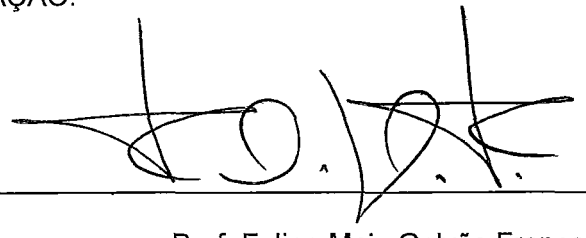
UM SISTEMA DE MEDIÇÃO REMOTA DE CONSUMO DE ENERGIA ELÉTRICA BASEADO NO PROTOCOLO ZIGBEE

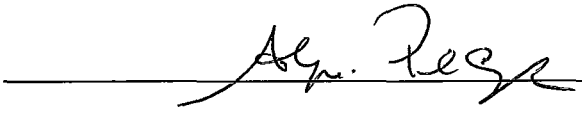
Cristiane Amaral de Magalhães

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

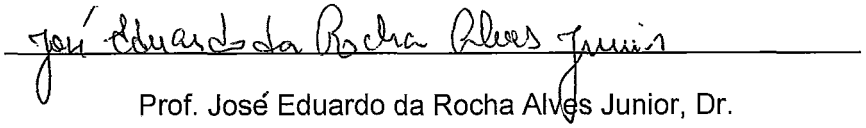


Prof. Felipe Maia Galvão França, Ph.D.



Prof. Aloysio de Castro Pinto Pedroza, Dr.

Prof. Cláudio Luis de Amorim, Ph.D.



Prof. José Eduardo da Rocha Alves Junior, Dr.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2008

MAGALHÃES, CRISTIANE AMARAL

Um sistema de Medição Remota de Consumo de Energia Elétrica baseado no Protocolo Zigbee [Rio de Janeiro] 2008

XIII, 114 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2008)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Sistema de Medição para Redução de Perdas
2. Protocolo IEEE 802.15.4/ZigBee
3. Protocolos de roteamento para redes Ad-Hoc

I.COPPE/UFRJ II.Título (série)

Ao meu pai....

Agradecimentos

Primeiramente a Deus por permitir que este objetivo pudesse ser alcançado.

Ao meu pai Roberto Perret pela ajuda na revisão deste modelo de dissertação, sugestões, críticas e apoio. Sem a sua persistência eu não teria chegado aqui. Ofereço-lhe esta vitória.

A minha mãe Leila Amaral pelo apoio e palavras de carinhos nos momentos mais difíceis.

A amiga Bárbara Espenchidt pela força, pela paciência em me ouvir nos momentos de maior dificuldade e pela revisão do abstract.

A todos os professores do PESC pelas lições apreendidas.

Aos meus orientadores Prof. Felipe Maia Galvão França e Prof. Vítor Manuel de Morais Santos Costa pela orientação, apoio, críticas, sugestões e amizade.

Aos professores Prof. Aloysio de Castro Pinto Pedroza, Prof. José Eduardo da Rocha Alves Junior e Prof. Cláudio Luis de Amorim pela presença na banca examinadora.

A todos os amigos da minha equipe do CEPEL em especial ao José Eduardo da Rocha Alves Junior, Fabio Cavaliere e César Bandim pelo constante incentivo sempre indicando a direção a ser tomada nos momentos de maior dificuldade. Agradeço, pela confiança depositada no meu trabalho de dissertação.

A todos os colegas do PESC em especial: Ivomar Soares, André Oliveira, Gustavo Dias, Patrícia Sampaio, Vivian Lengruber, André Pinho e Alexandre Alves pelas múltiplas ajudas, pela amizade e companheirismo.

A todos os meus amigos que direta ou indiretamente me ajudaram com palavras incentivadoras.

Resumo da Dissertação apresentada a COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM SISTEMA DE MEDIÇÃO REMOTA DE CONSUMO DE ENERGIA ELÉTRICA BASEADO NO PROTOCOLO ZIGBEE

