] e Serviços Diferenciados (DiffServ) [21].

O modelo IntServ consiste em fornecer reservas explícitas por fluxo através do protocolo RSVP.

No pedido de reserva são indicados os requisitos de QoS pretendidos. Caso seja possível cumprir as condições indicadas é efectuada a reserva, se não for possível, a reserva será negada.

O modelo DiffServ oferece um tratamento diferenciado de pacotes de acordo com a classe de serviço a que pertencem, sendo necessário a marcação dos pacotes para ficarem associados a uma determinada classe de serviço. Cada classe possui os seus próprios requisitos de QoS e os pacotes recebem um tratamento diferenciado em cada nó, mediante a classe a que pertencem (por exemplo através da criação de diferentes filas de espera por cada classe de serviço).

O modelo IntServ não é muito adequado aplica-lo directamente nas redes móveis ad hoc [22] [12], devido aos problemas bem conhecidos de escalabilidade, com o aumento do número de fluxos a necessidade de manter toda a informação de estado nos elementos da rede referente a cada fluxo torna o modelo pouco escalável. O mecanismo de reserva é efectuado através do protocolo RSVP, este consiste no envio de mensagens de sinalização de pedido de reserva (PATH) por parte do nó fonte para determinado destino, e posteriormente caso o destino pretenda receber os dados do nó fonte, procede ao envio de uma mensagem de confirmação (RESV). Durante o envio da mensagem RESV que percorre o caminho inverso utilizado pela mensagem PATH, é verificado se é possível efectuar a reserva dos recursos requisitados no caminho até ao nó fonte, sendo a decisão realizada nó a nó, caso os recursos disponíveis forem suficientes a reserva é efectuada em cada nó e a mensagem RESV é retransmitida no sentido do nó fonte, sempre que isto se verifique nó a nó os recursos ficam reservados no caminho entre o nó fonte e o destino. Este mecanismo de reserva com troca de mensagens nos dois sentidos, entre nó fonte e o destino, e o armazenamento da informação necessário nó a nó para cada fluxo, torna o processo lento e inadequado para se adaptar este modelo às redes móveis ad hoc, devido à sua natureza altamente dinâmica, com alterações frequentes nos caminhos existentes entre dois nós, tornando assim, os recursos reservados inutilizáveis e uma sobrecarga desnecessária de tráfego de controlo, para se efectuar as reservas e para libertar os recursos reservados.

No modelo DiffServ como descrito, os fluxos são agregados em classes de serviço (CoS). Os pacotes são marcados com um DiffServ Code Point (DSCP) incluído no campo Type of Service (ToS) do cabeçalho IP. Desta forma ficam associados a determinada classe de serviço, recebendo em cada nó um tratamento diferenciado de acordo com a classe de serviço a que pertencem. Este modelo foi concebido para colmatar os problemas de escalabilidade existente no modelo IntServ. No entanto,

adaptar este modelo directamente nas redes móveis ad hoc também não é praticável, visto que este tipo de redes não utiliza a noção existente de alguns tipos de nós utilizados neste modelo (nós Ingress, interior e egress), assim como a noção de Service Level Agreement (SLA). No modelo DiffServ os nós de fronteira são responsáveis pela marcação dos pacotes com um DSCP de acordo com o SLA, sendo o tratamento diferenciado efectuado nos nós interior. Este conceito como referido não existe nas redes móveis ad hoc, caso se pretende adoptar este modelo a este tipo de redes, cada nó terá de desempenhar o papel existente dos nós fronteira assim como o de nó interior, responsável pela marcação dos pacotes com o DSCP apropriado de acordo com os requisitos da aplicação. Desta forma caso um nó pretenda transmitir terá de se comportar como um nó fronteira, e no caso de ser um nó intermédio que tem de encaminhar os pacotes para o destino ou o próprio nó destino, terá de se comportar como um nó interior.

Parte da investigação actual dos protocolos de encaminhamento com QoS nas redes móveis ad hoc, consiste em analisar alguns protocolos de encaminhamento existentes e tentar melhora-los ou aproveitar algumas noções neles existentes e modifica-los, de forma a obterem um melhor desempenho e fornecer suporte de QoS.

As soluções propostas para dar garantias de QoS neste tipo de redes passam muitas vezes por soluções diferentes dos modelos IntServ e DiffServ. Existem propostas de soluções de QoS que utilizam alguns conceitos existentes nestes modelos descritos, enquanto outras propostas optam por tentar dar garantias de QoS através de diferentes abordagens, devido à dificuldade de aplicar estes modelos directamente nas redes móveis ad hoc.

Neste capítulo serão descritos alguns modelos e protocolos de encaminhamento com QoS concebidos para as redes móveis ad hoc.

2.6.2. Concepção de protocolos de encaminhamento com QoS

Inicialmente os protocolos de encaminhamento concebidos nas MANETs, focaram-se essencialmente na descoberta de rotas entre os nós e na sua manutenção, apenas para encontrar uma rota entre o nó fonte e o nó destino, sem qualquer preocupação em optimizar a utilização dos recursos da rede ou tentar suportar requisitos específicos da aplicação.

Para fornecer encaminhamento com QoS nestas redes, a principal dificuldade encontra-se na descoberta de uma rota que possua recursos disponíveis suficientes, de forma a satisfazer os

requisitos de QoS solicitados e se possível incorporar optimizações, ou seja, as rotas obtidas no processo de descoberta encontram-se entre as melhores de todas as rotas que cumpriam os requisitos de QoS solicitados.

Para conceber um protocolo de encaminhamento com QoS, tem de se ter em consideração alguns dos mecanismos mais importantes, tais como:

- Descoberta da rota: Nas redes móveis ad hoc, os protocolos de encaminhamento são normalmente classificados em duas categorias de acordo com a sua estratégia de encaminhamento. Os protocolos reactivos e os proactivos. A maior parte dos estudos realizados, defendem que os protocolos reactivos são mais eficientes comparativamente aos proactivos, por não causarem tanto *overhead* na rede, visto que o processo de descoberta de rota nestes protocolos, apenas é iniciado quando um nó pretende transmitir e necessita de conhecer uma rota para o destino. Enquanto os protocolos de encaminhamento proactivos, geram constantemente tráfego de controlo na rede, para tentar manter a informação actualizada de rotas para cada nó da rede. Desta forma, os dispositivos despendem mais energia e requerem mais poder de processamento, que pode ser um problema para muitos dispositivos que utilizam estas redes, visto possuírem características limitadas, processadores com pouca capacidade computacional e são alimentados a bateria.

  Em contrapartida, os protocolos de encaminhamento reactivos, demoram mais tempo na descoberta da rota que os proactivos, visto que nos protocolos proactivos os nós da rede ao trocarem mensagens de controlo periodicamente, mantêm a sua tabela de encaminhamento actualizada, desta forma, quando um nó pretende transmitir, já contém a rota para o destino pretendido, não existe portanto um atraso acrescido na procura de rotas, como sucede nos protocolos de encaminhamento reactivos.

- Manutenção da rota: Nas redes móveis ad hoc devido à mobilidade dos nós, ocorrem constantes alterações na topologia da rede, torna-se assim muitas vezes difícil satisfazer os requisitos de QoS solicitados, visto que uma rota obtida durante o processo de descoberta que cumpra os requisitos de QoS, a qualquer momento pode deixar de existir.

  Neste sentido, incorporar um mecanismo de manutenção de rotas torna-se importante. Existem várias abordagens para a manutenção das rotas, uma delas, consiste após a ocorrência da falha,

esperar que um nó a detecte e envie uma notificação ao nó fonte, de forma a informar o nó que ficou indisponível. Esta abordagem afecta a performance no encaminhamento, visto que uma mensagem tem de percorrer o caminho desde o nó que detecta a falha até ao nó fonte, sendo depois inicializado novamente o processo de descoberta de rota. Isto implica um grande atraso na transmissão, um aspecto muitas vezes relevante que não suceda, principalmente quando se refere a aplicações em tempo-real, visto ter como restrição, o atraso na propagação dos pacotes da fonte até ao destino.

Na tentativa de tornar mais eficiente a manutenção das rotas, são aplicadas em alguns protocolos concebidos técnicas para reduzir este atraso e reduzir a sobrecarga provocada na rede, com o envio de mais pacotes de controlo. Algumas das técnicas aplicadas são a descoberta de rotas para o destino por parte dos nós intermédios que detectam a falha, mecanismos de previsão [4], ou manter rotas alternativas, efectuando a comutação da rota activa para uma secundária quando é detectado um problema na rota activa [23] [24] [25] [26].

- Estimativa dos recursos: Uma estimativa dos recursos disponíveis existentes é essencial para conseguir obter rotas que satisfaçam certos requisitos. O principal objectivo, será receber informação dos recursos disponíveis de camadas inferiores, essencial para efectuar um controlo de admissão e um modelo de QoS adaptativo. A estimativa de recursos é efectuada muitas vezes em relação à largura de banda ou ao atraso fim a fim, visto serem requisitos de QoS solicitados actualmente por diversas aplicações.

  Nas redes ad hoc, os nós da rede partilham a largura de banda com os nós vizinhos. Deste modo, a largura de banda disponível para um nó pode sofrer constantes alterações, visto ser afectada através do tráfego gerado nos nós vizinhos. Para oferecer garantias de largura de banda ou encaminhamento com garantias de atraso fim-a-fim, a estimativa destes recursos é essencial, no entanto efectuar em cada nó da rede uma estimativa com precisão destes recursos é de complexa execução. Isto porque os requisitos de QoS são muitas vezes métricas dinâmicas, como a largura de banda disponível ou o atraso fim-a-fim, tendo constantes alterações consoante o tráfego gerado nos nós da rede e a mobilidade dos nós, torna-se assim muito complexo efectuar uma estimativa actualizada.

  Na estimativa da largura de banda disponível, existem dois aspectos principais, um consiste em determinar como a estimativa se realizará, e outro, com que frequência ocorre.

Quando se pretende efectuar uma estimativa de um recurso, é conveniente então, ter sempre consciência dos contras que uma estimação mal realizada poderá trazer. Neste sentido, a selecção ou concepção do tipo de estimativa a realizar e a frequência com que é realizada é importante, sendo que uma má opção pode provocar sobrecarga na rede, degradando muito a performance do encaminhamento ou obter valores desactualizados.

Se as estimativas forem realizadas com pouca frequência, em que os valores obtidos já se encontram muito desactualizados, a performance do encaminhamento com QoS é muito penalizado como referido. No entanto, na tentativa de se obter estimativas actualizadas, ao verificar com elevada frequência o estado actual da rede, pode-se prejudicar o encaminhamento, devido ao facto de se estar a gerar muito tráfego de controlo na rede para obter essa informação, visto provocar uma sobrecarga na rede. Torna-se assim uma decisão importante determinar como, e com que frequência se pretende realizar a estimativa dos recursos.

Alguns dos protocolos de encaminhamento com QoS concebidos assumem que a largura de banda disponível é conhecida, presumem que a informação é recebida pelas camadas inferiores e focam-se apenas na concepção do protocolo de encaminhamento com QoS, como é o caso dos protocolos CEDAR [8], Enhanced Ticket-based QoS Routing [27], ADQR [23] e TDR [28].

Outros protocolos concebidos descrevem possíveis soluções para a sua estimação, onde se verificam distintas execuções neste procedimento. No caso do protocolo Quality of service routing in ad-hoc networks using OLSR [29] propõe a exploração do IEEE 802.11 ao escutar o meio efectuar a medição entre o tempo inactivo e o ocupado. O protocolo AQOR [30] baseia-se no envio da informação da largura de banda consumida entre os nós vizinhos através das mensagens Hello, sendo realizada uma estimativa da largura de banda disponível em cada nó com base nessa informação.

- Reserva dos recursos: Sendo a largura de banda de cada nó nestas redes partilhada entre os nós vizinhos, torna-se assim um desafio, efectuar a reserva deste recurso. A escolha no tipo de mecanismo a implementar para efectuar a reserva do recurso e o tipo de controlo de admissão, são importantes na descoberta e manutenção de rotas que satisfaçam este requisito de QoS.
- Selecção da rota: Ao seleccionar uma rota pretende-se que seja o mais estável possível, visto que o encaminhamento com QoS tem requisitos exigentes, e a falha das rotas afecta muito a qualidade

de serviço fim-a-fim. Desta forma, alguns dos protocolos de encaminhamento com QoS existentes, preferem não utilizar apenas por exemplo, a largura de banda ou atraso fim-a-fim, mas ter em conta, várias métricas na selecção da rota que a tornem mais estável, ou seja, tentar seleccionar uma rota com maior probabilidade de se manter activa por mais tempo. Obter rotas que cumpram múltiplos requisitos, pode trazer problemas na execução do encaminhamento, uma vez que, encontrar uma rota que satisfaça dois requisitos independentes entre si é um problema NP-Completo.

2.7. Modelos e Protocolos de Encaminhamento com QoS em Redes Móveis Ad Hoc

2.7.1. Core-Extraction Distributed Ad hoc Routing (CEDAR)

O Core-Extraction Distributed Ad hoc Routing (CEDAR) [8] é um protocolo de encaminhamento que define dinamicamente um conjunto de nós para realizarem a descoberta e manutenção de rotas, oferecer garantias de QoS e para o encaminhamento dos dados.

O protocolo de encaminhamento CEDAR é um protocolo que elege dinamicamente e de forma distribuída, alguns nós da rede designados de core nodes. Cada um destes nós eleitos como core nodes, mantêm uma pequena topologia local. Deste modo formam-se vários grupos de nós na rede, sendo que a propagação do estado das ligações estáveis com grande largura de banda disponível será realizada entre todos os nós eleitos (core nodes), enquanto a informação do estado das restantes ligações (com baixa largura de banda disponível) ou ligações muito instáveis é mantida localmente (apenas entre os nós que contêm o mesmo core node). Este protocolo usa esta estratégia para reduzir a sobrecarga na rede.

A descoberta de rota entre a fonte e o destino consiste numa fase inicial em estabelecer um caminho entre o nó eleito (core node) da fonte com o nó eleito do destino. O requisito da largura de banda mínima é efectuado pela aplicação. Sendo o objectivo do protocolo encontrar uma rota estável que possa satisfazer esse requisito. O cálculo da estimação da largura de banda disponível não é explicada pelos autores do protocolo, apenas assumem que se pode estimar a largura de banda disponível na camada MAC e assumem uma interacção próxima entre a camada MAC e a camada de rede.

A Figura 2.6 ilustra a mesma topologia em três sequências distintas do processo de encaminhamento de dados entre uma fonte e o destino. Os nós eleitos (core nodes) estão representados a cor vermelha, com as ligações a tracejado que correspondem às rotas conhecidas inicialmente do nó A, isto é, apenas conhece as rotas para os nós locais que o identificaram como core node e os restantes core nodes (nó E e G). Para a transmissão de dados da fonte (nó F) para o destino (nó D), estabelece-se inicialmente o caminho entre o core node da fonte (nó A) e o core node do destino (nó G). Na última sequência de imagens pode-se observar o reenvio dos pacotes do nó G (core node do destino) para o destino (nó D). A principal desvantagem nesta abordagem é o congestionamento e a ocupação da largura de banda nas ligações entre os core nodes.

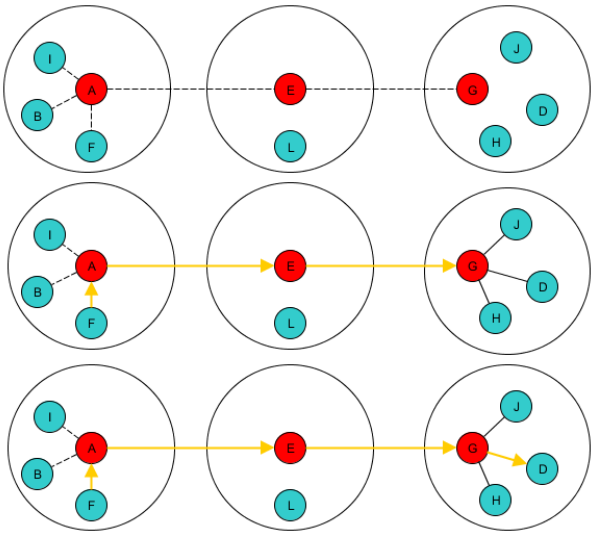


Figura 2.6: Exemplo encaminhamento no protocolo CEDAR

2.7.2. Adaptive QoS Routing (ADQR)

Y Hwang and P varshney propõem o protocolo designado por An Adaptive QoS Routing Protocol with Dispersity for Ad-hoc Networks (ADQR) [23], que consiste num algoritmo de descoberta de rotas que permite encontrar vários caminhos disjuntos.

Com base nas informações da largura de banda obtidas durante a descoberta de rotas, este protocolo procede à reserva de recursos nestes caminhos. A transmissão de dados é efectuada em todos os caminhos reservados. O protocolo ADQR monitoriza a rede na tentativa de detectar mudanças na topologia e com o objectivo de actualizar as rotas antes que as mesmas fiquem indisponíveis. Com os processos de descoberta e manutenção de rotas utilizados no protocolo, o ADQR procura melhorar

significativamente o desempenho da rede e dar suporte de QoS fim a fim em redes ad-hoc.

A principal desvantagem encontrada é o facto de não resolver o problema da reordenação dos pacotes, inerente ao balanceamento do tráfego pelos vários caminhos. Além disso para processar o pedido de rota, o mecanismo de encaminhamento proposto, necessita de armazenar em cada nó a informações de estado da topologia da rede. Por outro lado a monitorização proactiva necessária para o seu funcionamento dá origem a uma sobrecarga de pacotes de controlo na rede, que pode ser excessiva.

2.7.3. Delay-aware Multipath Source Routing Protocol (DMSR)

Shun Liu e Jian Liu propõem o protocolo Delay-aware Multipath Source Routing (DMSR) [31].

Este é um protocolo de encaminhamento de múltiplos caminhos, que utiliza o atraso fim a fim para fornecer suporte de QoS a aplicações multimédia de tempo real em redes ad hoc. O protocolo obtém a informação local e realiza o cálculo do atraso verificado no nó, utilizando-o como métrica para selecção do caminho. A métrica tem em conta o número de nós vizinhos dos nós de encaminhamento, o tempo de contenção e o número de pacotes na fila de espera.

O protocolo DMSR pretende desta forma reduzir o atraso fim a fim e atender aos requisitos de serviço de aplicações multimédia de tempo real.

2.7.4. Ad Hoc QoS On-demand Routing (AQOR)

O protocolo Ad hoc Qos on-demand routing (AQOR) [30], é um protocolo de encaminhamento com qualidade de serviço (QoS), que tem a capacidade de encontrar caminhos entre o nó fonte e o destino que cumpram os requisitos QoS, nomeadamente o atraso fim a fim e a largura de banda.

É um protocolo baseado na reserva de recursos, sendo que a reserva da largura de banda requisitada na aplicação para um determinado fluxo, é verificada nó a nó por um mecanismo de controlo de admissão, após encontradas rotas que cumpram os requisitos de QoS, é seleccionada no nó fonte uma rota entre as possíveis encontradas que cumprem os requisitos de QoS, e apenas nessa rota elegida a reserva será concretizada.

O AQOR é um protocolo reactivo, logo quando a fonte necessita de uma rota para o destino, a fonte inicia o processo de descoberta de rota, com o envio em broadcast de um pacote route request (RREQ), que contém a informação dos recursos requisitados pela aplicação, nomeadamente o atraso

fim a fim pretendido (Tmax) e a largura de banda mínima necessária (Bmin). Os nós intermédios ao receber um pacote RREQ, analisam a sua largura de banda disponível e verificam através da mensagem RREQ recebida se as condições exigidas podem ser cumpridas. No caso de o fluxo ser admitido no nó, é criado um registo na tabela de encaminhamento com um intervalo tempo de validação (2Tmax) até à recepção de um pacote route reply (RREP) e retransmite o pacote RREQ. No caso de o nó não receber um pacote RREP nesse período de tempo o registo será removido, sendo que os pacotes RREP recebidos após esse período de tempo serão descartados.

Durante o envio dos pacotes RREP do nó destino para a fonte utilizando o percurso inverso das rotas encontradas, será feita a validação da rota nos nós intermédios, após verificar novamente se a largura de banda disponível é suficiente para cumprir o requisito QoS.

Com a recepção dos vários RREP no nó fonte, entre todas as rotas que cumpriram os requisitos solicitados, será seleccionada a rota com o menor atraso fim a fim.

Com o início de transmissão do nó fonte para o nó destino, encaminhando os pacotes pela rota seleccionada, quando o primeiro pacote transmitido for recebido nos vários nós intermédios que constituem a rota, será realizado nó a nó a reserva da largura de banda para o fluxo.

Durante o processo de descoberta de rota, os nós intermédios ao enviar o pacote RREP para a fonte é definido um intervalo de tempo de validação (2Tmax) dessa mesma rota até receber a confirmação (primeiro pacote transmitido no nó fonte) que essa rota foi a elegida no nó fonte, neste sentido exceptuando a rota seleccionada no nó fonte para transmissão do fluxo de pacotes para o destino, as restantes rotas encontradas vão expirar este período de tempo, desta forma os nós intermédios pertencentes a cada uma dessas rotas vão proceder à eliminação do seu registo contido na tabela.

Após a reserva da largura de banda para um determinado fluxo, no caso de um nó intermédio pertencente à rota onde foi realizada a reserva não receber pacotes durante um determinado intervalo de tempo, o nó invalida a reserva efectuada. Desta forma evita-se a necessidade do envio de pacotes de controlo para a rede para libertar as reservas realizadas, ficando assim os recursos disponíveis para futuras reservas, este tipo de manutenção das reservas é designado de temporary reservation.

Os autores do protocolo para realizar o controlo de admissão e a reserva dos recursos, desenvolveram uma computação exequível para efectuar a medição do atraso fim a fim e a estimação

da largura de banda disponível, assumindo como mecanismo de acesso ao meio distribuído o IEEE 802.11 DCF.

A execução da estimação da largura de banda é executada em cada nó da rede, sendo realizada com a informação que é trocada periodicamente, através de mensagens Hello entre os nós vizinhos da rede. Nestas mensagens vai conter a informação da largura de banda utilizada em cada nó, designada de self-traffic (Bself(I)). Este protocolo assume que a estimação da largura de banda disponível efectiva que se verifica num nó (Bavailable(I)), consiste na largura de banda total do canal (B) subtraindo à soma total do trafego criado por todos os nós vizinhos, Equação (1).

- Estimação da largura de banda disponível efectiva:

$$Bavailable(I) = B - \sum_{J \epsilon N(I)} Bself(J) \quad (1)$$

Cada nó da rede mantém uma tabela com a informação de todos os seus nós vizinhos e a sua respectiva largura de banda utilizada (Bself), com esta informação consegue-se determinar a largura de banda disponível através dos cálculos descritos anteriormente.

No caso de um determinado nó, não receber uma mensagem Hello num determinado período de tempo (Tlost), é porque deixou de haver uma ligação com esse nó vizinho, logo será removido da sua tabela de nós vizinhos.

Para a medição do atraso, admite-se que o atraso fim a fim verificado entre o nó fonte e o destino (Tmax), é aproximadamente metade do atraso ocorrido durante o percurso de ida e volta entre o nó fonte e o destino (round trip delay - Tround). Onde o Tround ≤ 2Tmax, e com o tempo de diferença entre o atraso upstream e downstream (Tdiff), pretende-se garantir em cada descoberta de rota que Tround + Tdiff ≤ 2Tmax.

Está também definido no protocolo, um procedimento para recuperação de rotas para detectar algum tipo de incumprimento, quer do atraso fim a fim ou pela quebra de uma ligação devido à movimentação dos nós ou por simplesmente o dispositivo deixou de ter energia suficiente para estar activo. Para detectar um possível incumprimento do atraso fim a fim após o início de transmissão por parte do nó fonte, o nó destino realiza a verificação do atraso dos pacotes recebidos, no caso de se verificar pacotes consecutivos pertencentes ao fluxo ao qual foi realizada a reserva que não cumpram o atraso fim a fim requisitado, é desencadeado o processo de recuperação de rota. Este processo

consiste no envio em broadcast de um pacote designado de route update, que basicamente é um RREQ mas no percurso inverso, ou seja, percorre do nó destino para a fonte, designado este processo de reverse exploration. Este pacote é tratado nos nós intermédios como um pacote RREQ, sendo efectuado nó a nó o controlo de admissão respectivo para que na chegada deste pacote ao nó fonte, se encontre uma rota que cumpra os requisitos solicitados. O nó fonte ao receber o primeiro pacote route update, altera entre a rota activa para esta nova rota encontrada, transmitindo os restantes pacotes pertencentes ao fluxo por este caminho.

No processo de recuperação de rotas para o caso de ser detectada uma quebra de ligação, para evitar o atraso de transmissão dos pacotes por parte do nó fonte, em vez de o nó que detecta a falha transmitir uma notificação ao nó fonte, sendo depois iniciado novamente o processo de descoberta de rota, este protocolo utiliza o tempo de expiração da reserva da largura de banda (Tinterval) existente para detectar as falhas nas ligações. No caso do nó destino deixar de receber pacotes de um determinado fluxo no qual existe uma reserva, o nó destino inicia o processo de recuperação de rota.

2.7.5. Route Stability based QoS Routing (SMQR)

Nityananda Sarma e Sukumar Nandi propõem o protocolo Route Stability based Multipath QoS Routing (SMQR) [4] para dar suporte a aplicações tempo-real nas redes ad hoc. Estas aplicações são sensíveis ao atraso fim a fim e à taxa de transferência efectiva.

Este protocolo utiliza o modelo de estabilidade da rota (Route Stability Model) existente no protocolo Route Stability based QoS Routing (RSQR) [32]. Este modelo de estabilidade de rota reside na utilização da potência do sinal recebido para verificar a estabilidade das ligações ao longo da rota. Esta informação é fornecida pela subcamada MAC pertencente à camada de ligação de dados e é utilizada pela camada de rede através de uma interacção cross-layer.

O protocolo SMQR durante o processo de descoberta de rotas selecciona no máximo até três rotas disjuntas que cumpram os requisitos de qualidade de serviço solicitados pela aplicação, e elege como rota principal a que tiver mais estabilidade, ficando as restantes rotas (caso existam) como secundárias.

As rotas secundárias são analisadas periodicamente, no caso de se verificar mais estabilidade numa rota secundária do que na rota activa, o nó fonte comuta da rota activa para a rota secundária.

O processo de descoberta de rota só é desencadeado novamente apenas quando for detectado falhas em todas as rotas seleccionadas.

A vantagem de se obter múltiplas rotas disjuntas no processo de descoberta de rotas comparativamente aos protocolos onde se obtém apenas uma única rota, como se verifica por exemplo no protocolo AODV, é na recuperação de falhas da rota activa. Desta forma com a existência de rotas alternativas, quando a rota activa deixa de estar disponível não é necessário desencadear um novo processo de descoberta de rota, o nó fonte pode utilizar logo uma rota alternativa válida, gerando assim menos tráfego de controlo na rede.

Seleccionando as rotas mais estáveis entre as rotas descobertas, aumenta a confiabilidade das rotas seleccionadas, desta forma tenta-se prever a movimentação dos nós descartando as rotas com baixa estabilidade, rotas essas que seriam mais prováveis de se tornarem impraticáveis num curto período de tempo, minimizando assim o tráfego de controlo na rede e o atraso fim a fim.

Alguns mecanismos do protocolo SQMR foram concebidos da seguinte forma:

- Estimação do Atraso fim a fim e da Largura de banda: O cálculo do atraso fim a fim é realizado efectuando a medição do atraso em cada salto, esta informação é obtida através de uma interacção *cross-layer* entre a subcamada MAC e a camada de rede.

  Um nó calcula o atraso MAC (d) verificando o tempo que um pacote demora a chegar à camada MAC (t_s) subtraindo ao tempo que o nó recebe na confirmação de que o pacote foi entregue com sucesso no receptor, através da recepção da mensagem de "*acknowledge*".

  De forma a minimizar as alterações nos atrasos calculados o protocolo SMQR opta por utilizar o estimador Exponentially Weighted Moving Average (EWMA) para efectuar a medição do atraso médio MAC ($\mathrm{d}_{\mathrm{avg}}^{\mathrm{i}}$).

  O estimador é dado pela equação:

$$d_{avg}^{i} = \eta \times d_{avg}^{i-1} + (1 - \eta) \times d^{i} \qquad (1)$$

  Na Equação (1), a variável d^i corresponde ao atraso MAC obtido, a variável d^{i-1}avg corresponde ao atraso médio MAC previamente calculado no nó e a variável η é uma constante positiva e foi definida com o valor de 0.8 durante a avaliação efectuada ao protocolo.

O protocolo SMQR utiliza as especificações do IEEE 802.11 no modo distribuído (802.11 Distributed Coordination Function), normalmente utilizado nas redes sem fios com encaminhamento de múltiplos saltos, este método de acesso utiliza o Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) na camada MAC. Inicialmente o atraso MAC num determinado nó é definido pela soma do atraso de propagação, o tempo de backoff, o intervalo DIFS e o SIFS.

A estimação da largura de banda é concretizada utilizando a solução existente do protocolo Ad hoc QoS on-demand routing (AQOR) [30]. A largura de banda disponível para um nó nas redes ad hoc, não é apenas limitada pela capacidade total do meio de comunicação, é também limitada pela interacção existente entre os nós no seu alcance de transmissão, assim como pelos nós localizados no seu raio de interferência. Cada nó faz a estimativa da largura de banda disponível $(\mathrm{BW_{avail}})$ subtraindo a largura de banda máxima de transmissão do nó (BW) pelo somatório da largura de banda consumida pelos nós no seu raio de interferência. O cálculo é realizado pela seguinte equação:

$$BW_{avail}(I) = BW - \sum_{j \epsilon N(I)} BW_{self}(j) \qquad (2)$$

A informação da largura de banda reservada está contida na tabela designada de Flow Table em cada nó, e é trocada esta informação periodicamente com os nós vizinhos através das mensagens *Hello*. É com esta informação que um nó ao receber um pacote QRREQ verifica se o *throughput* mínimo requisitado pode ser admitido.

- Descoberta da rota: O processo de descoberta de rotas foi baseado no protocolo reactivo Ad Hoc On-Demand Distance Vector (AODV) [13], foram efectuadas alterações ao protocolo de forma a descobrir múltiplas rotas disjuntas que cumpram os requisitos de QoS solicitados pela aplicação entre o nó fonte e o nó destino.

  O protocolo SMQR assim como o protocolo reactivo AODV quando o nó fonte pretende transmitir para determinado destino e não está nenhuma rota activa em cache, o processo de descoberta de rota é desencadeado através do envio em *broadcast* de um pacote de pedido de descoberta de rota (Route Request – RREQ) de forma a tentar obter uma rota válida para o destino. No caso do protocolo SMQR este pacote foi alterado de forma a descobrir rotas que cumpram determinados requisitos de QoS, contrariamente ao protocolo AODV que é um protocolo *best effort*. O pacote

RREQ no protocolo SMQR foi designado de QoS Route Request (QRREQ) em que é acrescentado ao pacote a informação dos recursos requisitados pela aplicação, a informação do atraso fim a fim máximo pretendido ($D_{máx}$) e da taxa de transferência efectiva mínima (B_{min}).

É também acrescentado ao pacote QRREQ a informação do atraso acumulado (ADELAY) do percurso, a informação da estabilidade acumulada do percurso (APS) e o campo seq no para identificar unicamente a rota.

No pacote QRREQ desde que este é emitido pelo nó fonte vai sendo guardada a informação de todos os nós que o pacote vai percorrendo até ao destino, como acontece no protocolo DSR. Com esta informação completa do percurso entre a fonte e o nó destino, é possível determinar pelo nó destino quais as rotas disjuntas que foram obtidas durante o processo de descoberta de rota.

Além desta informação o pacote QRREQ antes de ser emitido pelo nó fonte, para desencadear o processo de descoberta de rota, vai conter a informação dos endereços dos nós vizinhos da fonte.

Sempre que o pacote QRREQ é recebido num determinado nó é incrementado o campo hop count, para guardar o número de saltos efectuados desde o nó fonte até ao nó actual. É armazenado também o percurso acumulado (APATH), contendo os IDs dos nós desde o início que é transmitido o pacote pelo nó fonte até chegar ao nó destino, desta forma será possível realizar pelo nó destino o cálculo das rotas disjuntas que cumpriram os requisitos QoS solicitados pela aplicação.

Quando um nó recebe um pacote QRREP a potência do sinal é analisada, no caso de ser inferior ao *threshold* estipulado o pacote é descartado, evitando assim a selecção de rotas com ligações com uma estabilidade baixa. Esta verificação da potência do sinal recebido, assim como do *throughput* é efectuada durante o processo de descoberta de rota pelo controlo de admissão em cada nó, para garantir que a qualidade de serviço requisitada pela aplicação é cumprida a cada salto durante o percurso completo até ao destino.

No processo de descoberta de rota os pacotes QRREP duplicados são analisados em cada nó limitando a sua retransmissão. Na primeira vez que um nó intermédio recebe um pacote QRREP para a descoberta de rota entre um nó fonte e o nó destino é armazenado na Reverse Routing Table (RRT) no campo Advertised hop count, a informação de que o pacote foi retransmitido e o número de saltos efectuados. Através desta informação é limitado o número de pacotes duplicados que são retransmitidos, diminui-se assim quantidade de pacotes de controlo na rede durante o

processo de descoberta, assim como resolve o problema de ciclos e previne que os pacotes QRREQ percorram na direcção inversa. Cada registo na RRT é armazenado por um período de tempo de $3 \times D_{máx}$. Se não for recebido nenhum pacote Qos Route Reply (QRREP) durante este período de tempo, será removido o registo armazenado na tabela RRT, assim como a informação do pedido de rota armazenada temporariamente na tabela Route Request Forwarded Table (RFT).

A análise realizada por um nó intermédio ao receber um pacote QRREQ duplicado para averiguar se este será retransmitido, consistirá em verificar se o número de saltos contido no pacote QRREQ é igual ou inferior ao contido na RRT, caso isto se verifique e se o pacote foi retransmitido por um nó vizinho da fonte diferente ou se foi retransmitido pelo mesmo vizinho contido no QRREQ anteriormente recebido mas este contém informação de mais estabilidade nas ligações, o pacote será retransmitido pelo nó intermédio, caso contrário será descartado. Parte desta informação está contida na RFT, esta tabela irá conter a informação referente ao nó vizinho e correspondente valor de estabilidade da rota para os pacotes QRREQ já retransmitidos.

Durante a descoberta de rota foi acrescentada uma regra especial na retransmissão do pacote QRREQ duplicado, caso este nó intermédio que está a analisar o pacote seja vizinho do nó destino, se for verificado que o pacote foi retransmitido por um nó vizinho da fonte diferente do que está armazenado, mesmo contendo um número de saltos maior que o valor contido no campo advertised hop da tabela RRT para o mesmo pacote QRREQ, este será retransmitido para o nó destino. Desta forma obtém-se mais uma rota disjunta sem aumentar muito o tráfego de controlo na rede.

Vai ser descrito de seguida o processo de descoberta de rota do protocolo SMQR com esta regra especial com a ajuda da Figura 2, que representa os nós constituintes da rede, e o nó S inicia o processo de descoberta de rota para o destino nó D.

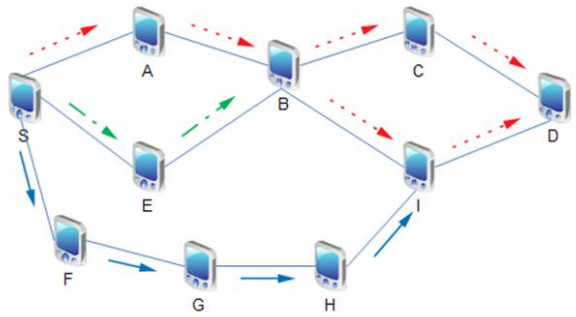


Figura 2.7: Processo de descoberta de rota com a regra especial aplicada no nó I

Supondo que o nó S inicia o processo de descoberta de rota para o nó D e o pacote QRREQ transmitido em *broadcast* por S chega ao nó I pelo caminho S-A-B-I primeiro que o pacote QRREQ que percorre o caminho S-F-G-H-I. Com a regra especial, o nó I ao receber o pacote QRREQ duplicado transmitido pelo nó H, tendo sido este pacote retransmitido por um nó vizinho da fonte diferente do anterior (nó F em vez do nó A), o nó I não vai descartar este QRREQ duplicado mesmo tendo um número de saltos superior ao pacote QRREQ recebido pelo nó B, uma vez que o nó I é vizinho do nó destino (nó D), respeitando assim a regra estabelecida. Desta forma o nó destino obtém mais uma rota possível para o cálculo das múltiplas rotas disjuntas, podendo obter no fim as rotas disjuntas S-A-B-C-D e S-F-G-H-I-D ou as rotas S-E-B-C-D e S-F-G-H-I-D.

- Selecção da rota: O nó destino após receber o primeiro pacote QRREQ, cujos requisitos QoS passem pelo controlo de admissão, espera um curto período de tempo designado de Route Reply Latency (RRL) para que receba os restantes pacotes QRREQ.

  Entre os pacotes recebidos pelo nó destino, vão ser seleccionadas apenas rotas disjuntas. Por cada ligação existente ao nó destino, apenas armazena um pacote QRREQ retransmitido por cada nó vizinho da fonte, será seleccionado o que possuir a informação de mais estabilidade na rota. Após este processo os pacotes QRREP armazenados serão colocados por ordem de estabilidade e será executado o algoritmo para seleccionar até três rotas disjuntas.

  Após estarem seleccionadas as rotas disjuntas, o nó destino por cada rota encontrada envia um pacote QRREP pelo percurso inverso contido no pacote QRREQ correspondente para a fonte. Durante este percurso sempre que um nó intermédio recebe o pacote QRREP realiza o controlo de admissão, cria entradas na tabela de encaminhamento para o nó destino e para o nó fonte e faz a reserva de largura de banda requisitada pela aplicação para o fluxo, introduzindo esta

informação na tabela designada de Flow Table. O nó fonte ao receber o pacote QRREP também acrescenta na sua tabela de encaminhamento as entradas correspondentes, diferenciando a rota com mais estabilidade, que ficará definida como rota principal das restantes duas rotas (caso existam) como secundárias.

- Manutenção da rota: O protocolo SMQR após admitir o tráfego na rota principal faz a manutenção das rotas para que continuem a garantir a qualidade serviço requisitada. O protocolo abordou o mecanismo de manutenção da rota em duas partes: uma será verificar se ocorreu o incumprimento de QoS na rota principal (QoS violation), e outra será manter as rotas alternativas sempre válidas.

    - Manutenção da rota principal: A manutenção da rota principal no que diz respeito ao atraso fim a fim é verificado pelo nó destino ao receber os pacotes transmitidos pela fonte. O nó destino calcula o atraso do pacote (T_d), subtraindo o *timestamp* armazenado no pacote colocado antes da sua transmissão pelo nó fonte pelo *timestamp* quando recebe o pacote. No caso de este tempo ser superior ao atraso fim a fim requisitado é detectada um incumprimento de QoS na rota principal.

      No caso da verificação de incumprimento da largura de banda será efectuada uma análise nó a nó. Caso seja detectada a ausência de um pacote durante um intervalo de tempo designado de *bandwidth reservation timer* (T) definido por $k \times N \times 8 / B_{min,}$ sendo k a perda de pacotes tolerável e N o tamanho máximo do pacote utilizado na camada física.

      Através dos pacotes Hello transmitidos periodicamente é possível detectar a ocorrência de falha de uma ligação. O nó que detecte uma falha de ligação, assim como um incumprimento do requisito de QoS, será responsável pelo envio em unicast de um pacote Route Error (RERR) para notificar a fonte desta ocorrência. Durante o percurso efectuado por este pacote RERR até à fonte, os nós intermédios que vão receber o pacote e retransmiti-lo para que este chegue ao nó fonte, vão libertar os recursos reversados para o fluxo.

    - Manutenção das rotas secundárias: Para a manutenção de rotas alternativas, é transmitido periodicamente um pacote de controlo designado de RouteM pelo nó fonte para estas rotas, quando o pacote chega ao destino é reencaminhado para a fonte

novamente, de forma a verificar se as rotas continuam activas e verificar se continuam a cumprir a estabilidade desejada. Este pacote também é transmitido para a rota principal, quando são recebidos os vários pacotes RouteM pelas três rotas (caso existam) transmitidos pelo nó destino, o nó fonte conforme a estabilidade de cada rota decide se comuta para uma rota secundária ou mantém a principal como a rota activa. Para não haver trocas constantes entre a rota principal por uma secundária mais estável, é definido um *threshold* para controlar o excesso de trocas de rotas.

- Recuperação da rota: Durante a manutenção de rotas, caso seja verificada alguma falha ou incumprimento de QoS nas rotas seleccionadas, procede-se ao envio de um pacote REER para o nó fonte, como descrito anteriormente.

  Se o pacote tiver sido recebido no nó fonte encaminhado por uma rota secundária, esta será removida da sua tabela de encaminhamento. No caso de o pacote ter sido transmitido pela rota principal, o nó fonte procede à comutação para a rota secundária com mais estabilidade e remove a rota principal, na condição de conter uma rota secundária válida na sua tabela de encaminhamento. No caso de o nó fonte não conter nenhuma rota secundária na sua tabela e ainda contenha pacotes para serem transmitidos, o nó fonte terá de iniciar novamente o processo de descoberta de rota.

2.7.6. Multipath QoS Routing for supporting DiffServ (MQRD)

O protocolo Multipath QoS Routing for supporting Diffserv (MQRD) [24] consiste na utilização do protocolo Node-Disjoint Multipath Routing (NDMR) [25] e dota-lo com garantias de qualidade de serviço.

O protocolo NDMR tem por objectivo minimizar os pacotes de controlo na rede e descobrir múltiplos caminhos nós-disjuntos entre um nó fonte e um nó destino. O encaminhamento com múltiplos caminhos tem a vantagem de evitar congestão nas ligações e perda de comunicação entre nós relativamente aos protocolos de encaminhamento unipath, que sempre que é detectado uma falha na rota activa é necessário desencadear um novo processo de descoberta de rota, criando mais overhead na rede e um atraso na entrega dos pacotes. Para aplicações mais exigentes, como as aplicações multimédia, este facto pode tornar-se um problema de risco, visto este tipo de aplicações imporem certos requisitos necessários para obter um desempenho aceitável e funcional para o

utilizador.

O protocolo MQRD combina as vantagens do protocolo NDMR e do DiffServ e torna-o adequado para as redes ad hoc, fornecendo assim garantias de QoS. Este protocolo obtém melhorias na performance relativamente à taxa de entrega de pacotes e no atraso fim a fim.

Segundo os autores do protocolo, a escolha do DiffServ baseou-se por ser um modelo com características simples, eficiente e escalável. Atribuindo a responsabilidade ao nó fonte pela classificação e marcação dos pacotes com o DiffServ Code Point (DSCP) correspondente à classe de serviço pretendida. Por cada classe de serviço existente é aplicado um escalonamento com prioridade (Priority Scheduling) e mecanismos de gestão das filas de espera (Queue Management) distinto. Os pacotes mais sensíveis são marcados para classes de serviço com maior prioridade, desta forma os pacotes têm um tratamento diferenciado nó a nó (per-hop behavior) de acordo com a classe a que pertencem. O escalonamento com prioridade é efectuado pelo Scheduler, que define para os diferentes tipos de classes, oportunidades de transmissão distintas. É necessário uma gestão das filas de espera que tente evitar a congestão para que a taxa de perda de pacotes com prioridade alta seja mínima. No caso de não ser possível evitar a congestão nas filas, o nó fonte é notificado procedendo ao balanceamento da carga (Load Balance).

A gestão dos recursos no protocolo MQRD serão então realizados por:

- Priority Scheduling: os nós da rede ao receberem tráfego ao qual são incapazes de o retransmitir com a mesma capacidade, os pacotes recebidos vão sendo armazenados em filas de espera até existir uma largura de banda disponível para a sua transmissão. O *Scheduler* concebido neste protocolo foi definido com duas filas de espera, uma designada de *high-priority queue* para tráfego de aplicações em tempo-real (voz e video) e outra de *low-priority queue* para tráfego *best-effort*. Na fila de espera *high-priority queue* os pacotes armazenados vão ter mais prioridades sobre os pacotes armazenados na fila de espera *low-priority queue,* só quando a fila de maior prioridade estiver vazia é que os pacotes da fila de baixa prioridade são encaminhados. A Figura 2.8 ilustra o Priority Scheduling descrito.

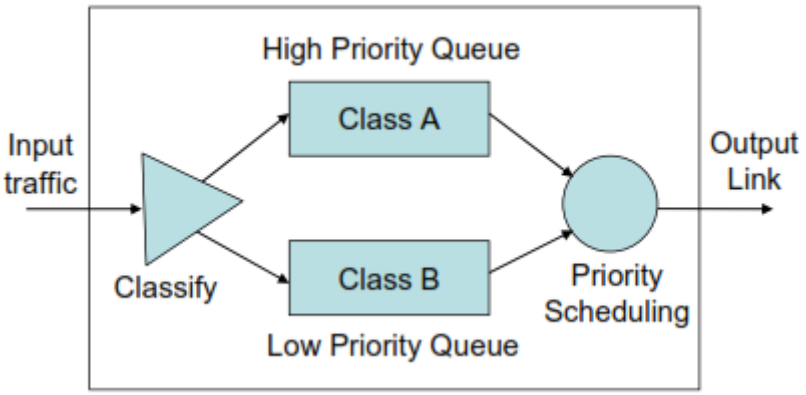


Figura 2.8: Imagem retirada de [24], Priority Scheduler

O tráfego de aplicações em tempo-real será então classificado e marcado com o DSCP correspondente, de forma a ficar associado à classe de serviço (Class A) que oferecerá tratamento com maior prioridade aos pacotes colocados na fila de espera high-priority queue. Este tipo de aplicações tem requisitos de qualidade de serviço específicos, atraso fim a fim, variações no atraso, perda de pacotes e largura de banda. Por este motivo este tipo de tráfego é adequado para o Expedited Forwarding (EF) Per-Hop Behavior.

O restante tráfego será classificado como *best-effort* (BE) e será colocado na fila de espera de baixa prioridade. Os pacotes armazenados nestas filas de espera vão receber um tratamento diferenciado no *Priority Scheduling* (Figura 2.8), que vai dar preferência aos pacotes contidos na fila de maior prioridade e só depois transmite os pacotes colocados na fila de baixa prioridade, visto que este tipo de pacotes definidos para esta classe de serviço não será sensível ao atraso.

- Queue Management: as filas de espera têm um *buffer* de tamanho fixo, podendo armazenar um número limitado de pacotes. Nas filas de espera do tipo Tail Drop, caso os pacotes armazenados ocupem a capacidade máxima da fila de espera, não tendo sido possível dar vazão à quantidade de pacotes recebidos, os pacotes que chegam são descartados até que a fila tenha espaço de armazenamento suficiente para receber os próximos pacotes.

  Neste sentido, pretende-se utilizar outro algoritmo que resolva o problema do Tail Drop, e não deixe a fila de espera chegar à sua capacidade máxima e comece a descartar pacotes, mesmo sendo pacotes marcados com prioridade alta. Desta forma, foi utilizado o algoritmo random early detection (RED) para evitar a congestão. Neste algoritmo se a fila se encontra praticamente vazia, os pacotes recebidos são aceites, à medida que a fila de espera aumenta e ultrapassa um

determinado *threshold* mínimo, a probabilidade de descartar pacotes aumenta até chegar a um *threshold* máximo. Depois de passar o valor do *threshold* máximo os pacotes que chegarem são imediatamente descartados.

- Load Balance e Congestion Avoidance: este protocolo no processo de descoberta de rotas obtém múltiplas rotas disjuntas como mencionado anteriormente. Para tentar evitar a congestão da rede, sempre que é detectado por um determinado nó que a ocupação da fila de espera é superior ao threshold mínimo definido no RED, o nó notifica a fonte com o envio de um pacote Congestion Notification (CN) pelo percurso inverso. Com a notificação recebida, o nó fonte faz a distribuição do tráfego pelas outras rotas disjuntas contidas na sua tabela de encaminhamento.

Capítulo 3

Descrição da proposta para encaminhamento com QoS em redes móveis Ad Hoc

Neste capítulo apresenta-se a solução proposta e desenvolvida no âmbito do objecto de estudo desta dissertação, sendo descrito detalhadamente o protocolo de encaminhamento com garantias de QoS para as redes móveis ad hoc proposto, designado de Ad hoc QoS On-Demand Multipath Routing with Route Stability (QMRS). Primeiramente neste capítulo, descrever-se-á as várias decisões e etapas necessárias para atingir a solução final, entre as quais, destaca-se o protocolo de múltiplos caminhos de nós disjuntos, designado de Ad hoc On-Demand Multipath Routing (AMR).

A solução proposta de um protocolo de encaminhamento com QoS em redes Ad hoc, teve como base o protocolo Ad Hoc On-Demand Distance Vector (AODV) [13]. Utilizaram-se algumas das características deste protocolo, tendo sido acrescentadas outras funcionalidades e abordagens diferentes, algumas das quais, utilizadas nos vários protocolos descritos no Capítulo 2, entre os quais se podem destacar o protocolo Ad hoc on-demand multipath distance vector (AOMDV) [18] e o protocolo Route Stability based Multipath QoS Routing (SMQR) [4].