Cristiane Amaral de Magalhães

Março/ 2008

Orientadores: Felipe Maia Galvão França
Vítor Manuel de Morais Santos Costa

Programa: Engenharia de Sistemas e Computação

Esta dissertação apresenta o estudo dos protocolos de roteamento *on-demand*: *Ad Hoc On-Demand Distance Vector Routing* (AODV) e *Dynamic Source Routing* (DSR) utilizando o protocolo IEEE 802.15.4/*Zigbee* voltado para uma aplicação específica de medição: Sistema de Medição para Redução de Perdas (SMRP). Este sistema está sendo implantado pelo Centro de Pesquisas de Energia Elétrica (CEPEL) através de um projeto piloto na cidade de Porto Velho – RO, junto a concessionária Centrais Elétricas de Rondônia S.A. (CERON). Atualmente a comunicação do SMRP realiza-se através de *Power Line Communication* (PLC). Como a principal desvantagem do PLC são os ruídos e as interferências, o padrão 802.15.4/*ZigBee* que utiliza redes sem fio foi escolhido como alternativa ao PLC. Com o objetivo de induzir algumas falhas, foram simuladas algumas quebras de enlaces através da retirada de alguns dispositivos e conseqüentemente a topologia foi alterada. Desta forma, foram simuladas sete estratégias para analisar o comportamento e o desempenho dos protocolos de roteamento utilizando dois parâmetros: pacotes recebidos com sucesso e pacotes perdidos. De acordo com os resultados obtidos, o protocolo que obteve o melhor desempenho em todas as estratégias foi o AODV. Este protocolo é o mais indicado para este tipo de topologia por ele ter características semelhantes ao protocolo *ZigBee* e ao funcionamento do SMRP.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A SYSTEM OF REMOTE METERING OF ELECTRICAL ENERGY CONSUMPTION OF THE PROTOCOL BASED ZIGBEE

Cristiane Amaral de Magalhães

March/2008

Advisor: Felipe Maia Galvão França

Vítor Manuel de Morais Santos Costa

Department: Systems Engineering and Computer Science

This dissertation presents the study of routing protocols on demand Ad Hoc On-Demand Routing Distance Vector (AODV) and Dynamic Source Routing (DSR) using the protocol IEEE 802.15.4/Zigbee dedicated to a specific application of measurement: Measurement System for Reduction of Losses (MSRL). This system is being deployed by the research center Centro de Pesquisa de Energia Elétrica (CEPEL), through a pilot project in the city of Porto Velho - RO, along with the company Centrais Elétricas de Rondônia S.A. (CERON). Presently the communication of the MSRL is held by Power Line Communication (PLC). As the mains disadvantages of the PLC are the noises and interferences, the standard 802.15.4/ZigBee that uses wireless networks was chosen as an alternative to the PLC. With the purpose to induce some failures were simulated losses of links through the removal of some devices e consequently the topology has been changed. Thus, were simulated seven strategies to analyze the behavior and performance of the routing protocols using two parameters: successfully received packages and lost packages. According to the results obtained, the protocol which obtained the best performance in all the strategies was the AODV. This protocol is the most suitable for this type of topology because it has similar characteristics to the protocol ZigBee and similar working method to the SMRL.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AES	Advanced Encryption Standard
AF	Application Framework
AODV	Ad hoc On-demand Distance Vector
AP	Access Point
APS	Application Support Sublayer
APSDE	Application support sub-layer data entity
APSME	Application support sub-layer management entity
BSN	Beacon sequence number
BTR	Registros de transações broadcast
BTT	Broadcast transaction table
BPSK	Binary phase shift keying
CBR	Constant Bit Rate
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DIFS	Distribution Coordination Function Interframe Space
DSR	Dynamic Source Routing
DSSS	Direct Sequence Spread Spectrum
ESM	Estações de suporte à mobilidade
FFD	Full Function Device
FHSS	Frequency Hopping Spread Spectrum
IBSS	Infrastructure Basic Service Set
IEEE	Institute of Electrical and Eletronics Engineers
ISM	Industrial Scientific Medical
LR-WPAN	Low Rate Wireless Personal Area Network
MAC	Medium Access Control
MAC	Módulo de Alimentação e Comunicação
MFR	Medium access control footer
MHR	Medium access control Header
MLDE-SAP	Médium Access Control Layer Data Entity - Service Access Point
MLME-SAP	Médium Access Control Layer management entity - Service access point
MM	Módulo de Medição
MPDU	MAC protocol data unit
MPDU	Médium Access Control PHY Data Unit
MSDU	Medium access control sub-layer service data unit
NS	Network Simulator
OSI	Open System Interconnection
O-QPSK	Offset quadrature phaseshift keying
PAN	Personal area network
PD-SAP	Physical layer data - service access point
PHR	PHY Header
PLC	Power line Communication
PLME-SAP	Physical layer management entity - service access point
PPDU	PHY protocol data unit
PSDU	PHY service data unit
RFD	Reduce Function Device
RREP	Route reply
RREQ	Route request
RERR	Route Error

SAP	Service access point
SFD	Sequence and start-of frame Delimiter
SPF	Shortest Path First
SiP	System in Package
SRMP	Sistema de Medição para Redução de Perdas
SSP	Security Service Provider
Trace	Arquivo Registro Dados
UM	Unidade de Medição
UMC	Unidade de Medição e Comunicação
UWB	Ultra-wide band
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WPAN	Wireless Personal Area Network
WWAN	Wireless World Area Network
ZC	ZigBee Coordinator
ZDO	Zigbee Device Object
ZDP	ZigBee Device Profile
ZED	ZigBee End - Device
ZR	ZigBee Router

Capítulo 1

Introdução

1.1 Contexto Geral

Em várias regiões do Brasil, os números associados às perdas comerciais são elevados e todas as concessionárias de distribuição de energia elétrica vêm desenvolvendo ações de combate de diferentes maneiras e com diferentes resultados. Entende-se por perdas comerciais as perdas das distribuidoras devido a fraudes, irregularidades, perdas decorrentes de má administração etc. As irregularidades encontradas normalmente são realizadas dentro das casas dos consumidores, uma vez que os medidores são instalados nas residências ou no limite entre a residência e a rua.