O protocolo AODV como foi descrito mais detalhadamente na Secção 2.3.2, é um protocolo reactivo. O seu processo de descoberta de rota consiste no envio de um pedido do nó fonte para encontrar uma rota para determinado destino, para que possivelmente obtenha uma resposta a este pedido de rota por algum nó com o conhecimento de uma rota para o destino. Este protocolo tem determinadas limitações, entre as quais se destaca o facto de, durante o processo de descoberta de rota apenas possibilitar encontrar uma única rota entre o nó fonte e o destino. Com a mobilidade dos nós inerente a este tipo de redes, onde ocorrem frequentes quebras de rotas, este factor torna-se uma limitação, visto que, a qualquer momento, quando o caminho deixa de ser exequível, terá de ser inicializado novamente por parte do nó fonte o processo de descoberta de rota. Isto pode causar um atraso na transmissão. Quando se pretende oferecer uma solução de um protocolo de encaminhamento com um bom desempenho e que forneça garantias de QoS no que diz respeito ao

atraso, este tipo de pedidos constantes de descoberta de rota não são desejados. Para colmatar o problema descrito, presente nos protocolos de encaminhamento *unipath* das redes móveis ad hoc, começou-se por desenvolver um protocolo de encaminhamento de múltiplos caminhos de nós disjuntos, designado de Ad hoc On-Demand Mulitpath Routing (AMR), protocolo esse que serviu de base à proposta final apresentada de um protocolo de encaminhamento com QoS, designado de Ad hoc QoS On-Demand Multipath Routing with Route Stability (QMRS).

3.1. Protocolo de encaminhamento de múltiplos caminhos

Nesta secção será descrito o protocolo de encaminhamento Ad hoc On-Demand Multipath Routing (AMR) proposto. Algumas das características deste protocolo foram baseadas no protocolo Ad Hoc On-Demand Distance Vector [13] e no protocolo Ad Hoc On-Demand Multipath Distance Vector (AOMDV) [18].

Com o objectivo de apresentar uma solução de um protocolo de encaminhamento de múltiplos caminhos com um bom desempenho, entre as várias categorias e abordagens utilizadas em vários protocolos de encaminhamento em redes móveis ad hoc, algumas das quais mencionadas no Capítulo 2, foram tomadas várias decisões por forma a alcançar a proposta apresentada.

A abordagem utilizada pelo protocolo AMR, utiliza apenas a informação disponível em cada nó de forma a tomar as decisões de encaminhamento, ao contrario da abordagem utilizada pelos protocolos que recorrem ao source routing, existente por exemplo no protocolo Dynamic Source Routing (DSR) [14], que consiste em durante a descoberta de rota, acumular o endereço de todos os nós entre a fonte e o destino. Após efectuada essa descoberta do caminho, antes de o nó fonte dar inicio à transmissão do fluxo de dados, incluí o caminho encontrado no cabeçalho dos pacotes, para que, durante a sua transmissão, estes possam ser encaminhados de acordo com essa informação.

O protocolo AMR é um protocolo reactivo (On-Demand), ou seja, apenas quando um nó pretende transmitir para determinado destino e não contém na sua tabela de encaminhamento uma rota válida para este, terá de iniciar o processo de descoberta de rota, para que possivelmente obtenha uma resposta de algum nó com a informação de uma rota válida para o destino. Este tipo de abordagem exige menos pacotes de controlo na rede, ao contrário dos protocolos que utilizam uma abordagem proactiva, que têm por objectivo, manter a informação actualizada nas tabelas de encaminhamento,

sendo para isso necessário uma troca de mensagens periódica entre os nós da rede, para obter e actualizar as rotas para cada nó destino pertencente à rede.

O protocolo proposto, ao possibilitar encontrar múltiplos caminhos de nós disjuntos entre uma fonte e um destino, permite que quando ocorre uma quebra de ligação entre dois nós pertencentes ao caminho principal, seja possível efectuar a comutação para uma rota alternativa para que a fonte possa dar continuidade à transmissão dos dados, sem a necessidade de iniciar um novo processo de descoberta de rota. Desta forma, juntamente com as restantes opções descritas anteriormente para a solução apresentada, verifica-se que o protocolo reduz o número necessário de pacotes de controlo na rede e diminui o atraso durante a transmissão do fluxo de dados entre as fontes e os destinos.

Nas secções seguintes descreve-se o formato das mensagens utilizadas no protocolo AMR, assim como, os mecanismos de descoberta e manutenção de rotas existentes que, funcionando em conjunto, permitem aos nós da rede, descobrir e manter rotas para determinados destinos na rede ad hoc.

3.1.1. Formato das mensagens

Em seguida apresentam-se os formatos das mensagens utilizadas no protocolo AMR, para que nas secções seguintes, durante a explicação dos mecanismos existentes no protocolo, seja mais fácil identificar os campos referidos de cada mensagem e a sua utilização para o funcionamento do protocolo.

Na Tabela 3.1 podem-se observar os campos pertencentes à mensagem de Route Request (RREQ).

Tabela 3.1: Mensagem RREQ do protocolo AMR

Request ID
flags
Origin IP Address
Origin Sequence Number
Destination IP Address
Destination Sequence Number
hopCount
firstHop

A mensagem de Route Request é constituída pelos seguintes dados:

- Request ID: um número sequencial, incrementado em cada pedido de rota, que juntamente com o endereço IP do nó fonte, permite identificar unicamente cada pacote RREQ;
- Flags: contém a informação que permite durante a procura de rota verificar se um nó intermédio com conhecimento de uma rota para o destino, poderá responder a esse pedido de rota (flag G – gratuitous Reply) ou terá de retransmiti-lo e apenas o próprio destino poderá responder ao pedido (flag D – Destination Only);
- Origin IP Address: endereço IP do nó que iniciou o enviou do RREQ (nó fonte);
- Origin Sequence Number: um número incrementado sequencialmente, utilizado para verificar se as rotas são válidas ou já se encontram obsoletas, conforme o valor contido nas mensagens e nas entradas das tabelas de encaminhamento na rota para o nó fonte;
- Destination IP Address: endereço IP do nó para o qual se pretende obter uma rota (nó destino);
- Destination Sequence Number: o último número de sequência do nó destino, do qual o nó fonte tem conhecimento;
- hopCount: o número de saltos entre o nó fonte e o nó actual que recebeu o RREQ;
- firstHop: o primeiro salto da mensagem RREQ, ou seja, conterá o endereço IP do vizinho do nó fonte que irá retransmitir a mensagem. Este campo corresponderá ao último salto para o nó fonte, inserido nas entradas construídas durante a retransmissão do RREQ, e permite deste modo identificar os caminhos de nós disjuntos encontrados;

Na Tabela 3.2 podem-se observar os campos pertencentes à mensagem de Route Reply (RREP).

Tabela 3.2: Mensagem RREP do protocolo AMR

Request ID
flags
Origin IP Address
Destination IP Address
Destination Sequence Number
hopCount
firstHop
lifetime

A mensagem de Route Reply é constituída pelos seguintes dados:

- Request ID: Request ID da correspondente mensagem RREQ
- Flags: contém a informação que possibilita verificar se é pretendido um *acknowledgment;*
- Origin IP Address: endereço IP do nó fonte que fez o pedido de rota, e que será o destino da resposta, com a informação da respectiva rota encontrada;
- Origin Sequence Number: número de sequência do nó fonte;
- Destination IP Address: endereço IP do destino da respectiva rota encontrada e enviada na mensagem RREP para o nó fonte;
- Destination Sequence Number: o número de sequência do nó destino, correspondente à rota para o destino enviada na mensagem RREP;
- hopCount: o número de saltos entre o nó fonte e o destino;
- firstHop: este campo corresponderá ao último salto para o nó destino, inserido nas entradas construídas durante a retransmissão do RREP, permite deste modo identificar os caminhos de nós disjuntos encontrados;
- lifetime: o tempo em milissegundos enviado na mensagem para definir o tempo de validade das rotas inseridas para o destino, durante a retransmissão do RREP em cada nó;

Na Tabela 3.3 podem-se observar os campos pertencentes à mensagem de Route Error (RERR).

Tabela 3.3: Mensagem RERR do protocolo AMR

Destination Count
Unreachable Destination IP Address (1)
Unreachable Destination Sequence Number (1)
...
...

A mensagem de Route Error é constituída pelos seguintes dados:

- Destination Count: número de destinos inacessíveis incluídos na mensagem;
- Unreachable Destination IP Address: endereço IP do destino que ficou inacessível devido a uma quebra de ligação.
- Unreachable Destination Sequence Number: número de sequência contido na tabela de encaminhamento da entrada para o destino correspondente ao endereço IP do campo anterior;

3.1.2. Descoberta de rota

O processo de descoberta de rota no protocolo AMR, sendo este um protocolo reactivo, apenas quando um nó pretende transmitir e não contém na sua tabela de encaminhamento uma rota válida para este, terá de iniciar o processo de descoberta de rota, para que possivelmente obtenha uma resposta de algum nó que tenha conhecimento de uma rota para o destino. Este protocolo permite ao nó fonte, durante um único processo de descoberta de rota, obter múltiplos caminhos para um único destino. Alguns estudos [26] demonstram que durante o processo de descoberta de rota, encontrar mais de três caminhos para um único destino não é vantajoso. Os resultados dos testes realizados indicam que o ideal será encontrar no máximo dois ou três caminhos num único processo de descoberta de rota. Neste sentido, o protocolo proposto possibilita durante o processo de descoberta de rota, encontrar no máximo até três caminhos de nós disjuntos.

A descoberta de múltiplos caminhos de nós disjuntos permite que, quando se verifica uma falha na rota principal devido à movimentação de algum nó pertencente a esta, o nó fonte reaja à notificação dessa quebra, procedendo à comutação para uma rota alternativa. A probabilidade de existir uma rota alternativa válida é elevada graças ao algoritmo utilizado para a descoberta de rotas. Este algoritmo considera como rotas alternativas apenas aquelas em que entre a fonte e o destino não existe um mesmo nó intermédio. Se os caminhos encontrados não fossem de nós disjuntos, com a movimentação de apenas um nó poder-se-iam quebrar logo todas as rotas encontradas entre a fonte e o destino, tendo o nó fonte que iniciar novamente o processo de descoberta de rota.

A Figura 3.1 ilustra o conceito de caminhos de nós disjuntos. Com a topologia apresentada, durante o processo de descoberta de rotas, seria possível obter três caminhos de nós disjuntos (fonte-A-D-destino; fonte-B-destino; fonte-C-E-destino), ou seja, o mesmo nó intermédio só poderá pertencer a um dos caminhos entre a fonte e o destino.

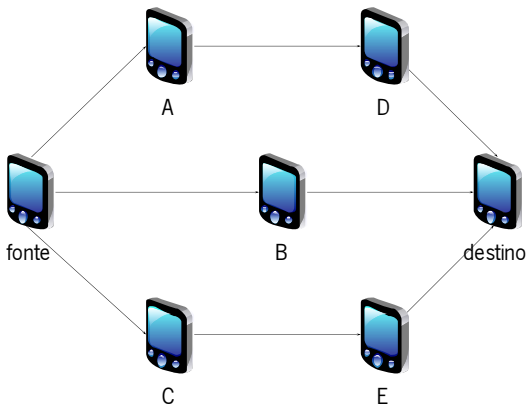


Figura 3.1: Caminhos de nós disjuntos

Um nó da rede deve ter associado a cada entrada na tabela de encaminhamento o último *destination sequence number* conhecido do nó destino. Deste modo, um nó da rede ao receber a informação de uma rota para um destino, verifica se essa informação é mais recente que a que detém até ao momento, através da comparação entre o destination sequence number contido nas mensagens de controlo (mensagem RREQ e RREP) com o contido na sua tabela de encaminhamento, e para o caso de o campo *destination sequence number* da mensagem ser superior ao da tabela, o nó actualiza o destination sequence number do destino correspondente, na entrada da sua tabela de encaminhamento.

Um nó da rede aquando a recepção de uma mensagem RREQ com um destination sequence number superior ao da tabela de encaminhamento, não irá responder ao pedido de rota, visto que esta se encontra obsoleta. O nó terá de invalidar a entrada da sua tabela de encaminhamento para o destino e retransmitir a mensagem, para que outro nó da rede que possua uma rota válida para o destino, possa responder ao pedido de rota. O campo *destination sequence number* contido nas mensagens de controlo, juntamente com o campo resquest ID e o Origin IP Address, são essenciais para permitir distinguir pedidos sucessivos e evitar ciclos no encaminhamento.

O processo de descoberta de rota no protocolo AMR, conforme referido anteriormente, é desencadeado no nó fonte, através do envio em *broadcast* de um pacote Route Request (RREQ). Este pacote será retransmitido sempre que necessário nos vários nós da rede até que possivelmente possa chegar a um nó que possua na sua tabela de encaminhamento uma rota válida para o destino, ou ao próprio destino, que respondem ao pedido de rota procedendo ao envio de um Route Reply (RREP) em *unicast* para o nó fonte.

Na Figura 3.2 pode-se observar o início do processo de descoberta de rota, com o envio do RREQ no nó fonte (a cor vermelha) e a sua retransmissão nos nós intermédios, até que este chega a um nó que detém uma rota válida para o destino (nós G,F e H). Um nó intermédio na recepção de um pacote RREQ verifica se é a primeira vez que recebe este pedido de rota do nó fonte, caso seja uma réplica, o pacote será descartado. É possível efectuar esta verificação, através dos campos requestID e o endereço IP do nó fonte contidos na mensagem RREQ, que permitem identificar unicamente cada pacote. Para isso a informação do primeiro pacote RREQ recebido tem de ser armazenada no nó para poder ser comparada com as possíveis réplicas que possa receber. Caso seja a primeira vez que um nó intermédio recebe um pacote RREQ retransmite-o em broadcast para os seus nós vizinhos e estes fazem-no igualmente, para que o pacote seja difundido na rede e possa chegar a um nó que contenha uma rota válida para o destino. A rota só é considerada válida, no caso de o nó intermédio conter na sua tabela de encaminhamento uma rota para o destino com um destination sequence number igual ou superior ao contido no pacote RREQ recebido.

Antes de cada nó intermédio proceder à retransmissão do pacote RREQ, estes criam entradas na tabela de encaminhamento para o nó fonte, para que ao receberem uma resposta ao pedido (pacote RREP), possam reencaminha-la para o nó fonte, pelo caminho encontrado para o destino, mas no

sentido inverso ao percorrido pelo pacote RREQ, ou seja, no sentido do nó destino para a fonte.

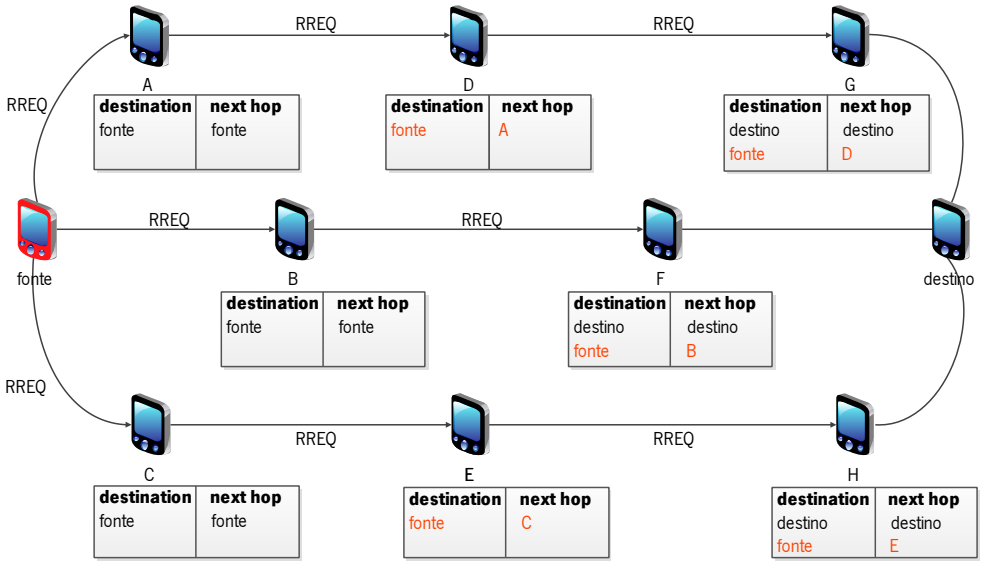


Figura 3.2: Nó fonte inicia o processo de descoberta de rota

Pode-se verificar no exemplo da Figura 3.2, as entradas criadas na tabela de encaminhamento para o nó fonte na recepção do pacote RREQ em cada nó intermédio, apresentadas a cor vermelha (para se diferenciar das rotas existentes para os vizinhos), de realçar que a figura apresentada contém apenas a informação relevante para descrever o processo desde, o início do pedido de rota até o pacote RREQ chegar a um nó com o conhecimento de uma rota válida para o destino.

A Figura 3.3 ilustra um exemplo que pretende dar seguimento ao anteriormente apresentado na Figura 3.2, que diz respeito à continuação do processo de descoberta de rota, com o envio da resposta (pacote RREP) por parte dos nós intermédios (nós G,F e H, exibidos a cor vermelha) a um pedido de rota do nó fonte para o destino.

O protocolo proposto, como já mencionado, permite descobrir múltiplos caminhos de nós disjuntos, ao difundir os pacotes RREQ por vários caminhos. Possivelmente o nó fonte obtém múltiplas respostas. As respostas por parte dos nós intermédios que possuem rotas válidas para o destino, consistem no envio de um pacote RREP em *unicast* endereçado ao nó fonte, que irá percorrer o caminho inverso, sendo encaminhadas nó a nó, através de uma consulta na tabela de encaminhamento da entrada correspondente ao nó fonte, entrada essa que corresponde às rotas anteriormente inseridas na tabela de encaminhamento durante a retransmissão do RREQ, apresentadas a cor vermelha na Figura 3.2.

Em cada nó intermédio pertencente a um dos caminhos encontrados, antes de retransmitir o pacote RREP, são criadas entradas na tabela de encaminhamento para o nó destino, para que depois de a fonte iniciar a transmissão do fluxo de dados, cada pacote possa ser encaminhado nó a nó conforme estas rotas, representadas no exemplo da Figura 3.2 a cor vermelha, nas tabelas de encaminhamento apresentadas em cada nó intermédio. Para além de um pacote RREP transmitido em *unicast* para o nó fonte, os nós intermédios que respondem ao pedido de rota (nós G,F e H) enviam também um pacote gratuitous RREP transmitido em *unicast* para o nó destino, para o informar da rota para o nó fonte. Desta forma, o nó destino também tem a possibilidade de obter múltiplas rotas para o nó fonte. As múltiplas rotas obtidas durante o processo de descoberta de rota, nos vários nós da rede intervenientes neste processo, permitem uma redução na frequência com que estes nós necessitam de utilizar o mecanismo de descoberta de rota, verificando-se por isso uma diminuição dos pacotes de controlo na rede.

O nó fonte, aquando a recepção da primeira resposta ao pedido, ou seja, na recepção do primeiro pacote RREP, define essa como a rota principal para o destino, ficando as restantes rotas como alternativas. O nó fonte, após obter a primeira resposta, procede imediatamente ao início da transmissão dos dados para o destino.

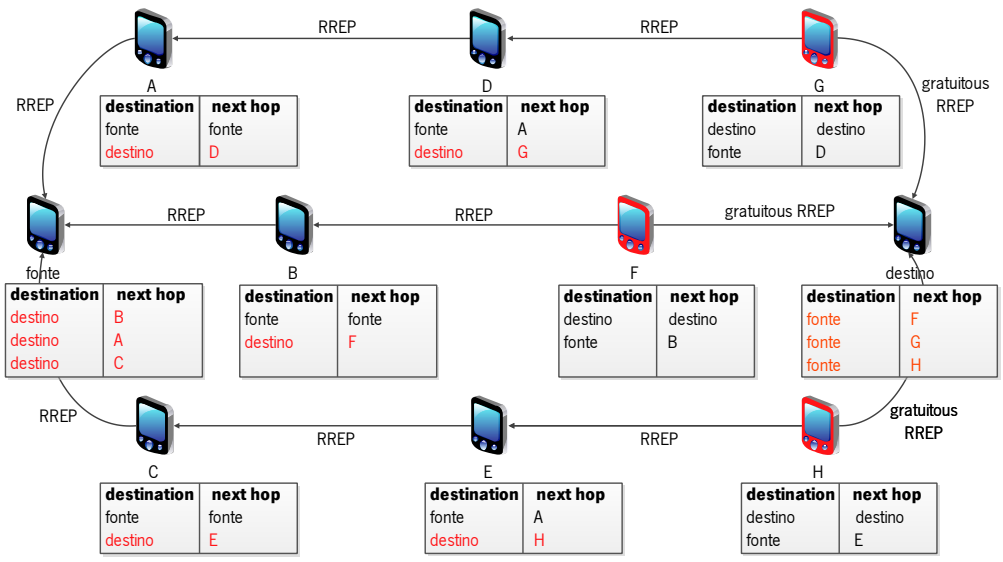


Figura 3.3: Resposta dos nós intermédios ao pedido de rota do nó fonte

O mecanismo de manutenção de rotas, apresentado de seguida, corresponde à manutenção das rotas encontradas, em que se descreve qual a vantagem e o objectivo, das rotas alternativas obtidas

durante o processo de descoberta de rota descrito, e como é efectuada a recuperação da rota para dar seguimento à transmissão por parte das fontes quando a rota principal deixa de válida.

3.1.3. Manutenção de rotas

O protocolo AMR, através de uma troca periódica de mensagens HELLO entre os nós vizinhos, permite verificar a ocorrência de uma falha na ligação entre os nós da rede, deste modo, caso um nó detecte essa falha de ligação e contenha rotas na sua tabela de encaminhamento para determinado destino que utilizem como próximo salto o nó vizinho que deixou de estar disponível, terá de informar os restantes nós vizinhos que deixou de ter uma rota válida para determinados destinos. Os nós que recebem essa notificação de falha, ou seja, na recepção de um pacote Route Error (RERR), removem as rotas correspondentes e retransmitem o pacote, para que todos os nós intermédios pertencentes ao caminho de nós disjuntos removam a rota correspondente das suas tabelas de encaminhamento e retransmitam a mensagem, até esta chegar ao nó fonte. O nó fonte, assim como os nós intermédios pertencentes a esse caminho que ficou inexequível, ao receber o pacote RERR, removem das suas tabelas de encaminhamento as rotas para o destino. No caso em que a rota para o destino indicada no pacote RERR recebido corresponda à rota definida como principal e utilizada ate ao momento pelo nó fonte para a transmissão dos dados, esta deverá ser substituída por uma rota alternativa para esse destino, caso exista na sua tabela de encaminhamento. Se for esse o caso, o nó procede à comutação para essa rota e continua a transmissão dos dados. No caso de o nó fonte ao remover a rota principal, não possuir nenhuma rota alternativa na sua tabela de encaminhamento, terá de iniciar o processo de descoberta de rota.

Seguindo a mesma topologia da Figura 3.2 e Figura 3.3, a Figura 3.4 demonstra um exemplo onde os nós da rede pertencentes aos caminhos de nós disjuntos encontrados, contêm as rotas nas suas tabelas de encaminhamentos para o nó fonte e para o nó destino, e ocorre uma falha de ligação entre o nó F e o destino (devido ao facto de por exemplo, um dos nós ter-se deslocado para fora do alcance de transmissão, deixando de ocorrer uma troca de mensagens “Hello” entre os nós vizinhos) apresentada a cor vermelha.

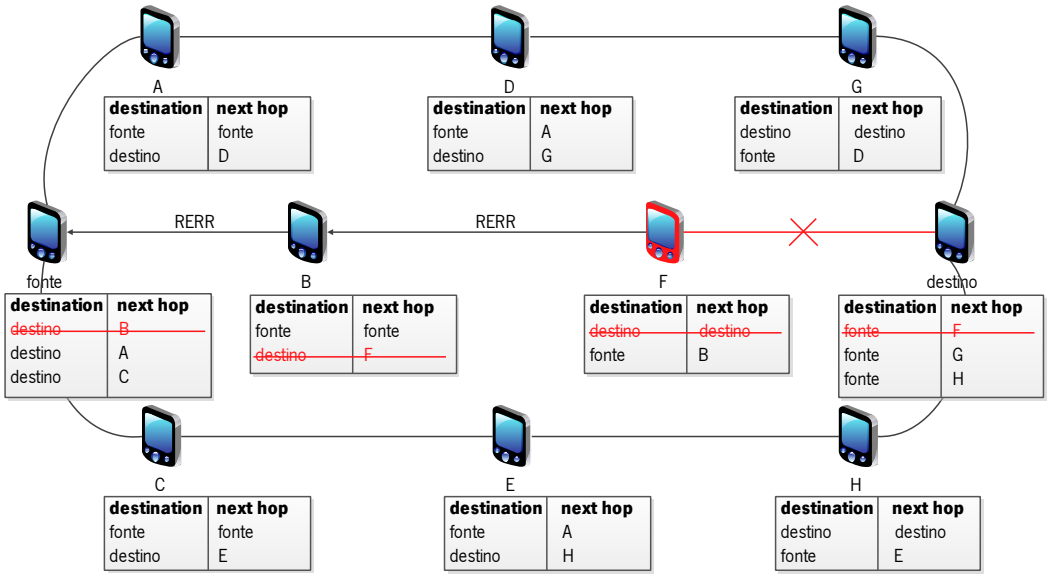


Figura 3.4: Envio do pacote RERR para informar a ocorrência de falha na ligação para destino

O destino ao detectar a falha de ligação remove da sua tabela de encaminhamento as rotas que usavam o nó F como próximo salto. O nó F ao detectar a ocorrência da falha de ligação, faz a remoção da entrada da tabela de encaminhamento que utilizava como próximo salto o destino, e procede ao envio do pacote RERR, de forma a informar o nó B que já não o pode utilizar como próximo salto para o destino. O nó B ao receber o pacote RERR remove da sua tabela de encaminhamento a rota para o destino que utilizava como próximo salto o nó que enviou o pacote (nó F) e retransmite a mensagem. O nó fonte ao receber a notificação de falha de rota, remove a rota correspondente para o destino que utilizava como próximo salto o nó que enviou a mensagem (nó B), e neste caso, como a rota eliminada era a rota principal, terá de comutar para uma rota alternativa, que seria a rota para o destino em que o próximo salto é o nó A.

O protocolo proposto, ao permitir encontrar múltiplos caminhos de nós disjuntos e a possibilidade de comutar para uma rota alternativa quando o caminho principal deixa de ser exequível oferece, desta forma, uma redução na frequência com que os nós fontes teriam em iniciar o processo de descoberta de rota sempre que fosse verificada uma falha. Deste modo, julga-se contribuir para uma diminuição de tráfego de controlo na rede, um menor atraso fim a fim e um aumento na taxa de pacotes entregues no destino.

3.2. Protocolo de encaminhamento com QoS

Nesta secção descreve-se o protocolo de encaminhamento com QoS para as redes móveis ad hoc proposto, designado de Ad hoc QoS On-Demand Multipath Routing with Route Stability (QMRS).

O protocolo QMRS proposto, é baseado no protocolo de encaminhamento de múltiplos caminhos descrito na Secção 3.1, o protocolo Ad hoc On-Demand Multipath Routing (AMR). Foi necessário assim, efectuar alterações ao protocolo AMR de forma a obter a solução final proposta, que deverá atender aos requisitos de QoS de determinada aplicação, sem comprometer a estabilidade do processo de encaminhamento.

O protocolo proposto possibilita num único processo de descoberta de rota, encontrar múltiplos caminhos de nós disjuntos que cumpram o requisito de atraso fim a fim máximo, entre a fonte e o destino. O nó fonte irá utilizar para transmissão o caminho mais estável entre todos os encontrados, com o objectivo de que este se mantenha exequível pelo maior período de tempo possível, sem ter a necessidade de comutar para outra rota alternativa, ou mesmo iniciar o processo de descoberta de rota no caso de não existir uma rota alternativa quando é detectada uma falha na rota principal. Apenas no caso de serem encontradas rotas com uma estabilidade idêntica é seleccionada a rota com menor atraso fim a fim. A descoberta de múltiplos caminhos de nós disjuntos alternativos permite que, quando se verifica uma falha na rota principal devido à movimentação de algum nó pertencente a esta, o nó fonte reaja à notificação dessa quebra, procedendo à comutação para uma rota alternativa. Neste sentido, com a solução proposta, diminui-se o tráfego de controlo na rede e verifica-se um menor atraso fim a fim, durante a transmissão do fluxo de dados.

Uma vez descobertas as rotas de nós disjuntos entre a fonte e o destino que cumpram o requisito de QoS, o protocolo QMRS utiliza, a potência de sinal recebido para determinar a rota mais estável e o atraso fim a fim em caso de empate, para seleccionar a melhor rota para transmissão. Sobre todos caminhos encontrados entre o nó fonte e o destino, é realizado o mecanismo de manutenção de rotas, que verifica periodicamente se o cumprimento de QoS se mantém, assim como, efectua uma actualização da potência do sinal mínima do caminho e do atraso fim a fim. Com base na actualização das métricas, poderá ocorrer uma comutação de rota. Com os mecanismos utilizados o protocolo proposto possibilita assim efectuar uma transmissão de dados eficiente oferecendo garantias de QoS.

Em seguida descreve-se o formato das mensagens utilizadas no protocolo QMRS, assim como, os seus mecanismos de descoberta e manutenção de rotas, que funcionando em conjunto permitem aos nós da rede, descobrir e manter rotas para determinados destinos na rede ad hoc que cumpram o requisito de QoS.

3.2.1. Formato das mensagens

Em seguida apresentam-se os formatos das mensagens utilizadas no protocolo QMRS, para que nas secções seguintes, durante a explicação dos mecanismos existentes no protocolo, seja mais fácil identificar os campos referidos de cada mensagem e a sua utilização para o funcionamento do protocolo.

Na Tabela 3.4 podem-se observar os campos pertencentes à mensagem de Route Request (RREQ).

Tabela 3.4: Mensagem RREQ do protocolo QMRS

Request ID
flags
Origin IP Address
Origin Sequence Number
Destination IP Address
Destination Sequence Number
hopCount
firstHop
RxSSPath
delayReq
delayAcc
delayToOrigin
delayPath

A mensagem de Route Request é constituída pelos seguintes dados:

- Request ID: um número sequencial, incrementado em cada pedido de rota, que juntamente com

o endereço IP do nó fonte, permite identificar unicamente cada pacote RREQ;

- Flags: contém a informação que permite durante a procura de rota verificar se um nó intermédio com conhecimento de uma rota para o destino, poderá responder a esse pedido de rota (flag G – gratuitous Reply) ou terá de retransmiti-lo e apenas o próprio destino poderá responder ao pedido (flag D – Destination Only);
- Origin IP Address: endereço IP do nó que iniciou o enviou do RREQ (nó fonte);
- Origin Sequence Number: um número incrementado sequencialmente, utilizado para verificar se as rotas são válidas ou já se encontram obsoletas, conforme o valor contido nas mensagens e nas entradas das tabelas de encaminhamento na rota para o nó fonte;
- Destination IP Address: endereço IP do nó para o qual se pretende obter uma rota (nó destino);
- Destination Sequence Number: o último número de sequência do nó destino, do qual o nó fonte tem conhecimento;
- hopCount: o número de saltos entre o nó fonte e o nó actual que recebeu o RREQ;
- firstHop: o primeiro salto da mensagem RREQ, ou seja, conterá o endereço IP do vizinho do nó fonte que irá retransmitir a mensagem. Este campo corresponderá ao último salto para o nó fonte, inserido nas entradas construídas durante a retransmissão do RREQ, e permite deste modo identificar os caminhos de nós disjuntos encontrados;
- RxSSPath: durante a retransmissão da mensagem RREQ nó a nó, este campo conterá a informação do valor da potência do sinal recebido mínimo desse caminho;
- delayReq: atraso fim a fim máximo requisitado por determinada aplicação;
- delayAcc: atraso fim a fim acumulado no caminho percorrido;
- delayToOrigin: atraso fim a fim entre o nó actual e o nó fonte;
- delayPath: conterá o valor do atraso fim a fim do caminho até ao destino;

Na Tabela 3.5 podem-se observar os campos pertencentes à mensagem de Route Reply (RREP).

Tabela 3.5: Mensagem RREP do protocolo QMRS

Request ID
flags
Origin IP Address
Destination IP Address
Destination Sequence Number
hopCount
lifetime
firstHop
RxSSPath
RxSSPathToDst
delayReq
delayToDst
delayPath

A mensagem de Route Reply é constituída pelos seguintes dados:

- Request ID: Request ID da correspondente mensagem RREQ
- Flags: contém a informação que possibilita verificar se é pretendido um *acknowledgment;*
- Origin IP Address: endereço IP do nó fonte que fez o pedido de rota, e que será o destino da resposta, com a informação da respectiva rota encontrada;
- Origin Sequence Number: número de sequência do nó fonte;
- Destination IP Address: endereço IP do destino da respectiva rota encontrada e enviada na mensagem RREP para o nó fonte;
- Destination Sequence Number: o número de sequência do nó destino, correspondente à rota para o destino enviada na mensagem RREP;
- hopCount: número de saltos entre o nó fonte e o destino;
- lifetime: o tempo em milissegundos enviado na mensagem para definir o tempo de validade das

rotas inseridas para o destino, durante a retransmissão da mensagem RREP em cada nó;

- firstHop: este campo corresponderá ao último salto para o nó destino, inserido nas entradas construídas durante a retransmissão da mensagme RREP, permite deste modo identificar os caminhos de nós disjuntos encontrados;
- RxSSPath: conterá a informação do valor da potência do sinal recebido mínima do caminho para o destino;
- RxSSPathToDst: durante a retransmissão da mensagem RREP, este campo vai conter a potência do sinal recebido mínima do caminho para o destino;
- delayReq: atraso fim a fim máximo requisitado por determinada aplicação;
- delayToDst: atraso fim a fim do nó actual para o destino;
- delayPath: atraso fim a fim do caminho encontrado, entre o nó fonte e o destino;

Na Tabela 3.6 podem-se observar os campos pertencentes à mensagem de Route Error (RERR).

Tabela 3.6: Mensagem RERR do protocolo QMRS

Destination Count
Unreachable Destination IP Address (1)
Unreachable Destination Sequence Number (1)
...
...

A mensagem de Route Error é constituída pelos seguintes dados:

- Destination Count: número de destinos inacessíveis incluídos na mensagem;
- Unreachable Destination IP Address: endereço IP do destino que ficou inacessível devido a uma quebra de ligação.
- Unreachable Destination Sequence Number: número de sequência contido na tabela de encaminhamento da entrada para o destino correspondente ao endereço IP do campo

anterior;

As mensagens seguintes são idênticas às mensagens RREQ e RREP descritas anteriormente, sendo que no mecanismo de manutenção de rotas na Secção 3.2.5, explica a utilidade de cada campo pertencente a estas mensagens.

Na Tabela 3.7 podem-se observar os campos pertencentes à mensagem de InspectPath.

Tabela 3.7: Mensagem InspectPath do protocolo QMRS

Request ID
Origin IP Address
Destination IP Address
Origin Sequence Number
Destination Sequence Number
hopCount
firstHop
lastHop
RxSSPath
delayAcc
delayPath
delayReq
delayToOrigin

Na Tabela 3.8 podem-se observar os campos pertencentes à mensagem de ReplyInspectPath

Tabela 3.8: Mensagem ReplyInspectPath do protocolo QMRS

Request ID
Origin IP Address
Destination IP Address
Origin Sequence Number
Destination Sequence Number
firstHop
RxSSPath
RxSSPathToDst
delayReq
delayToDst
delayPath

3.2.2. Manutenção da informação de estado entre vizinhos

Os nós da rede oferecem informação de conectividade para os seus vizinhos através do envio de mensagens "Hello". Este tipo de mensagem, corresponde a uma mensagem Route Reply com um Time To Live (TTL) de 1, enviada periodicamente em broadcast e recebida nos nós vizinhos. Desta forma, os nós localizados dentro do alcance de transmissão na recepção deste tipo de mensagens, são notificados que o vizinho está disponível e acessível (no seu raio de alcance), informação essa que é necessária para o procedimento de descoberta e manutenção das rotas.

Para além da acessibilidade, através das mensagens "Hello" é mantida a informação para cada vizinho da potência do sinal recebido e do atraso ocorrido (desde que o pacote é processado no nó e colocado na fila de espera, até que é a mensagem é enviada, inclui assim os atrasos de, processamento, gasto na fila de espera, de contenção no acesso ao meio e o de transmissão). Isto é necessário para manter os valores das métricas actualizadas e dar possibilidade aos nós intermédios de responder aos pedidos de rota por parte de uma fonte em que o destino é um dos seus vizinhos. A troca das mensagens de "Hello" é periódica e no caso de não ser recebida uma mensagem de um vizinho durante um determinado intervalo de tempo é desencadeado o processo de recuperação de

rota. A potência do sinal recebido é obtida directamente na camada física através de uma interacção cross-layer.

3.2.3. Descoberta de rota

O protocolo QMRS tendo sido baseado no protocolo AMR, o seu processo de descoberta de rotas inclui as mesmas características do algoritmo apresentado no protocolo AMR, descrito na Secção 3.1.2. Tendo sido efectuadas as alterações necessárias ao algoritmo e introduzidos mecanismos adicionais, para permitir a descoberta de múltiplos caminhos de nós disjuntos que cumpram determinado requisito de atraso fim a fim máximo. Além disso para cada caminho encontrado o algoritmo do QMRS determina a potência de sinal (métrica côncava, que resulta do cálculo do mínimo valor da potência de sinal recebido nas várias ligações que constituem o caminho), e o atraso fim a fim (métrica aditiva, que resulta do somatório de todos os atrasos obtidos entre os nós que constituem o caminho). Durante o processo de descoberta de rota, o protocolo QMRS, utiliza os campos adicionais nas mensagens de controlo (RREQ e RREP), apresentadas na Tabela 3.4 e Tabela 3.5, comparativamente às apresentadas para o protocolo AMR na Tabela 3.1 e Tabela 3.2. de forma a verificar o cumprimento do requisito do atraso fim a fim máximo e determinar a potência do sinal mínima do caminho e o atraso fim a fim.

Sendo este um protocolo reactivo, um nó da rede quando pretende transmitir para determinado destino e não contém na sua tabela de encaminhamento uma rota válida para o mesmo terá de dar início ao processo de descoberta de rota. O algoritmo utilizado neste protocolo para o processo de descoberta de rota, mantém as mesmas especificações do apresentado para o protocolo AMR, mas com o acréscimo dos aspectos que serão descritos de seguida.

O protocolo proposto possibilita num único processo de descoberta de rota, encontrar múltiplos caminhos de nós disjuntos que cumpram o requisito de atraso fim a fim máximo, entre a fonte e o destino. O nó fonte irá utilizar para transmissão o caminho mais estável entre todos os encontrados, com o objectivo de que o caminho principal seleccionado se mantenha exequível por um maior período de tempo possível, sem ter a necessidade de comutar para outra rota alternativa, ou mesmo iniciar o processo de descoberta de rota no caso de não existir uma rota alternativa quando é detectada uma falha na rota principal.

Para determinar a estabilidade de um caminho durante a descoberta de rota, é utilizado o valor da potência do sinal recebido verificado nos nós da rede aquando a recepção das mensagens de controlo (RREQ e RREP). Este valor é obtido directamente da camada física, através de uma interacção cross-layer. Ao obter para cada caminho encontrado o valor da potência do sinal recebido mínima durante o processo de descoberta de rota, pretende-se seleccionar no nó fonte para a transmissão dos dados, o caminho em que no conjunto das ligações entre vizinhos se verificou mais estabilidade. Isto é, entre todos os caminhos encontrados será seleccionado o que contém o maior valor da potência do sinal recebido. Desse modo pretende-se que quando os nós pertencentes a esse caminho se movimentarem durante a transmissão dos dados entre a fonte e o destino, a probabilidade será inferior de estes se descolarem o suficiente de forma a ficarem fora do alcance de transmissão, não ocorrendo assim tantas quebras de ligações (rotas).

Antes de um nó fonte efectuar o pedido de rota para um determinado destino, se o tipo de tráfego que pretende transmitir, exigir um requisito de atraso fim a fim máximo, esse requisito terá de ser incluído na mensagem RREQ (Tabela 3.4) no campo delayReq.

Para além do atraso fim a fim máximo requisitado, a mensagem RREQ terá de conter a informação do atraso que ocorre no envio de uma mensagem entre o nó fonte e o vizinho, incluído no campo delayAcc (isto é necessário para que durante a descoberta de rota este atraso possa estar incluído no atraso acumulado no caminho encontrado), sendo de seguida procedido a transmissão da mensagem RREQ em broadcast. Esta mensagem será retransmitida nos restantes nós que vão formar o caminho, que acrescentam também a informação do atraso para o envio de uma mensagem para o vizinho, para que seja acumulado o atraso nó a nó e incluído na mensagem RREQ no campo delayAcc, até que possivelmente possa ser descoberto um caminho que cumpra o requisito de QoS e obter o valor do atraso verificado nesse caminho.

É armazenado em cada nó o atraso que ocorre na transmissão das mensagens de controlo, com essa informação realiza-se uma estimativa do próximo atraso que ocorrerá para que um pacote seja entregue no vizinho, utilizado como próximo salto para o destino. A estimativa desse atraso é designado de delayNode (Equação (1)) e utiliza a informação do atraso verificado nas últimas cinco mensagens. Este atraso é determinado desde o envio de uma mensagem de controlo no protocolo de encaminhamento até esta ser transmitida na camada física para o meio de comunicação.

Desta forma, o atraso verificado em cada nó para o envio de uma menagem para o vizinho inclui,

o atraso de processamento, o atraso na fila de espera, o tempo de contenção no acesso ao meio e acrescenta-se o atraso de transmissão, calculado segundo a Equação (2). Este atraso não irá conter o atraso de propagação, visto que este dependerá da distância entre o nó actual e o vizinho, e comparativamente aos restantes atrasos é muito inferior, o que acaba por ser insignificante e não influenciará o cálculo realizado.

- Cálculo do atraso que ocorre num nó da rede para a transmissão de um pacote:

$$delayNode = atraso\ de\ processamento + atraso\ na\ fila\ de\ espera + atraso\ de\ contenção + atraso\ de\ transmissão \quad (1)$$

- Cálculo do atraso de transmissão:

$$atraso\ de\ transmissão = tamanho\ do\ pacote\ /\ taxa\ de\ transferência \quad (2)$$

Com a estimativa realizada do atraso que ocorrerá no envio do próximo pacote, através da informação do atraso calculado e armazenado em cada nó no envio das ultimas cinco mensagem de controlo, é possível durante a retransmissão da mensagem RREQ nos nós da rede durante a descoberta e manutenção de rotas, sempre que necessário, utilizar este atraso para determinar o atraso do caminho de nós disjuntos encontrado (delayPath) e verificar se este cumpre o requisito de atraso fim a fim máximo requisitado (delayReq).

Em seguida explica-se mais detalhadamente a utilização do atraso determinado em cada nó (delayNode), assim como a utilidade de todos os campos contidos nas mensagens de controlo RREQ e RREP, utilizados para obter o atraso dos caminhos de nós disjuntos encontrados que cumpram o requisito de QoS, recorrendo ao exemplo da Figura 3.5 e da Figura 3.6.

O exemplo da Figura 3.5 representa o início do processo de descoberta de rota no nó fonte (representado a cor vermelha) com a transmissão em broadcast de uma mensagem RREQ e a sua retransmissão nos nós intermédios, até que a mensagem chega a um nó intermédio com conhecimento de uma rota válida para o destino. E na Figura 3.6, apresenta-se um exemplo que corresponde à resposta nos nós intermédios ao pedido de rota do nó fonte. Sendo que as tabelas apresentadas por baixo de cada nó, correspondem à informação acrescentada às mensagens de controlo antes de serem retransmitidas.

O nó fonte, como se pode ver no exemplo da Figura 3.5, antes de efectuar o envio do pedido de rota, preenche o campo delayAcc da mensagem RREQ com o atraso armazenado no nó, correspondente à estimativa do atraso que ocorrerá durante a transmissão da mensagem RREQ para o vizinho (delayNode), e também o atraso fim a fim máximo requisitado pela aplicação, inserido no campo delayReq da mensagem.

O vizinho do nó fonte (nó A), antes de retransmitir a mensagem, acrescenta ao campo delayAcc o seu atraso delayNode, desta forma este campo irá acumular o atraso nó a nó, contendo no fim o atraso verificado no caminho encontrado.

Durante a descoberta de rota, é utilizado um mecanismo de controlo de admissão, sempre que se verifique o não cumprimento do atraso fim a fim máximo requisitado pela aplicação (delayReq) o pacote é descartado, permitindo encontrar outras rotas que cumpram o requisito de QoS. Para verificar se o requisito de QoS está a ser cumprido durante a descoberta de rota, um nó ao receber o pacote RREQ, efectua a comparação entre o atraso acumulado (delayAcc) e o atraso fim a fim máximo requisitado (delayReq), em que o atraso acumulado terá de ser inferior ao atraso requisitado (Equação (3)). Sempre que não se verifique esta condição, o pacote é descartado.

- Verificação se o atraso ocorrido até ao momento durante a retransmissão da mensagem RREQ entre os nós intermédios, cumpre o requisito de atraso fim a fim máximo requisitado:

$$RREQ.delayAcc < RREQ.delayReq \quad (3)$$

No caso de o caminho percorrido até ao momento cumprir o requisito de QoS, são colocados na mensagem RREQ a informação sobre o atraso acumulado (Equação (4)) e a potência do sinal recebido mínima (Equação (5)) verificada nesse caminho.

Para além de acumular o atraso nó a nó, desde que a fonte envia a mensagem de RREQ e durante a sua retransmissão (delayAcc), é necessário também actualizar o campo delayToOrigin, uma vez que como descrito na Secção 3.1.2, no processo de descoberta de rota durante a retransmissão do RREQ é criada ou actualizada nos nós intermédios a entrada nas suas tabelas de encaminhamento da rota para o nó fonte, sendo colocado o atraso correspondente, que será o atraso delayToOrigin, como apresentado no exemplo da Figura 3.5 nas tabelas por baixo de cada nó intermédio.

Durante a transmissão da mensagem RREQ, o valor obtido da potência do sinal recebido no nó

(RxSSRead) é comparado com o valor contido na mensagem (RxSSPath), apenas no caso em que o valor lido seja inferior ao contido no pacote, é que se altera o campo RxSStPath da mensagem RREQ por esse valor lido, caso contrário mantém-se o valor previamente existente na mensagem RREQ da potência do sinal recebido mínima do caminho percorrido até ao momento (Equação (5)). Com a informação das métricas actualizadas o pacote RREQ é retransmitido em *broadcast*, para continuar a procura de uma rota para o destino.

- Determinação do atraso acumulado durante a retransmissão da mensagem RREQ:

$$RREQ.delayAcc = REQ.delayAcc + delayNode \quad (4)$$

- Determinação da potência do sinal recebido mínima durante a retransmissão da mensagem RREQ nos nós intermédios:

$$RREQ.RxSSPath = min(RREQ.RxSSPath, RxSSRead) \quad (5)$$

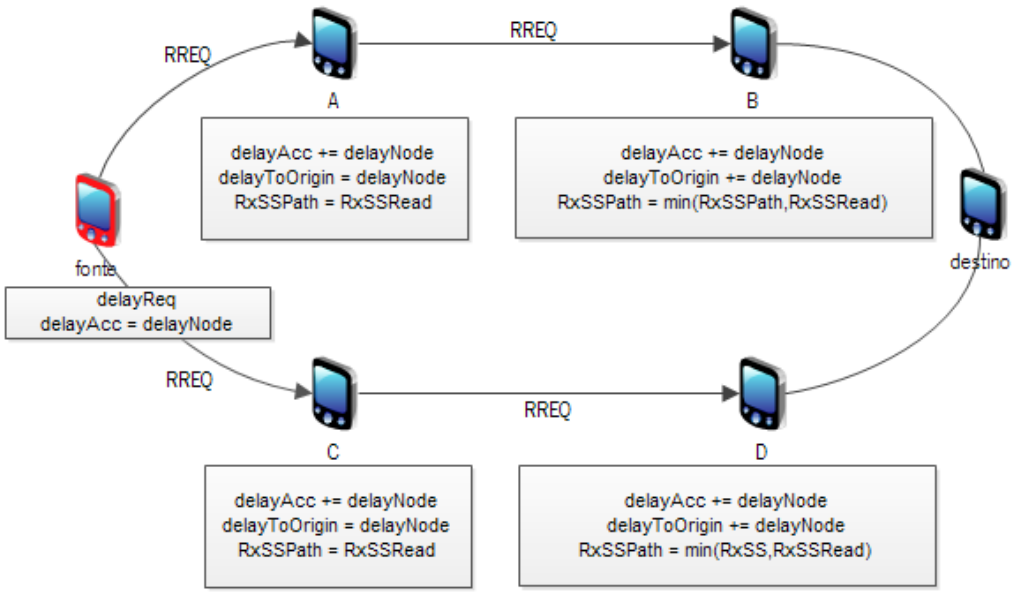


Figura 3.5: Pedido de rota e determinação do atraso fim a fim e da potência do sinal

A resposta ao pedido de rota, quer seja feita no nó destino, ou por um nó intermédio que detém uma rota válida para o mesmo, com a informação contida no campos da mensagem RREQ apresentados, actualizados nó a nó durante a sua retransmissão nos nós intermédios no caminho percorrido, desta forma quando esta chega a um nó intermédio com conhecimento de uma rota válida para o destino, contém a informação do atraso fim a fim (delayPath) e a potência do sinal recebido mínima (RxSSPath) do caminho encontrado. Essa informação vai contida na resposta (pacote RREP) transmitida em unicast para o nó fonte, utilizando as rotas contidas nos nós intermédios.

Durante a transmissão da resposta para o nó fonte, os nós intermédios actualizam a entrada na tabela de encaminhamento para o nó destino, e retransmitem o pacote RREP para o nó fonte. De forma a actualizar a entrada na tabela de encaminhamento para o nó destino em todos os nós pertencentes ao caminho encontrado, durante a retransmissão do RREP, a informação do atraso fim a fim e da potência do sinal recebido mínima é também determinada nó a nó e inserida nos campos delayToDst e RxSSPathToDst. Deste modo, se estes nós receberem algum pedido de rota de outro nó fonte para esse destino, podem verificar se o atraso fim a fim máximo requisitado pode ser cumprido, assim como informar da estabilidade deste caminho, ou seja, contêm a informação necessária para poderem responder ao pedido de rota.

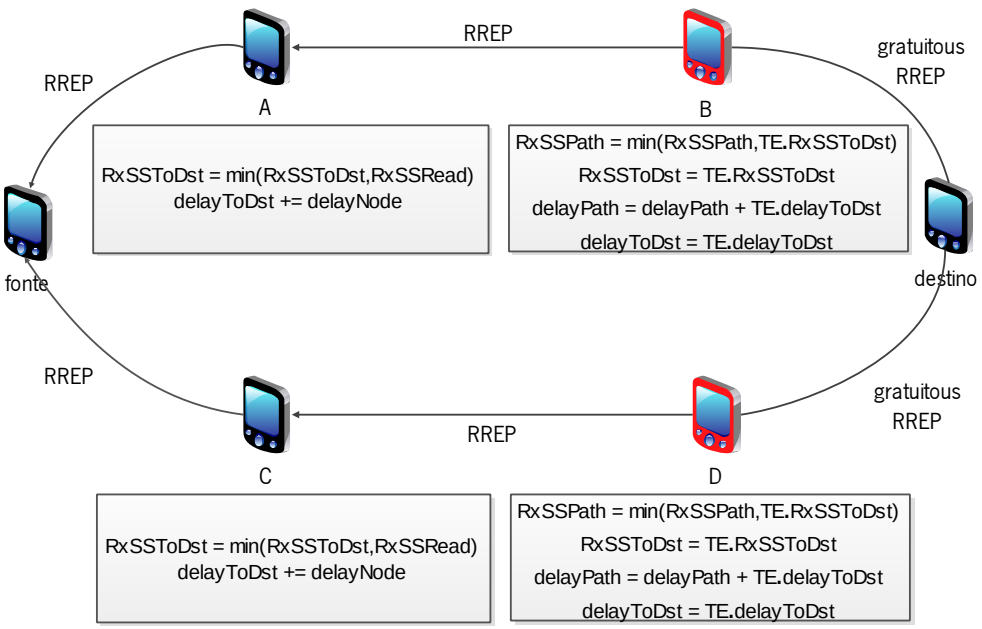


Figura 3.6: Resposta ao pedido de rota e cálculo das métricas utilizadas

O exemplo da Figura 3.6 demonstra a resposta ao pedido de rota descrito anteriormente, com a actualização dos campos na mensagem RREP, apresentados nas tabelas por baixo de cada nó. É possível verificar no exemplo apresentado, que os nós intermédios B e D respondem ao pedido de rota do nó fonte para o destino. Sendo o RxSSPath determinado através da verificação de qual é o valor mínimo entre o valor do RxSSPath obtido até ao momento que consta na mensagem RREQ, e o valor que consta na tabela de encaminhamento para o destino (TE.RxSSToDst), Equação (6). Para a determinação do atraso realiza-se também uma consulta da tabela de encaminhamento, em que o atraso do caminho encontrado (delayPath) e enviado na resposta, será o atraso verificado até ao momento (incluído no campo delayPath da mensagem RREQ) mais o atraso verificado para o nó destino (TE.delayToDst), Equação (7).

- Determinação da potência do sinal recebido mínima do caminho encontrado e enviada na resposta para o nó fonte (mensagem RREP) por um nó intermédio com uma rota válida para o destino:

$$RREP.RxSSPath = min(RREQ.RxSSPath, TE.RxSSToDst) \quad (6)$$

- Determinação do atraso fim a fim no nó intermédio que responde ao pedido de rota (envio da mensagem RREP):

$$RREP.delayPath = RREQ.delayPath + TE.delayToDst \quad (7)$$

3.2.4. Selecção do caminho

Entre os caminhos encontrados de nós disjuntos que cumpriram o requisito do atraso fim a fim máximo, com a informação da estabilidade de cada caminho, baseada na potência do sinal recebido mínima de cada caminho, é eleita a rota mais estável para transmissão. Apenas quando é verificada a existência de caminhos com uma estabilidade idêntica, dá-se preferência ao caminho com menor atraso fim a fim.

3.2.5. Manutenção das rotas

A topologia neste tipo de redes é dinâmica e imprevisível, com a frequente mobilidade dos nós, podem ocorrer a qualquer momento quebras nos caminhos encontrados.

Após a descoberta de rota desencadeada no nó fonte, todos os nós pertencentes a este caminho, armazenam na sua tabela de encaminhamento a respectiva informação das rotas para o nó fonte e destino. Quando um nó intermédio pertencente à rota, verifica que o nó de próximo salto utilizado para atingir o destino já não se encontra no seu alcance de transmissão, ou seja, detecta a quebra de ligação para o nó vizinho, terá de enviar um pacote Route Error (RERR) de forma a notificar o nó fonte, que aquele nó utilizado para atingir determinado destino ficou indisponível. Este pacote percorre os nós intermédios utilizados neste caminho, que fazem a remoção da entrada da tabela de encaminhamento para o destino que utiliza como próximo salto o nó de quem enviou o RERR, e depois retransmite-o para que este possa chegar ao nó fonte. O nó fonte ao receber o pacote RERR remove também da sua tabela de encaminhamento a rota correspondente, e caso esta rota esteja definida como rota principal e contenha rotas alternativas, procede à comutação para a rota alternativa mais estável entre as existentes. Se não existir nenhuma rota alternativa, terá de iniciar o processo de descoberta de rota. Caso a notificação de quebra de ligação seja sobre uma rota alternativa, apenas é realizada a remoção dessa entrada da tabela de encaminhamento, mantendo-se a rota principal para transmissão.

Nos caminhos encontrados durante o processo de descoberta de rota, executa-se uma verificação de incumprimento do requisito de QoS. São utilizados as mensagens InspectPath e ReplyInpectPath, transmitidos por esses caminhos, idênticas às mensagens de RREQ e RREP, de forma a actualizar as métricas e verificar se o requisito de atraso fim a fim máximo continua a ser cumprido.

Com a utilização de métricas dinâmicas para a decisão de encaminhamento, podem ocorrer constantes alterações em função do estado da rede. Neste sentido, a troca de mensagens necessita de garantir que a informação do estado da rede está actualizada, sem ser efectuada sem levar ao congestionamento da rede, de forma a não prejudicar o desempenho ou mesmo a utilidade do protocolo.

O procedimento utilizado para esta verificação é semelhante ao utilizado no controlo de admissão do processo de descoberta de rota, em que é verificado se o atraso acumulado é inferior ao atraso fim a fim máximo requisitado durante a retransmissão da mensagem InspectPath nó a nó. Quando é

verificado que o requisito de QoS não está a ser cumprido, é enviado um pacote RERR para notificar o nó fonte que aquele caminho não poderá ser utilizado, visto que não cumpre o requisito de atraso fim a fim. Os nós intermédios na recepção do pacote RERR executam o procedimento anteriormente descrito na recepção deste tipo de pacotes, ou seja, fazem a remoção da entrada na tabela de encaminhamento para o destino que utilize como próximo salto o nó que enviou o pacote e retransmitem-no. Durante o encaminhamento dos pacotes InspectPath nó a nó, ao percorrerem os caminhos encontrados durante a descoberta de rota, é realizada também a actualização das métricas utilizadas para seleccionar a rota principal, do atraso fim a fim e a potência do sinal recebido mínima de cada caminho.

Capítulo 4

Implementação

Este capítulo servirá para descrever os passos necessários na implementação dos protocolos de encaminhamento desenvolvidos no âmbito deste trabalho. Para a implementação e testes dos protocolos concebidos recorreu-se à simulação. O objectivo ao implementar um modelo de simulação é imitar o sistema real, a fim de estudar o seu comportamento. A simulação dos protocolos propostos faz com que seja possível a realização de vários testes, que seriam impraticáveis num cenário real.

Será apresentada a implementação de ambos os protocolos propostos: o protocolo Ad hoc On-Demand Multipath Routing (AMR) e o protocolo Ad Hoc QoS On-Demand Multipath Routing with Route Stability (QMRS), tendo sido ambos desenvolvidos no Network Simulator 3 (ns-3) [33]. É incluído inicialmente neste capítulo, um esclarecimento e discussão, da razão por se ter optado por utilizar a plataforma de simulação ns-3 em detrimento de outros simuladores.

Após realizada a introdução ao simulador, descrever-se-á a o primeiro protocolo implementado: o protocolo AMR. Esta implementação foi aplicada no segundo protocolo, o protocolo QMRS, servindo de base à sua implementação. Desta forma, na descrição do protocolo QMRS, descrever-se-ão apenas as alterações necessárias para a sua implementação. Será descrito para cada protocolo, a implementação dos vários mecanismos existentes apresentados no capítulo anterior (Capítulo 3).

4.1. Plataforma de desenvolvimento

Para criar uma plataforma de testes eficiente, antes de dar início à implementação da solução proposta, realizou-se uma comparação entre alguns dos simuladores de redes, de forma a chegar a uma decisão de qual o mais adequado para a implementação e avaliação da solução proposta.

Entre os vários simuladores de redes, dá-se destaque aos simuladores *open source* Network Simulator 2 (ns-2) [34], Network Simulator 3 (ns-3) [33] e o OMNeT++ [35], por serem simuladores muito usados a nível académico e na comunidade de investigação na área das redes.

É realizada uma análise ao funcionamento de cada simulador e pretende-se averiguar quanto à eficiência da sua utilização, dando algum destaque no que diz respeito ao objecto de estudo desta dissertação, as redes móveis ad hoc. Será referindo quais as vantagens e desvantagens em cada um dos simuladores durante o período de execução e dá-se referência a algumas das ferramentas de análise existentes, durante ou após o período de execução da simulação.

O OMNET++ é um simulador discreto por eventos, desenvolvido por Andras Varga. Os módulos são desenvolvidos em C++ e são estruturados por uma linguagem de alto nível designada de network definition language (NED), que permite agregar componentes individuais, formando os *compound models*. Oferece um IDE baseado no Eclipse e um ambiente gráfico sofisticado que se torna muito útil para debugging, em que permite durante o período de simulação inspeccionar cada módulo e verificar a ocorrência de cada evento. Este simulador tem facilidade na criação automática de gráficos que permitem analisar a simulação realizada.

Para as redes móveis ad hoc, pode-se utilizar frameworks como a Mobility Framework, INETMANET ou a INET framework, onde contêm vários protocolos de encaminhamento, modelos de mobilidade, etc. Alguns dos factores negativos verificados neste simulador, foi a documentação das frameworks mencionadas, ainda se encontrar em fase de desenvolvimento, e o código implementado de alguns protocolos comparativamente a outros simuladores, não estar tão bem organizado e comentado.

O ns-2 é um simulador discreto por eventos. Entre os três simuladores referidos, o ns-2 é o que tem sido mais utilizado e o mais popular entre a comunidade de investigação nos últimos anos [36], este simulador tem capacidade de simular redes com fios e redes sem fios, inclui vários protocolos de transporte, protocolos de encaminhamento e vários tipos de aplicações. Após a execução da simulação é possível utilizar várias ferramentas de análise sobre os dados obtidos durante o período de simulação. Para uma visualização gráfica da rede pode ser utilizado o Network Animator (NAM). Costumam ser usadas ferramentas como o xgraph e o gnuplot para uma análise gráfica de alguns dados obtidos durante a simulação, etc.

A existência de uma boa documentação, a partilha de informação e a contribuição entre o vasto número de utilizadores, são uma grande vantagem na utilização deste simulador. No entanto, encontram-se alguns problemas neste simulador, um dos quais por ser desenvolvido em C++ e fornecer a interface de simulação através de OTcl, uma extensão da linguagem Tcl orientada a

objectos, a utilização de duas linguagens torna o *debugging* mais complexo.

Para além deste factor menos favorável em relação aos outros simuladores, alguns testes realizados em [37], indicam que para as redes móveis ad hoc, este não será o simulador mais adequado a utilizar, verificam-se vários problemas a nível de eficiência de processamento e de memória, assim como problemas de escalabilidade, com o aumento do número de nós neste tipo de redes.

Por último referencia-se o simulador Network Simulator 3, um simulador discreto por eventos, distribuído sob a licença GNU GPLv2. Foi desenvolvido com o propósito de melhorar alguns dos problemas do ns-2, alguns dos quais foram mencionados, e espera-se que o ns-3 eventualmente possa vir a substituir o ns-2.

Assim como o seu antecessor, o ns-3 é desenvolvido em C++, no entanto já não usa a linguagem OTcl, removendo assim o problema da utilização de duas linguagens de programação, verificado no ns-2. A simulação pode ser totalmente implementada em C++ e pode ser utilizada opcionalmente a linguagem de programação Python.

Este simulador não consiste numa alteração ao ns-2, foi construído de raiz, apenas aproveitou algum do código existente no ns-2 que se encontrava maioritariamente em C++, como por exemplo, o protocolo de encaminhamento OLSR.

Este simulador contém uma boa documentação, e o código desenvolvido em C++ encontra-se bem organizado e comentado [36], facilitando aos utilizadores uma rápida adaptação. Nos testes realizados em [37], comparativamente com o ns-2, o simulador ns-3 demonstrou mais eficiência de processamento e de memória, verificou-se com o aumento de número de nós na rede boas características de escalabilidade, um dos problemas verificados no ns-2. Quanto às ferramentas de análise, é possível uma visualização gráfica da simulação utilizando o NetAnim, que utiliza um ficheiro de *trace* XML, que contém a informação necessária para a visualização gráfica obtida durante o período de simulação. Também é possível observar a simulação durante o período de execução, com a ferramenta PyViz, muito útil para debug, como por exemplo, verificar se o desempenho do modelo de mobilidade corresponde ao esperado, perda de pacotes, etc. Para uma análise gráfica sobre os dados obtidos durate a simulação, podem ser utilizadas ferramentas como o xgraph e o gnuplot. Pode-se utilizar também vários analisadores de ficheiros de trace, como por exemplo o wireshark, para análise a ficheiros de trace pcap. É possível também configurar no ns-3, assim como no ns-2, um IDE C++

como o Eclipse ou o Netbeans, para facilitar os utilizadores no desenvolvimento de software.

Após a análise realizada aos simuladores ns-2, ns-3 e o OMNeT++, foi construída uma tabela com o intuito de atribuir uma nota (de 0 a 5) a cada um dos simuladores, relativamente a alguns dos aspectos referidos acima.

Tabela 4.1: Visão comparativa dos simuladores de redes ns-2, ns-3 e OMNeT++

	ns-2	ns-3	OMNeT++
Linguagem programação	3	4	4
Eficiência no tempo de simulação	3	5	4
Eficiência de Memória	3	5	4
Escalabilidade	3	4	4
IDE	2	2	4
Debugging e Tracing	3	3	4
Visualização da simulação	3	3	5
Documentação, partilha de informação e contribuições.	5	4	2
Organização do código	3	5	3
Variedade de modelos disponíveis	4	3	3
Nota Final	3.2	3.8	3.7

4.2. Protocolo de encaminhamento de múltiplos caminhos

O primeiro passo para a implementação do protocolo Ad Hoc Multipath Routing (AMR) proposto, foi o de acrescentar ao simulador ns-3 um novo módulo, tendo sido usado de base parte do protocolo Ad Hoc On-Demand Distance Vector (AODV), já existente no simulador ns-3. Em seguida, o segundo passo foi adicionar mais funcionalidades e mecanismos, para possibilitar ao protocolo implementado, desempenhar os objectivos funcionais pretendidos e apresentados anteriormente, tendo sido utilizado como referência o protocolo Ad Hoc On-Demand Multipath Distance Vector (AOMDV) existente no simulador ns-2.

O protocolo ARM, descrito com maior detalhe no Capitulo 3, com a implementação efectuada, irá possibilitar encontrar múltiplos caminhos de nós disjuntos entre uma fonte e um destino. E permite

que um nó detecte e notifique o nó fonte, quando ocorre uma quebra de ligação entre dois nós pertencentes aos caminhos encontrados. No caso de ser uma quebra de uma ligação pertencente ao caminho principal usado para transmissão entre a fonte e o destino, será possível efectuar a comutação para uma rota alternativa, para que a fonte possa dar continuidade à transmissão dos dados, sem a necessidade de iniciar um novo processo de descoberta de rota.

Em seguida descreve-se as implementações necessárias, desde a alteração à estrutura da tabela de encaminhamento e os métodos usados durante os mecanismos descritos, de forma a que o protocolo AMR realize os objectivos pretendidos e obtenha o desempenho desejado.

4.2.1. Estruturas utilizadas

Sendo que o protocolo AMR utiliza a informação contida nas tabelas de encaminhamento de cada nó para o processo de encaminhamento, foi necessário alterar a constituição da tabela de encaminhamento existente no AODV (Tabela 4.2), de forma a possibilitar armazenar a informação de múltiplas rotas para o mesmo destino (Tabela 4.3).

Um nó quando pretende iniciar a transmissão para determinado destino, é efectuado um pedido de rota ao protocolo de encaminhamento para o respectivo endereço IP do destino, de forma a encontrar a rota correspondente. Para isso é utilizado o método RouteOutput contido na classe RoutingProtocol do protocolo, que retorna um apontador para a rota contida na tabela de encaminhamento (Ipv4Route). Este método faz uma consulta à tabela de encaminhamento para verificar se existe uma rota para o destino, no caso de não conter uma rota válida para o destino, inicia-se o processo de descoberta de rota para tentar encontrar uma rota válida. Processo esse que será descrito na próxima secção (Secção 4.2.2).

Em seguida apresentam-se as tabelas de encaminhamento do protocolo AODV (Tabela 4.2) e do protocolo AMR (Tabela 4.3), para demonstrar as alterações efectuadas, contendo apenas a informação mais relevante.

Tabela 4.2: Tabela de encaminhamento do protocolo AODV

destination	sequence number	hop count	next hop	lifetime

Tabela 4.3: Tabela de encaminhamento do protocolo AMR

destination	sequence number	advertised hop count	next hop	last hop	hop count	lifetime	list of alternative routes			
							next hop	last hop	hop count	lifetime
							next hop	last hop	hop count	lifetime

A Tabela 4.3 apresenta os campos pertencentes a uma entrada na tabela de encaminhamento do protocolo AMR, que durante o processo de descoberta de rota, permite armazenar a informação da rota principal (corresponde à primeira rota encontrada, que será utilizada para transmissão dos dados), assim como a informação das restantes rotas alternativas.

Para a construção da tabela, efectuaram-se as alterações necessárias na classe RoutingTable, esta classe permite armazenar toda a informação da tabela de encaminhamento de cada nó e os métodos para operar sobre as entradas criadas. As entradas na tabela são colocadas num map (std::map<Ipv4Address, RoutingTableEntry> m_ipv4AddressEntry) em que o primeiro campo (key) será o endereço IP do destino e o segundo campo objectos do tipo entrada (classe RoutingTableEntry), que possibilita armazenar a informação de uma entrada e os respectivos métodos para realizar operações na entrada. Sendo que para cada entrada da tabela de encaminhamento, a informação da rota principal será armazenada no campo m_ipv4Route. Em que a classe Ipv4Route contém alguns dos campos apresentados na tabela de encaminhamento (Tabela 4.3) e os métodos correspondentes, para permitir a sua inserção, remoção e consulta. A informação das rotas alternativas será armazenada num vector (std::vector<AmrRoute> m_routeList). Em que na classe AmrRoute estão definidos todos os campos necessários e os métodos correspondentes de forma a realizar as operações necessárias sobre as rotas alternativas.

4.2.2. Descoberta de rota

Em seguida descrevem-se as decisões de implementação mais relevantes nos métodos utilizados para o processo de descoberta de rota.

Resumidamente o processo de descoberta de rota é desencadeado quando um nó pretende transmitir para determinado destino e não contém uma rota válida na sua tabela de encaminhamento para esse destino. Sendo que o início da descoberta consiste no envio de um pedido de rota por parte do nó fonte, na tentativa de que um nó com o conhecimento de uma rota válida para o destino responda ao pedido, e que essa resposta seja encaminhada até ao nó fonte. O nó fonte ao receber a resposta com a informação da rota para o destino, poderá dar início à transmissão dos dados.

Durante a descoberta de rota ocorrem trocas de pacotes de controlo entre os nós da rede, de forma a possibilitar encontrar múltiplos caminhos de nós disjuntos entre o nó fonte e o destino.

A seguir são descritos os métodos mais importantes utilizados durante a descoberta de rota, sendo invocados outros métodos que realizam tarefas de menor dimensão durante a execução destes métodos principais, que permitem ao protocolo AMR realizar os objectivos descritos.

- **Método SendRequest**

Este método deverá ser invocado quando um nó pretende transmitir para determinado destino, e verifica que não contém na sua tabela de encaminhamento, uma rota válida para esse destino.

Na descrição deste método, aproveita-se também para explicar a implementação necessária para aplicar alguns dos conceitos importantes e utilizados no protocolo, de forma a possibilitar encontrar rotas para o destino e tomar decisões de encaminhamento, de modo a evitar a ocorrência de ciclos no encaminhamento e controlar a difusão dos pedidos de rota, sendo que para isso são utilizados vários métodos invocados durante a realização do pedido de rota.

Para controlar o excesso de pedidos de rota consecutivos, é necessário controlar o número de mensagens de Route Request (RREQ) que um nó poderá transmitir para tentar obter uma rota para o destino. Para tal utilizam-se várias variáveis e temporizadores que controlam este aspecto. A cada pedido de rota é incrementada a variável m_rreqCount, e sempre que se atinge o número máximo de mensagens RREQ por segundo (RreqRateLimit) a transmissão da mensagem será adiada.

Antes do envio da mensagem de RREQ, é efectuada uma consulta na tabela de encaminhamento, para tentar obter a ultima informação conhecida do destino, através da informação que o nó possa

conter numa entrada inválida para esse destino. Caso possua uma entrada colocada como inválida para o destino, a informação do último sequence number do destino conhecido é enviado na mensagem RREQ, para evitar durante a descoberta de rota obter uma resposta de um nó da rede que contenha uma rota obsoleta para o destino, podendo originar ciclos no encaminhamento. No caso de não conter uma rota inválida, envia-se na mensagem RREQ que o sequence number do destino não é conhecido (rreqHeader.SetUnknownSeqno(true)).

Após verificada a informação que o nó fonte detém sobre o destino, é criada uma entrada na tabela de encaminhamento para esse destino, sendo indicado nessa entrada de que se está à espera de uma resposta para a validar (através de uma flag colocada no modo "INSEARCH"), sendo que o tempo de espera varia conforme o número de pedidos consecutivos de rota para este destino (m_rreqCount). A mensagem RREQ é transmitida em broadcast, (será explicado nos próximos métodos como e quando esta mensagem é retransmitida nos nós vizinhos) para que possivelmente se adquira uma resposta com a informação da rota para o destino dentro do tempo de espera estipulado.

Para o cálculo do tempo de espera utiliza-se a informação de qual o número de pedidos consecutivos até ao momento, que multiplica por uma estimativa do tempo que poderá demora a chegar a resposta (NetTraversalTime), Equação (1). A variável m_addressReqTimer contém para cada destino o temporizador respectivo, para se poder controlar o tempo de espera por uma possível resposta ao pedido de rota, ou seja, a recepção de uma mensagem Route Reply (RREP) com a informação da rota para o destino, correspondente ao pedido efectuado pela fonte.

$$rt.GetRreqCnt() * NetTraversalTime \quad (1)$$

A Equação (1) corresponde ao tempo de espera por uma resposta ao pedido de rota, sendo o número de pedidos (m_rreqCount) obtido ao consultar a entrada da tabela de encaminhamento (rt.GetRreqCnt()) para o destino, ou seja, a entrada criada (actualizada) quando a primeira mensagem RREQ é transmitida. O tempo estimado de NetTraversalTime é calculado segundo a Equação (2).

$$2 * NodeTraversalTime * NetDiameter \quad (2)$$

A variável NodeTraversalTime é uma estimativa da média do tempo que demora a transmissão de um pacote para o vizinho, incluindo o atraso de na fila de espera, o tempo de processamento e o tempo

de transferência. Sendo a variável NetDiameter o número máximo de saltos entre dois nós da rede.

A variável NetDiameter é utilizada também para aplicar a técnica de "expanding ring search", que consiste em utilizar um TTL de NetDiameter de forma a controlar a difusão das mensagens RREQ, para não sobrecarregar toda rede com estes pacotes de controlo.

- **Método RecvRequest**

Este método deverá ser invocado quando a mensagem recebida no nó é uma mensagem de RREQ. Isto verifica-se no método RecvAmr da classe RoutingProtocol, que ao confirmar que a mensagem é do tipo AMRTYPE_RREQ invoca este método (recvRequest), com os seguintes argumentos: a mensagem recebida, o endereço IP do nó que recebeu a mensagem (receiver) e o endereço IP do nó que enviou a mensagem (sender).

A mensagem RREQ recebida no método RecvRequest será analisada, ou seja, verificam-se os campos que a mensagem contém (Tabela 3.1), e é descartada (não será retransmitida) quando se verificam as seguintes situações:

- O nó que recebe a mensagem RREQ é o próprio nó fonte, ou seja, a mensagem foi retransmitida por um nó vizinho e o nó fonte recebeu o seu próprio pedido de rota, logo descarta a mensagem. Isto verifica-se ao comparar a informação do endereço IP do nó que recebeu a mensagem, com o endereço IP do nó fonte contido na mensagem RREQ.
- Um nó intermédio que recebe uma mensagem RREQ e verifica que a mensagem (mesma informação) já foi anteriormente recebida no nó, não retransmite a mensagem. Isto é possível verificar ao analisar na mensagem os campos: endereço IP do nó fonte e o requestID. Estes dois campos permitem identificar unicamente cada mensagem.

  Utiliza-se o método idGetCache, que recebe como argumento o endereço IP e o requestID, para fazer a verificação descrita. E o método idInsertCache para o caso de ser a primeira vez que a mensagem é recebida no nó, armazenar a informação do endereço IP e requestID no vector m_bidCache.

Um vizinho do nó fonte na recepção da mensagem RREQ, antes de a retransmitir, terá de acrescentar o seu endereço IP no campo firstHop da mensagem, para isso é invocado o método SetFirstHop em que recebe como argumento o seu endereço (rreqHeader.SetFirstHop(receiver)).

Este campo é necessário para que durante a retransmissão da mensagem RREQ entre os nós

intermédios que vão constituir o caminho encontrado, possam criar entradas na tabela de encaminhamento para o nó fonte. Caminho esse que servirá para a retransmissão da resposta (mensagem RREP) ao pedido de rota, para que possa chegar ao nó fonte.

A entrada da tabela de encaminhamento do protocolo AMR contém os campos apresentados na Tabela 4.3. A classe RoutingTableEntry é responsável por armazenar a informação das entradas, ao criar uma instância desta classe pode-se utilizar o construtor com parâmetros que recebe os campos todos da entrada, ou utilizar os métodos existentes que permitem actualizar a entrada.

Neste método (RecvRequest) quando se verifica que o nó não contém nenhuma entrada (ou uma entrada inválida) para o nó fonte na sua tabela de encaminhamento, é criada (ou actualizada) a entrada.

Para criar (ou actualizar) a entrada, utilizam-se os seguintes métodos para obter a informação da rota encontrada para a fonte:

- Destination: será o endereço IP da fonte, contido na mensagem RREQ no campo m_origin. Para obter este endereço na mensagem, utiliza-se o método rreqHeader.GetOrigin();
- Sequence number: corresponde ao sequence number da fonte, será obtido através do método rreqHeader.GetOriginSeqno();
- NextHop: será o endereço IP do nó que enviou a mensagem. Corresponde à variável sender, recebida como argumento no método RecvRequest.
- lastHop: corresponde ao último salto para o nó fonte, que permite identificar o caminho encontrado entre o nó actual e a fonte. Utiliza-se o método rreqHeader.GetFirstHop() para obter o endereço IP do nó vizinho da fonte.
- HopCount: conterá o número de saltos do caminho entre o nó actual e a fonte. Utiliza-se o método rreqHeader.GetHopCount() para obter o numero de saltos acumulado até ao momento, e incrementa-se o ultimo salto realizado para o nó actual.

Na recepção da mensagem RREQ quando se verifica que o nó já contém uma entrada na sua tabela de encaminhamento para a fonte, apenas será actualizada no caso de se verificarem as seguintes situações:

- Se a mensagem RREQ contém um sequence number da fonte mais recente que o

armazenado na entrada da tabela de encaminhamento. Neste caso removem-se todas as encontradas anteriormente para a fonte, e actualiza-se a entrada da rota principal, segundo a descrição acima referida para a actualização da entrada para a fonte.

- Se a mensagem RREQ contém um sequence number da fonte igual ao da entrada da tabela de encaminhamento, tenta-se inserir a nova rota na tabela. Esta nova entrada apenas será inserida se corresponder a um caminho de nós disjuntos e se o número de rotas armazenadas na tabela para a fonte não atingiu o limite (armazena-se no máximo até três rotas). Utiliza-se o método NewDisjointPath para verificar se o caminho novo é de nós disjuntos, este método recebe como argumento o endereço IP do nó que enviou a mensagem e o endereço do nó de ultimo salto para a fonte desse caminho (toOrigin.NewDisjointPath(sender, rreqHeader.GetFirstHop()).

Após criar (actualizar) a rota para a fonte, verifica-se se o nó actual que recebe a mensagem de RREQ irá responder ao pedido de rota com o envio de uma mensagem RREP para o nó fonte.

Apenas será enviada a mensagem RREP se forem verificadas as seguintes situações:

- O nó actual que recebe o pedido de rota para o destino é o próprio nó destino. Para verificar, utiliza-se o método IsMyOwnAddress que recebe como argumento o endereço IP do destino, obtido com o método rreqHeader.GetDst(). Para processar a resposta, é invocado o método SendReply, descrito de seguida, que recebe como argumento a mensagem RREQ e a entrada da tabela de encaminhamento para a fonte.
- O nó actual é um nó intermédio com conhecimento de uma rota válida para o destino e o sequence number do destino contido na entrada da tabela de encaminhamento é igual ou superior ao contido na mensagem RREQ. Utilizam-se os seguintes métodos que permitem realizar as verificações descritas: m_routingTable.LookupRoute(dst, toDst)); toDst.GetFlag() que terá de retornar um boolean VALID; e é invocado o método toDst.GetSeqNo() que terá de ser maior ou igual ao retorno do método rreqHeader.GetDstSeqno(). Para além destas verificações, analisa-se o campo flag da mensagem RREQ, que terá de ter o bit de destination Only não activo (obtido com o método rreqHeader.GetDestinationOnly()). Após serem confirmadas estas condições, é invocado o método SendReplyByIntermediateNode que recebe como argumento a

mensagem RREQ, a entrada da tabela de encaminhamento para o destino, a entrada da tabela de encaminhamento para a fonte e um boolean obtido com o método rreqHeader.GetGratiousRrep(), que retorna *true* ao verificar que o bit de gratuitous RREP da flag contida na mensagem RREQ está activa. O método SendReplyByIntermediateNode será explicado de seguida, assim como a flag mencionada de gratuitous RREP.

No caso de não se verificaram as condições descritas, quer para o nó descartar a mensagem de RREQ, assim como de cumprirem as condições necessárias para que o nó possa processar a resposta ao pedido de rota, o nó irá retransmitir a mensagem de RREQ em broadcast. Desta forma, a mensagem pode ser difundida na rede e que possivelmente outro nó possa responder ao pedido de rota do nó fonte.

- **Método SendReplyByIntermediateNode**

Este método deverá ser invocado quando um nó intermédio quer processar a resposta ao pedido de rota da fonte. É invocado no método RecvRequest descrito anteriormente, e recebe como argumentos a entrada da tabela de encaminhamento para o destino, a entrada da tabela de encaminhamento para a fonte e um boolean que indica se na flag contida na mensagem RREQ o bit de gratuitous RREP está activo.

Com a informação contida na entrada para a fonte, é processado neste método a resposta ao pedido de rota. Esta resposta corresponde a uma mensagem RREP, transmitida em unicast para a fonte. Cada interface tem um socket correspondente, sendo invocado o método socket->SendTo (packet, 0, InetSocketAddress (toOrigin.GetNextHop (), AODV_PORT)) para efectuar a transmissão da mensagem RREP em unicast para a fonte. Com a indicação que a flag de gratuitous RREP está activa e com a entrada para o destino (recebidos como argumento no método), transmite-se em unicast para o destino uma mensagem gratuitous RREP de forma a informar o destino da rota encontrada para a fonte.

- **Método SendReply**

Este método deverá ser invocado quando o nó destino quer processar a resposta ao pedido de rota da fonte. É invocado no método RecvRequest descrito anteriormente, e recebe como argumentos a mensagem RREQ recebida no destino e a entrada para a fonte.

O nó de destino deve incrementar o seu próprio número de sequência se o número de sequência no pacote RREQ é igual ao valor incrementado. Caso contrário, o destino não muda seu número de sequência antes de gerar a mensagem RREP.

É invocado o método socket->SendTo (packet, 0, InetSocketAddress (toOrigin.GetNextHop (), AODV_PORT)) para efectuar a transmissão da mensagem RREP em unicast para a fonte.

- **Método RecvReply**

Este método deverá ser invocado na recepção de uma mensagem RREP. Será invocado no método RecvAmr da classe RoutingProtocol, quando se verifica que a mensagem recebida no nó é do tipo AMRTYPE_RREP. Sendo invocado com três argumentos, a mensagem recebida, o endereço IP do nó actual (receiver) e o endereço IP do nó que enviou a mensagem (sender).

Inicialmente neste método (RecvReply) verifica-se se a mensagem recebida de RREP corresponde a uma mensagem de Hello (com um TTL = 1), no caso de corresponder a esse tipo de mensagem é invocado o método ProcessHello. Isto verifica-se ao analisar os campos de m_origin e m_destination da mensagem, que para o caso das mensagens Hello, ambos contêm o endereço IP do vizinho. Ao contrário das mensagens RREP utilizadas para o envio da resposta para o nó fonte ao pedido de rota recebido, que contêm nestes campos a informação do endereço IP da fonte (m_origin) e o endereço IP do destino (m_destination).

Na recepção de uma mensagem RREP o nó cria (ou actualiza) a entrada na sua tabela de encaminhamento para o destino e retransmite a mensagem RREQ em unicast para a fonte, utiliza-se da informação da entrada anteriormente criada para a fonte (explicado no método RecvRequest) para encaminhar a resposta ao pedido de rota. Deste forma, o nó actual (nó intermédio pertencente ao caminho de nós disjuntos encontrado) ao criar (ou actualizar) uma rota para o destino antes de retransmitir a mensagem RREP, quando o nó fonte receber a mensagem RREP e iniciar a transmissão dos dados para o destino por este caminho encontrado, este nó poderá encaminhar os pacotes segundo a rota criada (ou actualizada) para o destino.

Neste método (RecvReply) quando se verifica que o nó não contém nenhuma entrada (ou uma entrada inválida) para o destino na sua tabela de encaminhamento, é criada (ou actualizada) a entrada.

Para criar (ou actualizar) a entrada utilizam-se os seguintes métodos para obter a informação da rota encontrada para a fonte:

- Destination: será o endereço IP do destino, contido na mensagem RREP no campo m_origin. Para obter este endereço na mensagem, utiliza-se o método rrepHeader.GetDst();
- Sequence number: corresponde ao sequence number do destino, será obtido através do método rrepHeader.GetDstSeqno();
- NextHop: será o endereço IP do nó que enviou a mensagem. Corresponde à variável sender, recebida como argumento no método RecvReply.
- lastHop: corresponde ao último salto para o nó destino, que permite identificar o caminho encontrado entre o nó actual e a destino. Utiliza-se o método rrepHeader.GetFirstHop() para obter o endereço IP do nó vizinho do destino.
- HopCount: conterá o número de saltos do caminho entre o nó actual e o destino. Utiliza-se o método rrepHeader.GetHopCount() para obter o numero de saltos acumulado até ao momento, e incrementa-se o ultimo salto realizado para o nó actual.

Na recepção da mensagem RREP quando se verifica que o nó já contém uma entrada válida na sua tabela de encaminhamento para o destino, apenas será actualizada no caso de se verificarem as seguintes situações:

- Se a mensagem RREP contém um sequence number do destino mais recente que o armazenado na entrada da tabela de encaminhamento. Neste caso removem-se todas as encontradas anteriormente para o destino, e actualiza-se a entrada da rota principal, segundo a descrição acima referida para a actualização da entrada para o destino.
- Se a mensagem RREP contém um sequence number do destino igual ao da entrada da tabela de encaminhamento, tenta-se inserir a nova rota na tabela. Esta nova entrada apenas será inserida se corresponder a um caminho de nós disjuntos e se o número de rotas armazenadas na tabela para o destino não atingiu o limite (armazena-se no máximo até três rotas). Utiliza-se o método NewDisjointPath para verificar se o caminho novo é de nós disjuntos, este método recebe como argumento o endereço IP do nó que enviou a mensagem e o endereço do nó de ultimo salto para o destino desse caminho (toDst.NewDisjointPath(sender, rrepHeader.GetFirstHop()).

No caso de não serem verificadas nenhuma das situações apresentadas para criar (ou actualizar) a

entrada para o destino, a mensagem RREP é descartada, visto que se observava uma das seguintes situações no caminho encontrado para o destino: continha informação de uma rota obsoleta (poderia criar ciclos no encaminhamento se a mensagem fosse retransmitida e utilizada pela fonte para a transmissão dos dados); a informação de um caminho que não seria de nós disjuntos (ou seja, utilizava um nó intermédio já pertencente a outro caminho anteriormente encontrado e ainda válido); ou o número de entradas para o destino já tinha atingido o limite de caminhos encontrados, logo este novo caminho encontrado não será utilizado.

Depois das verificações efectuadas neste método (RecvReply) apresentadas até ao momento e a entrada criada (ou actualizada) para o destino, utiliza-se o método IsMyOwnAddress(rrepHeader.GetOrigin()) para verificar que o nó actual que recebeu a mensagem RREP corresponde ao nó fonte. No caso de ser o nó fonte a receber a resposta, é efectuada uma consulta à tabela de encaminhamento para verificar se a flag da entrada para o destino se encontra "IN_SEARCH", nesse caso remove-se o temporizador correspondente SendRequest criado no inicio da procura de rota, através do método m_addressReqTimer[dst].Remove(). Este temporizador foi descrito anteriormente no método SendRequest, que permite controlar a espera da resposta ao pedido de rota da fonte. Após remover este temporizador, dá-se inicio à transmissão dos dados que estavam na fila de espera até ao momento para o destino, através do método SendPacketFromQueue(dst, toDst.GetRoute()) que utiliza a rota encontrada.

Para o caso de não ser o nó fonte a receber a mensagem RREP, mas sim um nó intermédio pertencente ao caminho de nós disjuntos encontrado, este terá de retransmitir a mensagem para a fonte, para isso utiliza-se o método LookupRoute que recebe o endereço IP da fonte de forma a obter a entrada correspondente. Depois é transmitida a mensagem para a fonte, utilizando o socket correspondente da interface do nó, ao invocar o método SendTo para processar o envio (socket->SendTo(packet, 0,InetSocketAddress(toOrigin.GetNextHop(), QMAODV_PORT))).

4.2.3. Manutenção das rotas

Entre os nós vizinhos da rede são enviadas periodicamente mensagens Hello, de forma a detectar quebras de ligações e possibilitar às fontes a comutação para uma rota alternativa, ou no caso de não conter nenhuma iniciar o processo de descoberta de rota. Deste modo, com a notificação de quebra enviada, as fontes podem dar seguimento à transmissão dos dados.

As mensagens Hello correspondem a mensagens RREP com um TTL = 1, sendo que na mensagem os campos de endereço IP fonte e o endereço IP do destino, coloca-se o endereço do próprio nó. Para a troca destas mensagens, utiliza-se o temporizador m_htimer responsável em controlar os tempos de envio. Utiliza-se deste timer o método HelloTimerExpire. Sempre que se verifica que o nó que contém uma rota activa (com a utilização do método HasActiveRoute), invoca-se periodicamente o método SendHello para processar o envio da mensagem.

No método RecvReply, descrito anteriormente, quando se verifica que a mensagem recebida corresponde a uma mensagem Hello, é invocado o método ProcessHello. Com a mensagem recebida neste método, o nó actualiza a rota para o seu vizinho e permite verificar assim que este se encontra disponível e acessível.

Quando não é recebida uma mensagem Hello dos vizinhos durante um período de tempo, indica que ocorreu uma quebra na ligação entre os vizinhos, e será necessário enviar mensagem Route Error (RERR) pelos caminhos de nós disjuntos, ao qual este nó que ficou indisponível pertencia. Para esta situação utiliza-se o método SendRerrWhenBreaksLinkToNextHop que invoca o método SendRerrMessage com dois argumentos, um consiste na mensagem RERR criada e outro um vector (precursors) com os endereços IP dos nós vizinhos pertencentes a caminhos de nós disjuntos encontrados anteriormente e ainda activos. É também utilizado o método SendRerrWhenNoRouteToForward, quando se pretende encaminhar um pacote (no método Forwarding) e verifica-se que o nó não contém uma rota para o destino, logo é necessário notificar os restantes nós que utilizam este nó como próximo salto para o destino.

Quando é verificada a recepção de uma menagem RERR num nó, é invocado o método RecvError. Este método permite invalidar a rota para o destino que utilizava como próximo salto o nó que enviou a mensagem, nó esse que pertence ao caminho em que ocorreu a quebra de ligação detectada. E para o caso de conter nós precursores (nós no sentido da fonte) de caminhos activos, processa-se a retransmissão da mensagem RERR para que esta possa percorrer todo o caminho até chegar ao nó fonte. Desta forma o nó fonte na recepção desta notificação, se verificar que o caminho em que ocorreu a quebra da ligação corresponde à rota principal, poderá assim comutar para uma rota alternativa. No de ser uma notificação de quebra numa das ligações de um caminho alternativo, a fonte apenas remove (coloca-a como inactiva) a rota correspondente. Utiliza-se o método InvalidateRoutesWithDst

para colocar a entrada da tabela de encaminhamento da rota correspondente ao caminho de nós disjuntos que ficou inexequível.

4.3. Protocolo QMRS

O primeiro passo para a implementação do protocolo Ad Hoc QoS On-Demand Multipath Routing with Route Stability (QMRS) proposto, foi o de acrescentar ao simulador ns-3 um novo módulo, que teve como base o protocolo Ad Hoc Multipath Routing (AMR) descrito anteriormente.

Em seguida, o próximo passo para a implementação do protocolo QMRS, consistiu em efectuar as alterações necessárias, de forma a modifica-lo para obter um protocolo de encaminhamento de múltiplos caminhos de nós disjuntos com QoS, para que este possa atender aos requisitos de QoS de determinada aplicação, sem comprometer a estabilidade do processo de encaminhamento.

Na descrição que se segue da implementação do protocolo QMRS, tendo este como base o protocolo AMR, apenas serão descritas as alterações das estruturas utilizadas, assim como a implementação acrescida nos métodos principais, relativamente aos apresentados na Secção 4.2.2 e na Secção 4.2.3 para o protocolo AMR. Neste sentido, para uma boa compreensão da implementação do protocolo QMRS apresentado de seguida, recomenda-se primeiramente uma leitura da descrição da implementação do protocolo AMR.

4.3.1. Estruturas utilizadas

O protocolo QMRS utiliza a informação contida nas tabelas de encaminhamento de cada nó para o processo de encaminhamento, assim como o protocolo ARM descrito anteriormente. Este protocolo além de possibilitar armazenar a informação de múltiplas rotas para o mesmo destino, permite também armazenar a informação em cada entrada do atraso fim a fim verificado em cada um dos caminhos de nós disjuntos encontrados. Assim como armazenar o valor mínimo da potência do sinal recebido, verificado entre as ligações dos nós pertencentes a cada um dos caminhos encontrados. A Tabela 4.4 apresenta as alterações efectuadas à tabela de encaminhamento do protocolo ARM (Tabela 4.3).

Tabela 4.4: Tabela de encaminhamento do protocolo QMRS

destination	sequence number	advertised hop count	next hop	last hop	hop count	rxSS Path	Delay	lifetime	list of alternative routes					
									next hop	last hop	hop count	rxSS Path	Delay	lifetime
									next hop	last hop	hop count	rxSS Path	Delay	lifetime

A Tabela 4.4 apresenta os campos pertencentes a uma entrada na tabela de encaminhamento do protocolo QMRS, que durante o processo de descoberta de rota, permite armazenar a informação da rota principal e das rotas alternativas. Permite armazenar para cada rota, o respectivo atraso fim a fim verificado no caminho, armazenado no campo Delay. Assim como armazenar no campo rxSSPath, o mínimo valor lido da potência do sinal recebido nas várias ligações que constituem o caminho.

4.3.2. Manutenção da informação entre vizinhos

Entre os nós da rede localizados dentro do alcance de transmissão, ocorrem trocas periódicas de mensagens "Hello" que indicam que os vizinhos estão disponíveis e acessíveis, informação essa que é necessária para o processo de descoberta e manutenção das rotas. A implementação para realizar estas verificações e os procedimentos utilizados, já foram descritos na Secção 4.2.3. para o protocolo ARM.

Para além da acessibilidade, através das mensagens "Hello" é mantida a informação para cada vizinho da potência do sinal recebido, métrica utilizada para apurar depois do processo de descoberta de rota, entre os possíveis caminhos encontrados qual é o mais estável para ser utilizado na transmissão dos dados da fonte para o destino. Este valor é lido durante a troca de mensagens Hello entre vizinhos, nos vários métodos descritos na Secção 4.2.3. Faz-se essa leitura da potência do sinal recebido directamente na camada física através de uma interacção cross-layer. Para este processo é invocado o método GetRxPowerDbmPhy(receiver) da classe RoutingProtocol que retorna o valor lido da camada física, obtido ao invocar no protocolo o método GetRxPowerDbm contido no ficheiro yans-wifi-phy da camada física. Este ficheiro está colocado no módulo wifi e foi alterado para obter o valor

lido da potência do sinal recebido. Foi necessário realizar algumas alterações para armazenar o valor pretendido e possibilitar a interacção directa com o protocolo de encaminhamento para a leitura deste valor. Para tal foram adicionadas à classe YansWifiPhy a variável m_rxSSDbm e os métodos GetRxSSDbm e SetRxSSDbm, para armazenar e possibilitar obter esse valor lido da potência do sinal recebido.

Para além desta métrica, actualiza-se durante a troca de mensagens de controlo entre vizinhos, o atraso ocorrido no nó para a transmissão da mensagem (delayNode). Este atraso irá conter o atraso de processamento, o atraso ocorrido na fila de espera, o tempo de contenção no acesso ao meio e o tempo de transmissão.

Em todos os métodos que processam a transmissão das mensagens de controlo para a rede, actualiza-se este delayNode. Será actualizado através da implementação de uma tag no pacote transmitido, que irá percorrer as várias camadas. O simulador ns-3 possibilita através da utilização de PacketTags, a partilha de informação entre as camadas até 20 bytes. Para tal foram criadas umas tags designadas de QueueTags colocadas no módulo wifi e incluído no protocolo, que durante a troca de mensagens de controlo adiciona-se à tag criada o tempo de simulação (tag.Set(Simulator::Now().GetMicroSeconds()) e acrescenta-se ao pacote essa tag (packet->AddPacketTag(tag)). Quando o pacote chega à camada física e pronto para a transmissão, ou seja, no método SendPacket da classe YansWifiPhy, determina-se o atraso ocorrido no nó até ao momento e soma-se por fim o atraso de transmissão (txDuration.GetMicroSeconds()). Esse atraso é colocado num vector para o cálculo da estimativa do próximo atraso verificado no nó para a transmissão de um pacote. Este vector irá conter sempre os últimos cinco atrasos verificados, e a média desse valor consistirá no atraso do nó (delayNode).

Este atraso (delayNode) será utilizado durante o processo de descoberta de rota para acumular o atraso fim a fim em cada um dos caminhos de nós disjuntos encontrados, assim como para a verificação realizada nó a nó, do atraso fim a fim máximo requisitado por determinada aplicação durante a procura de rota, explicado com maior detalhe como é usado na descrição realizada ao protocolo QMRS no Capitulo 3.

4.3.3. Descoberta de rota

Durante a descoberta de rota ocorrem trocas de pacotes de controlo entre os nós da rede, de forma a

possibilitar encontrar múltiplos caminhos de nós disjuntos entre o nó fonte e o destino que cumpram o atraso fim a fim máximo requisitado por determinada aplicação.

A seguir descreve-se a implementação realizada nos métodos mais importantes, utilizados para o processo de descoberta de rota, sendo invocados outros métodos que realizam tarefas de menor dimensão durante a execução destes métodos principais, que permitem ao protocolo QMRS realizar os objectivos descritos.

Tendo sido descrito para o protocolo AMR (aplicando-se a mesma descrição para o protocolo QMRS) o processo de descoberta de rota, para possibilitar encontrar múltiplos caminhos de nós disjuntos (Secção 4.2). Para o protocolo QMRS apenas serão descritos para os métodos principais, como são realizadas e quando, as operações necessárias para o cálculo do atraso, da potência do sinal recebido e para a verificação do cumprimento de QoS, durante a descoberta e manutenção das rotas e na selecção do caminho principal para a transmissão dos dados.

- **Método SendRequest**

Como foi referido anteriormente no protocolo AMR, este método deverá ser invocado quando um nó pretende transmitir para determinado destino, e verifica que não contém na sua tabela de encaminhamento, uma rota válida para esse destino.

Na implementação do protocolo QMRS foi necessário proceder a alterações de forma a obter as métricas de atraso fim a fim e da potência do sinal recebido mínima de cada caminho encontrado durante o processo de descoberta de rota. Assim como de verificar nó a nó durante a procura de rota se o caminho cumpre o requisito de atraso fim a fim máximo. Para efectuar o cálculo destas métricas foi necessário alterar a mensagem RREQ utilizada no protocolo AMR. Podem-se verificar as alterações efectuadas na mensagem RREQ comparando a versão do protocolo ARM (Tabela 3.1) com a do protocolo QMRS (Tabela 3.4).

Sendo este o método responsável por transmitir em broadcast o pedido de rota para a rede, é necessário enviar na mensagem RREQ o requisito de atraso fim a fim máximo que a aplicação solícita para a transmissão do fluxo de dados. Deste modo é necessário incluir no campo delayReq da mensagem RREQ, o requisito de atraso fim a fim máximo. Para isso utiliza-se o método SetDelayReq que recebe como argumento o requisito de atraso fim a fim máximo.

Para determinar o atraso fim a fim nos caminhos encontrados entre a fonte e o destino, sendo esta uma métrica aditiva, que resulta do somatório de todos os atrasos obtidos entre os nós que constituem o caminho. É necessário inserir na mensagem RREQ o atraso verificado na fonte para a transmissão para o vizinho (delayNode), tendo sido o cálculo deste atraso explicado na Secção 4.3.2. Para obter o delayNode é invocado o método GetAverageDelayPhy que recebe como argumento o endereço IP do nó, sendo depois incluído na mensagem RREQ através do método SetDelayAcc.

Antes da transmissão da mensagem neste método (SendRequest), inclui-se na PacketTag implementada (QueueTag) o tempo actual, para determinar o atraso que irá ocorrer no envio desta mensagem para o vizinho e possibilitar actualizar a estimativa realizada do delayNode.

- **Método RecvRequest**

Durante a retransmissão da mensagem de RREQ nos nós intermédios, de forma a obter o atraso fim a fim do caminho de nós disjuntos encontrado, é necessário acumular o atraso verificado entre os nós que vão constituir esse caminho. Para acumular este atraso, inicialmente é invocado o método GetAverageDelayPhy recebendo como argumento o endereço IP do nó, para obter a estimativa do atraso na transmissão de uma mensagem (delayNode). A este valor soma-se o atraso acumulado até ao momento contido na mensagem RREQ e é colocado na mensagem antes da retransmissão. Utiliza-se o método SetDelayAcc para colocar a soma deste atraso verificado no caminho percorrido até ao momento para o destino (rreqHeader.SetDelayAcc(rreqHeader.GetDelayAcc() + delayNode).

Para além do atraso para o destino, é necessário determinar durante a retransmissão da mensagem RREQ nos nós intermédios, do atraso verificado no caminho para a fonte. Este atraso para a fonte é necessário nas entradas criadas para a fonte, que fica com o atraso correspondente. Para este cálculo, verifica-se se o nó intermédio é vizinho da fonte, caso seja é invocado o método SetDelayToOrigin para inserir no campo DelayToOrigin da mensagem RREQ o delayNode. No caso de não ser um nó vizinho da fonte, é invocado o método SetDelayToOrigin para inserir no campo DelayToOrigin da mensagem o atraso verificado no caminho até ao momento para a fonte mais o atraso verificado no nó (rreqHeader.SetDelayToOrigin(rreqHeader.GetDelayToOrigin() + delayNode)).

Para verificar se o cumprimento de QoS está a ser cumprido durante a descoberta de rota, é necessário comparar o valor determinado do atraso no caminho percorrido até ao momento com o

atraso fim a fim máximo requisitado pela aplicação. Esta comparação é realizada observando os campos contidos na mensagem RREQ do DelayAcc com o campo DelayReq, utilizando-se os métodos GetDelayAcc e GetDelayReq para obter estes valores. Sendo que o DelayAcc terá de ser inferior ao DelayReq para continuar o processo de descoberta de rota, com a retransmissão da mensagem RREQ. Para o caso de não se verificar o cumprimento de QoS, a mensagem é descatada, visto que o caminho até ao momento já não cumpre o requisito, possibilitando encontrar outros caminhos durante a descoberta de rota que possam cumprir o requisito.

Durante a retransmissão da mensagem de RREQ nos nós intermédios, é necessário também determinar a potência de sinal. Sendo esta uma métrica côncava, que resulta do cálculo do mínimo valor da potência de sinal recebido nas várias ligações que constituem o caminho. Neste sentido o nó vizinho da fonte que recebe a mensagem, terá de colocar o valor verificado da potência do sinal recebido na mensagem antes de a retransmitir. Para obter o valor da potência do sinal recebido (readPhyRxPowerDbm) é invocado o método GetRxPowerDbmPhy que recebe como argumento o endereço IP do nó. Compara-se este valor lido readPhyRxPowerDbm com o valor da potência do sinal recebido mínima verificado no caminho percorrido até ao momento, contido no campo RxSSPath da mensagem RREQ. Caso o valor do RxSSPath seja superior ao readPhyRxPowerDbm, troca-se o campo RxSSPath da mensagem para o valor lido, para que no caminho percorrido até ao momento fique com a informação da potência do sinal recebido mínimo.

Neste método quando o nó que recebe a mensagem contém a informação de uma rota válida para o destino, antes de retransmitir a resposta (mensagem RREP) ao pedido de rota, necessita de verificar o atraso fim a fim e a potência do sinal recebido mínima do caminho para o destino.

Para o caso de ser o próprio destino, é invocado o método SendReply. No caso de ser um nó intermédio com a informação na tabela de encaminhamento de uma rota válida para o destino é invocado o método SendReplyByIntermediateNode.

- **Método SendReply**

Este método deverá ser invocado quando o nó destino quer processar a resposta ao pedido de rota da fonte.

É necessário enviar na resposta o respectivo atraso fim a fim do caminho encontrado entre a fonte e o destino. É invocado o método SetDelayPath para incluir no campo delayPath da mensagem

RREP esse atraso, que recebe como argumento o valor acumulado do atraso (delayPath) contido na mensagem RREQ recebida no destino.

Para além do atraso fim a fim do caminho entre a fonte e o destino, é necessário enviar na mensagem RREP a potência do sinal recebido mínima verificada nesse caminho, para isso é invocado o método SetRxPower que recebe como argumento o valor contido no campo RxSSPath na mensagem RREQ.

A mensagem RREP contém também os campos RxSSPathToDst e delayToDst, para que nos nós intermédios pertencentes ao caminho encontrado durante a retransmissão da resposta em unicast para a fonte, permitirem verificar e actualizar estas métricas do caminho entre os nós intermédios e o destino. Deste modo, os nós ao receberem a resposta e ao criarem (actualizarem) as entradas da tabela de encaminhamento da rota para o destino, incluem na entrada o respectivo atraso fim a fim e a potência do sinal recebido mínima desse caminho. Sendo este método (SendReply) invocado apenas pelo nó destino que irá proceder ao envio da resposta, estes dois campos estarão definidos com os valores por defeito, que o nó vizinho do destino é que vai inserir os primeiros valores observados das métricas. No método RecvReply explicar-se-á esta parte em que o nó vizinho da fonte recebe a mensagem RREP.

- **Método SendReplyByIntermediateNode**

Este método deverá ser invocado quando um nó intermédio quer processar a resposta ao pedido de rota da fonte.

É necessário enviar na resposta, o atraso fim a fim e a potência do sinal recebido mínimo do caminho entre a fonte e o destino. O nó intermédio que vai transmitir a mensagem RREP terá de incluir na mensagem no campo delayPath o atraso entre a fonte e o destino. Na mensagem RREQ no campo delayAcc, contém o atraso verificado até ao momento em que este nó recebe a mensagem, logo para determinar o atraso fim a fim entre a fonte e o destino é necessário acrescentar o conhecimento do atraso que este nó tem para o destino, contido na sua tabela de encaminhamento. Utiliza-se o método GetDelay para obter o atraso da entrada para o destino (toDst) da tabela de encaminhamento. Esse atraso será acrescentado ao delayAcc contido na mensagem RREQ e inserido na mensagem RREP no campo delayPath.

Esse atraso determinado entre a fonte e o destino (delayPath), é utilizado pelo mecanismo de

verificação de cumprimento de QoS, que compara o delayPath determinado com o delayReq contido na mensagem RREQ. Para o caso de o atraso fim a fim máximo requisitado (delayReq) não puder ser cumprido no caminho encontrado, o nó não irá transmitir a mensagem RREP para a fonte.

No caso de cumprir o requisito de atraso fim a fim máximo, o nó intermédio antes de responder ao pedido de rota necessita de determinar a potência do sinal recebido mínimo (rxSSPath) do caminho encontrado para incluir esse valor na mensagem RREP. É invocado o método GetRxPower para obter a potência do sinal recebido mínima do caminho entre o nó intermédio e o destino (rxSSToDst), contido na entrada para o destino (toDst). O valor mínimo entre esse valor (rxSSToDst) e o atraso contido no campo rxSSPath na mensagem RREQ, corresponderá à potência do sinal recebido mínima do caminho encontrado entre a fonte e o destino, e será inserido no campo rxSSPath na mensagem RREP.

Será também inserido nos campos rxSSPathToDst e delayToDst da mensagem RREP a informação que o nó tem na entrada para o destino do atraso (toDst.GetDelay()) e da potência do sinal recebido (toDst.GetRxPower()).

- **Método RecvReply**

Este método deverá ser invocado na recepção de uma mensagem RREP. Neste método é necessário actualizar o atraso acumulado e a potência do sinal mínima do caminho entre o nó actual e o destino.

É invocado o método GetRxPowerDbmPhy que recebe como argumento o endereço IP, para obter a potência do sinal recebido verificado na recepção da mensagem (readRxSS). Será inserido no campo RxSSPathToDst na mensagem RREP, assim como na entrada da tabela de encaminhamento para o destino, o valor mínimo verificado entre o readRxSS e o rxSSPathToDst contido na mensagem até ao momento. Para inserir esse valor na mensagem RREP é invocado o método SetRxPower.

Para actualizar o atraso acumulado no caminho entre o nó actual e o destino, é invocado o método GetDelayToDst para obter do campo delayToDst da mensagem RREP o atraso verificado para o destino até ao momento, soma-se a esse atraso o atraso do nó (delayNode) e é inserido no campo delayToDst da mensagem RREP antes de esta ser retransmitida para o destino.

Ao actualizar as duas métricas (delayToDst e rxSSPathToDst) permite também aos nós intermédios ao criarem (actualizarem) as entradas para o destino, incluir na entrada o respectivo atraso e a potência de sinal recebido mínima desse caminho.

4.3.4. Manutenção das rotas

Como já referido no protocolo AMR, entre os nós vizinhos da rede são enviadas periodicamente mensagens Hello, de forma a detectar quebras de ligações e possibilitar às fontes a comutação para uma rota alternativa, ou no caso de não conter nenhuma, iniciar o processo de descoberta de rota. Deste modo, com a notificação de quebra enviada, as fontes podem dar seguimento à transmissão dos dados. Sendo o processo de detecção e envio da notificação de quebra já explicado na Secção 4.3.3.

No protocolo QMRS foi necessário acrescentar funcionalidades neste mecanismo de forma a actualizar o atraso fim a fim e a potência do sinal recebido mínima dos caminhos de nós disjuntos encontrados durante o processo de descoberta de rota. Assim como verificar se o requisito de QoS continua a ser cumprido nesses caminhos. Foram implementados novos tipos de mensagens utilizadas neste mecanismo. A mensagem InspectPath (Tabela 3.7) e a mensagem ReplyInspectPath (Tabela 3.8).

É invocado periodicamente o método SendInspectPath, para efectuar a transmissão da mensagem InspectPath nos caminhos de nós disjuntos encontrados. É invocado o método RecvInspectPath quando os nós recebem mensagens do tipo InspectPath, para acumular nó a nó o atraso ocorrido e verificar o valor mínimo da potência do sinal recebido nos caminhos de nós disjuntos encontrados. E permitir verificar em cada nó pertencente a esses caminhos, durante a retransmissão da mensagem, se o requisito do atraso fim a fim máximo continua a ser respeitado.

É invocado o método SendReplyInspectPath para enviar a resposta, apenas quando o atraso fim a fim máximo requisitado continua a ser cumprido no caminho, tendo sido as métricas actualizadas para a fonte. Durante a transmissão da resposta para a fonte, é invocado o método RecvReplyInspectPath sempre que a mensagem do tipo ReplyInspectPath é recebida nos nós intermédios pertencentes ao caminho para a fonte. Neste método actualiza-se o atraso e a potência do sinal recebido mínima, que permite actualizar as entradas da tabela de encaminhamento para o destino.

Capítulo 5

Análise e Discussão dos Resultados

Neste capítulo, são apresentados os resultados das simulações realizadas às soluções apresentadas e implementadas no âmbito do objecto de estudo desta dissertação, assim como a análise e discussão dos resultados obtidos.

Os resultados das simulações efectuadas, permitem analisar o desempenho do protocolo Ad hoc On-Demand Multipath Routing (AMR) e do protocolo Ad Hoc QoS On-Demand Multipath Routing with Route Stability (QMRS). Foi utilizado o simulador NS-3 [9] para implementação e realização dos testes aos protocolos propostos.

Na realização dos testes e análise do desempenho dos protocolo de encaminhamento propostos, utilizou-se o protocolo AODV incluído no simulador, para servir de referência para uma comparação dos resultados obtidos com os mesmos parâmetros de simulação, de forma a verificar o desempenho dos protocolos em relação ao atraso fim a fim, taxa de transferência efectiva e a taxa de pacotes entregues no destino.

De forma a obter resultados fiáveis, realizaram-se 60 simulações para cada um dos protocolos, sendo apresentados os resultados nos gráficos seguintes (Figura 5.1,Figura 5.2 e Figura 5.3) em que se indica para cada diferente velocidade dos nós, a média e o intervalo de confiança de 95% correspondente. A simulação efectuada consistiu em dispor 80 nós de forma aleatória e com uma movimentação aleatória, utilizando o modelo de mobilidade random waypoint. Os nós deslocam-se a velocidades máximas entre 0 e 10m/s, num local com uma dimensão de 600m x 1500m. Todos os nós da rede têm um alcance de transmissão de 160m. A Tabela 5.1 indica os parâmetros utilizados nas simulações realizadas.

Durante o tempo de simulação de 200s, são transmitidos no total 15 fluxos de dados, em que são seleccionados aleatoriamente o nó fonte e o destino. É usado o protocolo UDP para transportar tráfego Constant Bit Rate (CBR), os pacotes de dados foram definidos com um tamanho de 64bytes e são transmitidos a uma taxa de transferência de 2kbps.

Tabela 5.1: Parâmetros de Simulação

Parâmetros	Valor
Dimensão	600m x 1500m
Número de nós	80
Modelo de Mobilidade	Random Waypoint
Posição dos nós	Aleatório
Alcance transmissão	160m
Tipo de tráfego	Constant Bit Rate (CBR)
Tamanho dos pacotes	64 bytes
Número de fluxos	15
Taxa de transmissão	2kbps
Especificações Wifi	IEEE 802.11b, freq. 2.4Ghz Taxa de transf. até 2Mbps
Tempo de simulação (s)	200

5.1. Análise dos resultados

Para verificar o desempenho dos protocolos, são utilizadas como meio de comparação as métricas de atraso fim a fim, taxa de entrega de pacotes no destino e a taxa de transferência efectiva.

- **Atraso fim a fim**

Relativamente ao atraso médio verificado durante a transmissão dos fluxos de dados entre a fonte e o destino, pode-se constatar ao analisar a Figura 5.1, para as velocidades máximas entre 0 e 10m/s apresentadas, o protocolo QMRS consegue dar garantias do requisito de atraso fim a fim para os vários fluxos transmitidos, pode-se observar um atraso fim a fim médio entre 0 e 150 milissegundos. Enquanto para o protocolo AODV, verifica-se um atraso médio muito superior durante a transmissão dos pacotes até ao nó destino. Como anteriormente descrito, o protocolo AODV não oferece quaisquer garantias de qualidade de serviço, em que o caminho usado na transmissão poderá estar congestionado, com o acréscimo de ter de iniciar o processo de descoberta de rota sempre que ocorre uma falha numa ligação entre nós pertencentes à rota usada para transmitir. Verifica-se desta forma nos resultados obtidos apresentados na Figura 5.1 um atraso significativo nos pacotes até serem entregues no destino. Enquanto com as garantias que o protocolo QMRS oferece, os caminhos encontrados e utilizados na transmissão dos dados, são caminhos pouco congestionados que cumprem o atraso requisitado, com um número menor de quebra de rotas, uma recuperação de rota

mais rápida e os mecanismos de manutenção de rotas e verificação de incumprimento de QoS utilizados, permitem obter um atraso fim a fim significativamente inferior ao protocolo AODV.

Em relação ao protocolo AMR, que serviu de base para a implantação do protocolo QMRS, este ao permitir encontrar múltiplos caminhos de nós disjuntos entre a fonte e o destino, segundo os resultados obtidos, verifica-se uma melhoria significativa comparativamente ao protocolo AODV. O atraso fim a fim nos resultados obtidos permitem verificar que, para o protocolo AMR o atraso fim a fim médio foi de cerca de 2.5s, enquanto para o AODV foi cerca de 6s.

No entanto o protocolo AMR não tem qualquer garantia de QoS, nem selecciona o caminho principal para a transmissão dos dados nas fontes entre os caminhos de nós disjuntos encontrados, aquele que detiver uma maior estabilidade. Neste sentido, verificam-se nos resultados obtidos que este protocolo é um pouco instável, comparativamente ao protocolo QMRS. Ao analisar os resultados apresentados na Figura 5.1, verifica-se que a média do atraso fim a fim é muito superior no protocolo AMR comparativamente ao verificado no protocolo QMRS. Isto ocorre porque entre as várias simulações realizadas, existem várias eventualidades que podem suceder durante a transmissão dos dados entre as fontes e os destinos para que o atraso médio fim a fim neste protocolo seja superior. Entre as quais pode-se referir que, mesmo que para o caminho principal seleccionado para a transmissão dos dados nas fontes, não se verificar um atraso fim a fim elevado no momento em que este é descoberto, durante a transmissão do fluxo de dados nesse caminho, este poderá ficar congestionado. Assim como seleccionar um caminho que está na iminência de ocorrer uma quebra de ligação, ou seja, um caminho pouco estável, e que pouco tempo depois de se ter dado o início da transmissão de dados na fonte, ocorrer imediatamente uma quebra de ligação (por exemplo, pelo simples facto de que apenas um dos nós pertencentes ao caminho se ter movimentado para fora do alcance de transmissão do nó vizinho que o utilizava como próximo salto para reencaminhar os pacotes para o destino), tendo o nó fonte de comutar para uma rota alternativa, caso esta exista. Pode-se também verificar um aumento no atraso fim a fim, porque ao comutar para uma rota alternativa, o caminho seleccionado poderá estar completamente sobrecarregado, visto que não há qualquer restrição imposto de atraso fim a fim máximo nos caminhos encontrados, originando assim um atraso fim a fim muito superior para as diferentes velocidades máximas, comparativamente ao protocolo QMRS.

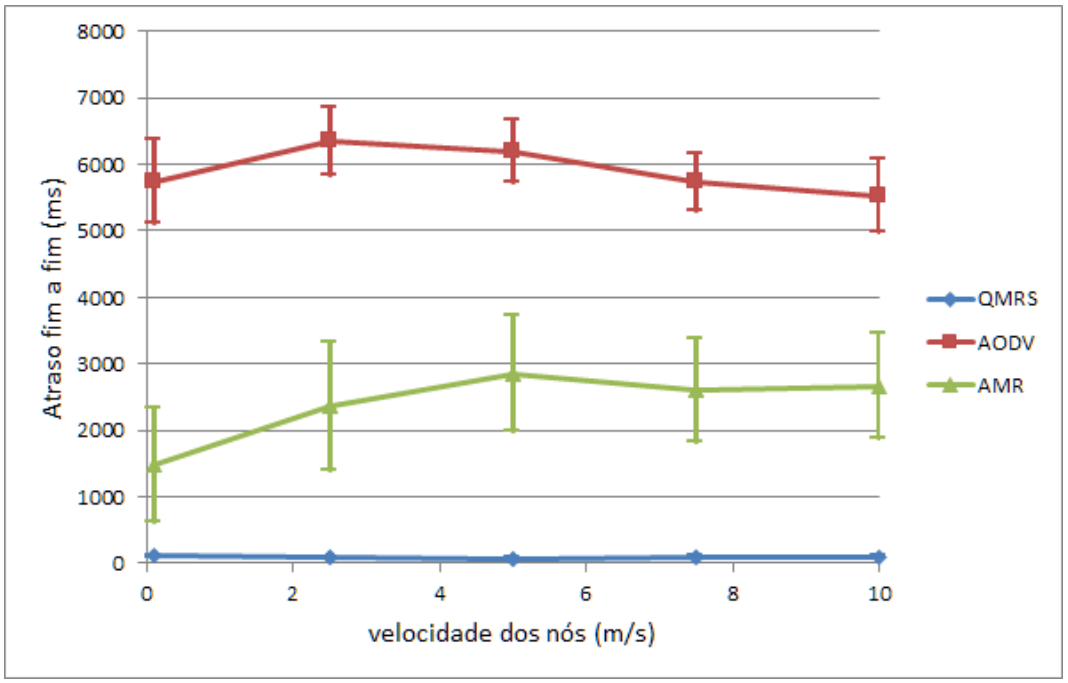


Figura 5.1: Atraso fim a fim

- **Taxa de Entrega de Pacotes e Taxa de Transferência**

A Figura 5.2 mostra que durante as simulações realizadas, para as diferentes velocidades máximas entre 0 e 10 m/s, que a taxa de entrega de pacotes no destino no protocolo QMRS mantém-se entre os 70% e 80%, verificando-se para o protocolo AODV taxas de entrega de pacotes muito inferiores, entre os 10% e 20%. E para o protocolo ARM taxas de entrega de pacotes entre os 45% e 60%.

Relativamente à taxa de transferência efectiva, pode-se observar na Figura 5.3 quanto aos resultados obtidos nas simulações realizadas, que a média da quantidade de dados transferidos entre a fonte e o destino, para o protocolo QMRS para as velocidades máximas entre 0 e 10m/s apresentadas, mantém-se nos 2kbps, enquanto para o protocolo AODV verificam-se valores inferiores a 1kbps. Para o protocolo ARM verificam-se valores intermédios, entre os 1,5kbps e os 2kbps.

O protocolo QMRS, que tem por objectivo garantir a estabilidade do processo de encaminhamento, na selecção que faz da rota para transmissão com ligações estáveis entre os nós, e com o mecanismo de descoberta de rota utilizado, que verifica o cumprimento do requisito de atraso fim a fim. Os nós que constituírem o caminho durante a transmissão, serão nós da rede com filas de espera pouco congestionadas. O inverso é verificado no protocolo AODV, como apenas possibilita a descoberta de uma rota, não sendo realizada qualquer verificação durante a sua descoberta ou

manutenção, o caminho utilizado poderá conter ligações de tal forma congestionadas, que os nós não conseguem ter acesso ao meio e encaminhar os pacotes todos, ocorrendo congestão nas filas de espera, que ao atingir a capacidade máxima, grande parte dos pacotes acabam por ser descartados, originando os resultados observados na Figura 5.2 para as diferentes velocidades máximas entre 0 e 10 m/s, de taxas de entrega no destino entre 10% e 20%, e na Figura 5.3 de taxas de transferência efectiva inferiores a 1kbps.

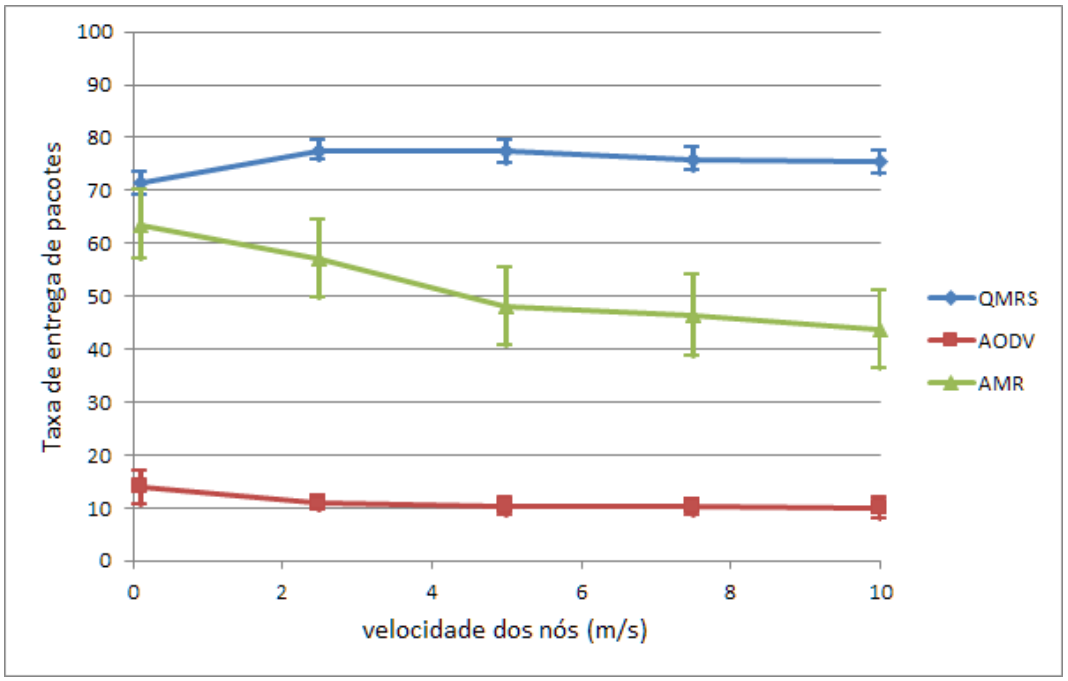


Figura 5.2: Taxa de entrega de pacotes no destino

No protocolo AMR verificam-se valores intermédios entre os outros dois protocolos apresentados. Visto que este tem a vantagem de descobrir múltiplos caminhos entre a fonte e o destino, logo permite efectuar a comutação para uma rota alternativa, caso ocorra uma quebra na rota principal, o que permite encaminhar os pacotes de dados mal a fonte seja notificada da quebra de rota. Enquanto o protocolo AODV terá de iniciar o processo de descoberta de rota, o que origina a grande diferença para a perda de pacotes entre os dois protocolos.

No entanto, o protocolo AMR comparativamente ao protocolo QMRS, ao não oferecer nenhum tipo de garantias de QoS e não seleccionar para transmissão de dados o caminho mais estável, pode-se apurar segundos os resultados obtidos, que ocorrem com maior frequência quebras nas ligações entre os nós pertencentes ao caminho principal, assim como a descoberta de caminhos

congestionados. Deste modo, conforme as desvantagens descritas e segundo os resultados apresentados na Figura 5.2, explica-se a diferença de cerca de 15% na taxa de entrega de pacotes no destino entre os dois protocolos.

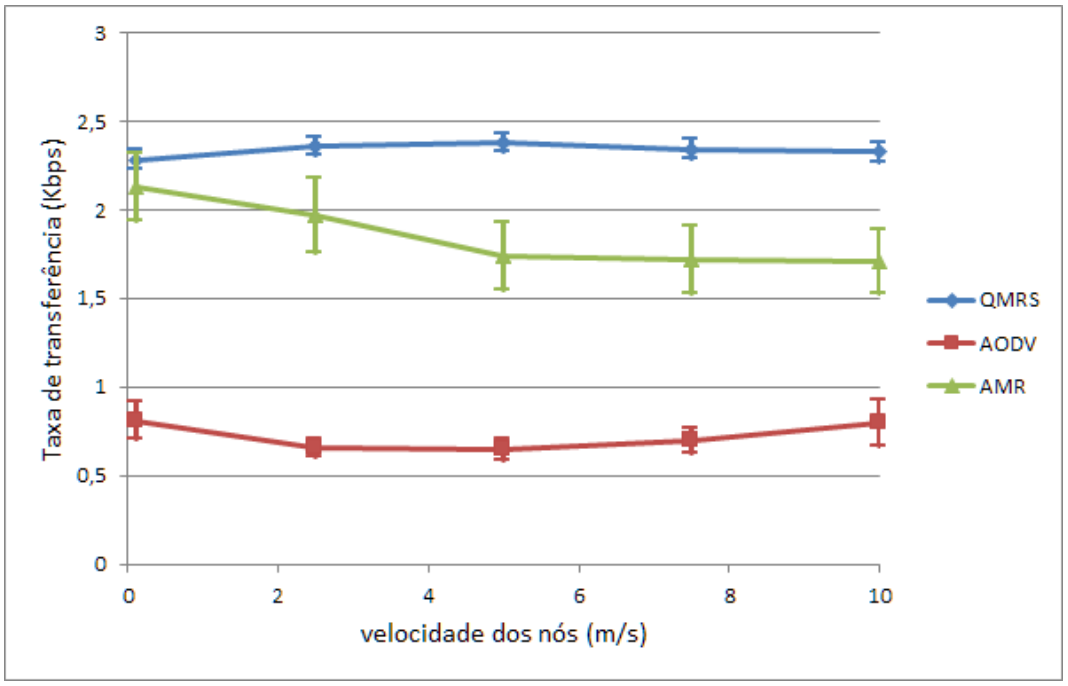


Figura 5.3: Taxa de transferência efectiva

5.2. Exemplo da tabela de encaminhamento na fonte

Nesta secção apresenta-se uma topologia de rede criada no simulador ns-3, utiliza-se para visualização gráfica a ferramenta pyviz, que permite exibir para a simulação realizada a transmissão de dados da fonte para o destino. Apresentam-se também as entradas criadas na tabela de encaminhamento da fonte para o destino, após o processo de descoberta de rota, na simulação realizada para o protocolo QMRS.

A Figura 5.4 ilustra uma topologia de rede apenas com oito nós, onde é possível observar a transmissão de dados entre a fonte e o destino a uma taxa de 536kbps, após o processo de descoberta de rota desencadeado na fonte (representada com o endereço IP 10.0.0.1/24) para o destino (representado com o endereço IP 10.0.0.4/24).

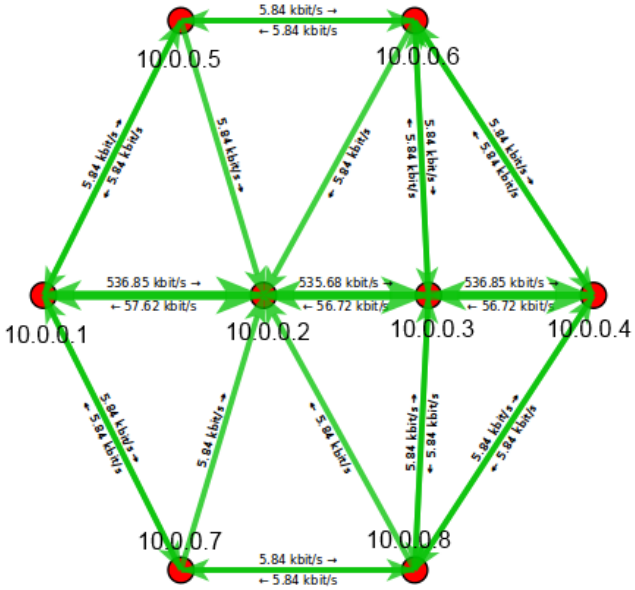


Figura 5.4: Transmissão dos dados após processo de descoberta de rota na fonte

A Figura 5.5 mostra a tabela de encaminhamento da fonte com as respectivas entradas para o destino, obtidas durante a simulação, após o processo de descoberta de rota desencadeado na fonte (endereço 10.0.0.1). É possível observar as três entradas na tabela para o destino (endereço 10.0.0.4). Para cada entrada é possível verificar os diferentes campos de próximo salto (nextHop) e ultimo salto (lastHop) para o destino, que correspondem aos três caminhos de nós disjuntos encontrados entre a fonte e o destino. Tendo sido sempre seleccionada como rota principal e utilizada para a transmissão dos dados (como se pode observar na Figura 5.4) a entrada em que o nextHop corresponde ao nó da rede com o endereço IP 10.0.0.2 e o lastHop o nó com o endereço IP 10.0.0.4, visto ser este o caminho de nós disjuntos mais estável entre todos os encontrados.

```
------- Tabela de Encaminhamento ------ node:10.0.0.1
QMRS Routing table
Destination     SeqNumber       AdvertidedHop   Gateway(nextHop)        HopCount        RxSSPath        Delay   Expire
10.0.0.2            0           255             10.0.0.2                  1               -85           172     2.99918
interface -> m_local=10.0.0.1; m_mask=255.0.0.0; m_broadcast=10.255.255.255; m_scope=2; m_secondary=0   UP

10.0.0.4            0           255             10.0.0.2                  3               -85           1502    2.82718
interface -> m_local=10.0.0.1; m_mask=255.0.0.0; m_broadcast=10.255.255.255; m_scope=2; m_secondary=0   UP

       NumPaths   Nexthop      LastHop     HopCount    RxSSPath    Delay     LifeTime
       1  (3)     10.0.0.2     10.0.0.3          3         -85     1502      2.82718
       NumPaths   Nexthop      LastHop     HopCount    RxSSPath    Delay     LifeTime
       2  (3)     10.0.0.5     10.0.0.6          3         -90     1514      2.99937
       NumPaths   Nexthop      LastHop     HopCount    RxSSPath    Delay     LifeTime
       3  (3)     10.0.0.7     10.0.0.8          3         -90     1556        2.834
10.0.0.5            0           255             10.0.0.5                  1               -90           344     2.99937
interface -> m_local=10.0.0.1; m_mask=255.0.0.0; m_broadcast=10.255.255.255; m_scope=2; m_secondary=0   UP

10.0.0.7            0           255             10.0.0.7                  1               -90           344     3
interface -> m_local=10.0.0.1; m_mask=255.0.0.0; m_broadcast=10.255.255.255; m_scope=2; m_secondary=0   UP
```

Figura 5.5: Tabela de encaminhamento da fonte após processo de descoberta de rota

Capítulo 6

Conclusão

Neste último capítulo são apresentadas algumas considerações finais sobre o trabalho efectuado, além das principais contribuições desta dissertação. Por último são apontadas algumas direcções para trabalho futuro.

6.1. Considerações Finais

O tema de estudo desta dissertação é o encaminhamento em redes móveis ad hoc. Este tipo de redes tem sido alvo de um crescente interesse pela comunidade científica, devido ao crescimento nas vendas e utilização de dispositivos móveis, particularmente os smartphones e tablets. As redes ad hoc possibilitam aos utilizadores destes dispositivos, estabelecerem facilmente e rapidamente uma rede entre eles, sem assistência de um ponto de acesso, de forma a potenciarem a utilização de diversos serviços, que envolvem a partilha de informação disponível nos vários dispositivos que estabelecem a rede.

A grande diversidade, o aumento da capacidade dos dispositivos móveis sem fios e simultaneamente a evolução das aplicações multimédia, criou a necessidade de propor e avaliar formas de, neste tipo de redes, se oferecerem garantias de Qualidade de Serviço (QoS). Estas garantias deverão ser disponibilizadas ao tráfego fim a fim, de forma a cumprirem-se os requisitos de QoS solicitados pelas aplicações multimédia e que envolvem habitualmente, largura de banda, atraso fim a fim, variação do atraso, etc. No entanto, este tipo de redes tem características que tornam a introdução de QoS um desafio ainda maior do que fazê-lo nas redes fixas. A topologia da rede é muito dinâmica devido à mobilidade dos nós, à entrada e saída de nós na rede e às quebras de ligações constantes, que provocam alterações inesperadas das rotas. Além disso, o meio de comunicações sem fios é partilhado entre os nós vizinhos.

O objectivo inicial deste trabalho era dotar as redes ad hoc de garantias de qualidade de serviço no tráfego fim a fim, para proporcionar aos utilizadores uma boa experiencia de utilização da rede com

aplicações exigentes, nomeadamente aplicações sensíveis ao atraso. Para alcançar este objectivo foram tomadas várias decisões nas fases distintas necessárias para atingir a solução proposta final.

Começou-se por os protocolos de encaminhamento existentes para as redes móveis ad hoc e as várias abordagens possíveis, com o objectivo de fazer um levantamento das desvantagens e inconvenientes de cada estratégia utilizada no processo de encaminhamento.

Analisaram-se também os protocolos existentes no simulador ns-3 de forma a tomar a decisão inicial de qual a melhor estratégia a utilizar na solução a propor. Verificou-se que no simulador ns-3 não existia qualquer protocolo de encaminhamento de múltiplos caminhos. Apenas existiam protocolos de encaminhamento *unipath*, que durante o processo de descoberta de rotas encontram apenas uma rota para o destino.

Assim sendo, a primeira solução concebida e desenvolvida no âmbito desta dissertação, teve como objectivo construir um protocolo de encaminhamento para redes ad hoc, capaz de encontrar múltiplos caminhos de nós disjuntos entre pares origem destino. A ideia subjacente a este tipo de solução tem como objectivo tornar o protocolo mais resistente às mudanças topológicas constantes pois quando uma rota falha poderá ser imediatamente substituída por uma rota alternativa, sem ser necessário recorrer ao processo de descoberta de rotas. Desta forma é possível reduzir o tráfego de controlo na rede e diminuir, simultaneamente, o atraso fim a fim durante a transmissão de dados. A solução de múltiplos caminhos de nós disjuntos proposta foi implementada no simulador ns-3, e designou-se Ad hoc On-Demand Multipath Routing (AMR).

Como ponto de partida para a implementação deste protocolo foram usadas as implementações do protocolo Ad Hoc On-Demand Distance Vector [13] que acompanha o ns-3 e do protocolo Ad Hoc On-Demand Multipath Distance Vector (AOMDV) [18], existente no ns-2. A implementação do AMR foi devidamente testada e avaliada tendo sido os resultados obtidos comparados com os do AODV. Estes resultados permitiram concluir que a solução proposta, ao possibilitar encontrar múltiplos caminhos de nós disjuntos entre uma fonte e um destino reduz o número necessário de pacotes de controlo na rede e diminui o atraso durante a transmissão do fluxo de dados entre as fontes e os destinos, assim como regista uma melhoria significativa na taxa de entrega de pacotes no destino.

A segunda proposta apresentada e também implementada no simulador ns-3, consiste num protocolo de encaminhamento com QoS em redes ad hoc, designado de Ad hoc QoS On-Demand

Multipath Routing with Route Stability (QMRS). Inicialmente analisaram-se as métricas de QoS, assim como algumas estratégias para melhorar o desempenho dos protocolos de encaminhamento em redes ad hoc e permitir dar garantias de QoS, nomeadamente no atraso fim a fim. O protocolo QMRS teve como base o protocolo AMR, tendo sido efectuadas as alterações necessárias para que este pudesse dar garantias de QoS sem comprometer a estabilidade do processo de encaminhamento.

O protocolo QMRS proposto, com os mecanismos existentes de descoberta, manutenção, recuperação de rotas e verificação de incumprimento do requisito de QoS nos caminhos encontrados, possibilita num único processo de descoberta de rota, encontrar múltiplos caminhos de nós disjuntos que cumpram o requisito de atraso fim a fim máximo, entre a fonte e o destino. O nó fonte irá utilizar para transmissão o caminho mais estável entre todos os encontrados, com o objectivo de que o caminho principal seleccionado se mantenha exequível por um maior período de tempo possível, sem haver a necessidade de comutar para outra rota alternativa, ou mesmo reiniciar o processo de descoberta de rota no caso de não existir uma rota alternativa quando é detectada uma falha na rota principal. Apenas no caso de serem encontradas rotas com uma estabilidade idêntica é seleccionada a rota com menor atraso fim a fim. A descoberta de múltiplos caminhos de nós disjuntos alternativos permite que, quando se verifica uma falha na rota principal devido à movimentação de algum nó pertencente a esta, o nó fonte reaja à notificação dessa quebra, procedendo à comutação para uma rota alternativa.

A referida proposta foi também devidamente testada e avaliada, tendo-se verificado segundo os resultados obtidos, que o protocolo QMRS comparativamente com os protocolos AODV e AMR, apresentou um menor atraso fim a fim na transmissão de dados e um aumento significativo na taxa de entrega de pacotes e na taxa de transferência efectiva.

6.2. Trabalho Futuro

Nas redes móveis ad hoc, existem várias possibilidades diferentes para abordar o encaminhamento com QoS. As soluções propostas e implementadas no contexto deste trabalho mostraram resultados positivos, apresentando melhorias significativas comparativamente aos protocolos existentes no simulador ns-3.

No entanto, ficou a vontade de realizar mais testes e acrescentar mecanismos complementares

ao protocolo de encaminhamento com QoS, de forma a reforça-lo para que este fique mais robusto e permita dar suporte de QoS em larga escala, suportando uma percentagem significativa dos nós da rede a transmitir. Devido ao tempo necessário para a implementação e testes efectuados aos protocolos propostos, ficaram alguns testes interessantes por realizar.

A solução apresentada pretendeu ter sempre em conta o ponto de vista do utilizador, ou seja, é conveniente que a utilização do dispositivo móvel seja realizada de forma normal, sendo no entanto possível que o dispositivo seja utilizado como encaminhador de tráfego para determinado destino, mesmo em circunstâncias em que não pretenda utilizar a rede. Neste sentido, todas as soluções apresentadas promovem uma baixa utilização dos nós da rede sempre que não seja necessário a sua utilização, para permitir um baixo processamento e um baixo consumo da energia dos dispositivos. Dai as várias opções tomadas, desde ter-se optado inicialmente por um protocolo reactivo, até à não exigência de marcação e classificação de pacotes, etc., durante o processo de encaminhamento. Em vez disso foram propostos mecanismos de manutenção, recuperação rápida de rotas e verificação de incumprimento do requisito de QoS nos caminhos encontrados.

O próximo passo poderia passar por equacionar métricas alternativas, além do atraso fim a fim ou da potência de sinal recebido, Outra possível abordagem seria acrescentar mecanismos complementares para expandir e transformar a solução proposta num modelo de QoS, aplicando algo semelhante ao modelo DiffServ com as devidas adaptações. Nesse aspecto em particular é de realçar que nas redes ad hoc não existem routers de fronteira ou seja, todos os nós podem-se movimentar por decisão própria em qualquer direcção e a diferentes velocidades, logo, a marcação dos pacotes teria de ser da responsabilidade de todos os nós da rede, com a marcação com o DSCP apropriado de acordo com os requisitos da aplicação. Na utilização do protocolo QMRS, após o mecanismo de descoberta de rota, ao obter os múltiplos caminhos de nós disjuntos poder-se-ia agrupar esses caminhos de forma a disponibilizar caminhos por classes de serviço. Adicionalmente o modelo poderia, à semelhança do que acontece no modelo DiffServ dar um tratamento diferenciado nó a nó aos fluxos de tráfego, de acordo com a classe de serviço a que cada fluxo pertence.

Outro possível desenvolvimento seria proceder a melhorias na camada MAC, de forma a aumentar a probabilidade de acesso ao meio para os fluxos que necessitam de maior prioridade para cumprir determinado requisito. Utilizar algo como o *standard* 802.11e EDCA para permitir prioridades

diferentes no acesso ao meio entre as categorias existentes, e definir para cada categoria filas de espera dedicadas com os respectivos parâmetros de Contention Window (CW) e Arbitration Interframe Space (AIFS).

Bibliografia

[1] *Xu, Ya, John Heidemann, and Deborah Estrin. "Geography-informed energy conservation for ad hoc routing." Proceedings of the 7th annual international conference on Mobile computing and networking. ACM,* 2001.

[2] *Shah, Rahul C., and Jan M. Rabaey. "Energy aware routing for low energy ad hoc sensor networks." Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE. Vol. 1. IEEE,* 2002.

[3] *Singh, Suresh, Mike Woo, and Cauligi S. Raghavendra. "Power-aware routing in mobile ad hoc networks." Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking. ACM,* 1998.

[4] *N. Sarma, S. Nandi, "A Route Stability based Multipath QoS Routing (SMQR) in MANETs," First International Conference on Emerging Trends in Engineering and Technology,* 2008.

[5] *Sarma, Nityananda, and Sukumar Nandi. "Route stability based QoS routing in mobile Ad Hoc networks." Wireless Personal Communications 54.1,* 2010.

[6] *Shah, Samarth H., and Klara Nahrstedt. "Predictive location-based QoS routing in mobile ad hoc networks." Communications, 2002. ICC 2002. IEEE International Conference on. Vol. 2. IEEE,* 2002.

[7] *SJ, Jinyang Li John Jannotti Douglas, R. Karger De Couto David, and Robert Morris. "A scalable location service for geographic ad hoc routing,* 2000.

[8] *Sivakumar, Raghupathy, Prasun Sinha, and Vaduvur Bharghavan. "CEDAR: a core-extraction distributed ad hoc routing algorithm." Selected Areas in Communications, IEEE Journal on 17.8,* 1999.

[9] *Wu, Shih-Lin, et al. "A multi-channel MAC protocol with power control for multi-hop mobile ad hoc networks." The Computer Journal 45.1,* 2002.

[10] *Li, Xuefei, and Laurie Cuthbert. "Distributed Cross-Layer QoS Provisioning in Mobile Ad Hoc Networks." ITS Telecommunications Proceedings, 2006 6th International Conference on. IEEE,* 2006.

[11] *Goldsmith, Andrea J., and Stephen B. Wicker. "Design challenges for energy-constrained ad hoc wireless networks." Wireless Communications, IEEE 9.4,* 2002.

[12] X. Wang, Mobile Ad Hoc Networks: Protocol Design, InTech, January 30, 2011.

[13] *Perkins, Charles, E. Belding-Royer, and Samir Das. "Ad hoc on demand distance vector (AODV) routing (RFC 3561)." IETF MANET Working Group (August. 2003),* 2003.

[14] *Johnson, David B., David A. Maltz, and Josh Broch. "DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks." Ad hoc networking 5,* 2001.

[15] *Perkins, Charles E., and Pravin Bhagwat. "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers." ACM SIGCOMM Computer Communication Review. Vol. 24. No. 4. ACM,* 1994.

[16] *Clausen, Thomas, et al. "Optimized link state routing protocol (OLSR).",* 2003.

[17] *Haas, Zygmunt J., Marc R. Pearlman, and Prince Samar. "The zone routing protocol (ZRP) for ad hoc networks.",* 2002.

[18] *Marina, Mahesh K., and Samir R. Das. "Ad hoc on-demand multipath distance vector routing." ACM SIGMOBILE Mobile Computing and Communications Review 6.3,* 2002.

[19] *Wang, Zheng, and Jon Crowcroft. "Quality-of-service routing for supporting multimedia applications." Selected Areas in Communications, IEEE Journal on 14.7,* 1996.

[20] *R. Braden, D. Clark, and S. Shenker, "Integrated Services in the internet architecture: an overview", RFC 1633, USC/Information Sciences Institute, MIT, Xerox PARC,* 1994.

[21] *S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December,* 1998.

[22] *Wu, Kui, and Janelle Harms. "QoS support in mobile ad hoc networks." Crossing Boundaries-the GSA Journal of University of Alberta,* 2001.

[23] *Hwang, Youngki, and Pramod Varshney. "An adaptive QoS routing protocol with dispersity for ad-hoc networks." System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on. IEEE,* 2003.

[24] Li, Xuefei and Laurie Cuthbert., "Multipath QoS routing of supporting DiffServ in mobile ad hoc networks," *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on. IEEE,* 2005.

[25] *Li, Xuefei, and Laurie Cuthbert. "On-demand node-disjoint multipath routing in wireless ad hoc networks." Local Computer Networks, 2004. 29th Annual IEEE International Conference on. IEEE,* 2004.

[26] *Nasipuri, Asis, Robert Castañeda, and Samir R. Das. "Performance of multipath routing for on-demand protocols in mobile ad hoc networks." Mobile Networks and applications 6.4,* 2001.

[27] *Xiao, Li, Jun Wang, and M. Nahrstedt. "The enhanced ticket-based routing algorithm." Communications, 2002. ICC 2002. IEEE International Conference on. Vol. 4. IEEE,* 2002.

[28] *De, Swades, et al. "Trigger-based distributed QoS routing in mobile ad hoc networks." ACM SIGMOBILE Mobile Computing and Communications Review 6.3,* 2002.

[29] *Ge, Ying, Thomas Kunz, and Louise Lamont. "Quality of service routing in ad-hoc networks using OLSR." System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on. IEEE,* 2003.

[30] *Qi Xue, Aura Ganz, Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks, Journal of Parallel and Distributed Computing, Volume 63, Issue 2, February,* 2003.

[31] *Liu, Shun, and Jian Liu. "Delay-aware multipath source routing protocol to providing QoS support for wireless ad hoc networks." Communication Technology (ICCT), 2010 12th IEEE International Conference on. IEEE,* 2010.

[32] *Sarma, N., and S. Nandi. "Route Stability based QoS Routing (RSQR) in MANETs." Proc. of the 10th International Symposium on Wireless Personal Multimedia Communications (WPMC),* 2007.

[33] *NS-3, http://www.nsnam.org/.*

[34] *Simulator, Network. "ns-2." http://www.isi.edu/nsnam/ns/,* 1989.

[35] *Varga, András. "The OMNeT++ discrete event simulation system." Proceedings of the European Simulation Multiconference (ESM'2001). Vol. 9. sn,* 2001.

[36] *Sarkar, Nurul I., and Syafnidar A. Halim. "A review of simulation of telecommunication networks: simulators, classification, comparison, methodologies,* and recommendations." Cyber Journals: Multidisciplinary Journals in Science and Technology-Journal of Selected Areas in Telecommunications (JSAT) 2.03, 2011.

[37] *Weingartner, Elias, Hendrik Vom Lehn, and Klaus Wehrle. "A performance comparison of recent network simulators." Communications, 2009. ICC'09. IEEE International Conference on. IEEE,* 2009.

[38] *Ahn, Gahng-Seop, et al. "SWAN: Service differentiation in stateless wireless ad hoc networks." INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Vol. 2. IEEE,* 2002.

[39] *Lee, Seoung-Bum, et al. "INSIGNIA: An IP-based quality of service framework for mobile ad hoc networks." Journal of Parallel and distributed Computing 60.4,* 2000.

[40] *Liu, Shun, and Jian Liu. "Delay-aware multipath source routing protocol to providing QoS support for wireless ad hoc networks." Communication Technology (ICCT), 2010 12th IEEE International Conference on. IEEE,* 2010.

[41] *Chen, Lei, and Wendi B. Heinzelman. "A survey of routing protocols that support QoS in mobile ad hoc networks." Network, IEEE 21.6,* 2007.