Uma maneira de minimizar o problema é o deslocamento da medição para fora da residência e também a reunião de vários medidores em um mesmo equipamento, denominado Unidade de Medição. Esta unidade é instalada em um poste, na rua, onde qualquer irregularidade poderá ser verificada com facilidade, devido à maior visibilidade. A Figura 1 apresenta um exemplo da situação real. As Unidades de Medição se comunicam com uma Unidade de Medição e Comunicação [1]. A finalidade é em primeiro lugar concentrar todas as medições de uma rua em um mesmo lugar o que facilita a leitura pela concessionária. Esta comunicação pode ser realizada por diversos meios físicos: o próprio cabo de energia *Power Line Comunication* (PLC), comunicação por ondas eletromagnéticas (usando protocolo de baixa velocidade como *ZigBee* ou GSM celular) por um fio dedicado de sinal de fibra ótica etc.

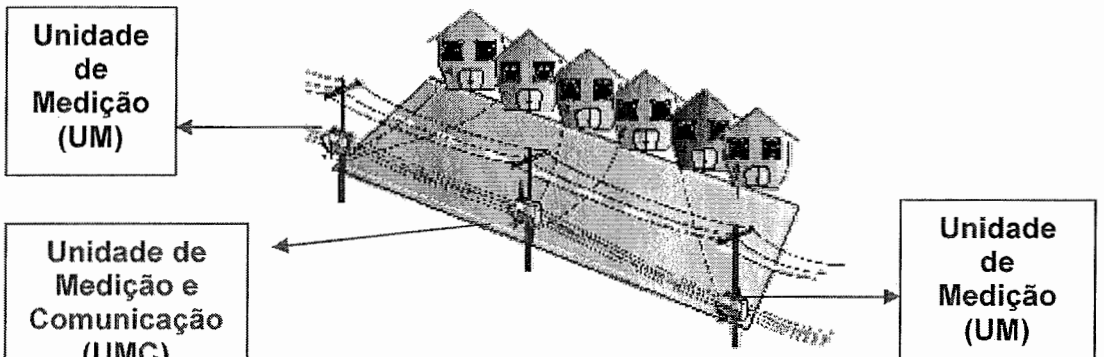


Figura 1 – Exemplo da situação real

A arquitetura do SMRP foi desenvolvida para permitir às concessionárias de energia elétrica, além da medição para faturamento, resolver problemas comuns ao seu dia-a-dia, principalmente reduzir os custos envolvidos com consumidores inadimplentes (especificamente corte e religamento) e o problema da verificação das perdas de energia. Este sistema é uma extensão do conceito de medição centralizada, desenvolvido pelo CEPEL e patenteado no Brasil, EUA e alguns países da Europa entre outros. No contexto de automação da distribuição, este sistema é denominado Sistema de Medição para Redução de Perdas (SMRP). Ele está sendo implantado através de um projeto piloto, na cidade de Porto Velho – RO, junto à concessionária local, a CERON. A Figura 2 apresenta a localização via satélite da quadra em Porto Velho em Rondônia onde está instalado o SMRP.

Figura 2 – Localização via satélite da quadra em Rondônia

Atualmente, a comunicação entre as Unidades de Medição é realizada com uma solução baseada em *Power Line Communication* (PLC) [2]. Esta solução apresenta problemas de custo, interferência, falta de padronização etc. O objetivo deste trabalho é estudar uma alternativa da comunicação baseada em *Power Line Communication* (PLC) por um sistema de comunicação sem fio baseado no protocolo IEEE 802.15.4/*ZigBee* [3] entre postes de redes de distribuição especificamente focando o sistema em Rondônia.

1.2 Sistema de Medição e Redução de Perdas (SMRP)

Apresenta-se na Figura 3 o diagrama esquemático da arquitetura do SMRP usada em Porto Velho. As unidades de medição colhem as informações de consumo de energia do consumidor e as-repassam para as UM vizinhas e assim sucessivamente até chegar á Unidade de medição e comunicação (centralizadora). Este trabalho se propõe a estudar esta comunicação, via protocolo *ZigBee*. Para que as informações de medição cheguem a sede da concessionária de energia, usa-se atualmente um *link* de rádio dedicado que não é estudado neste trabalho.

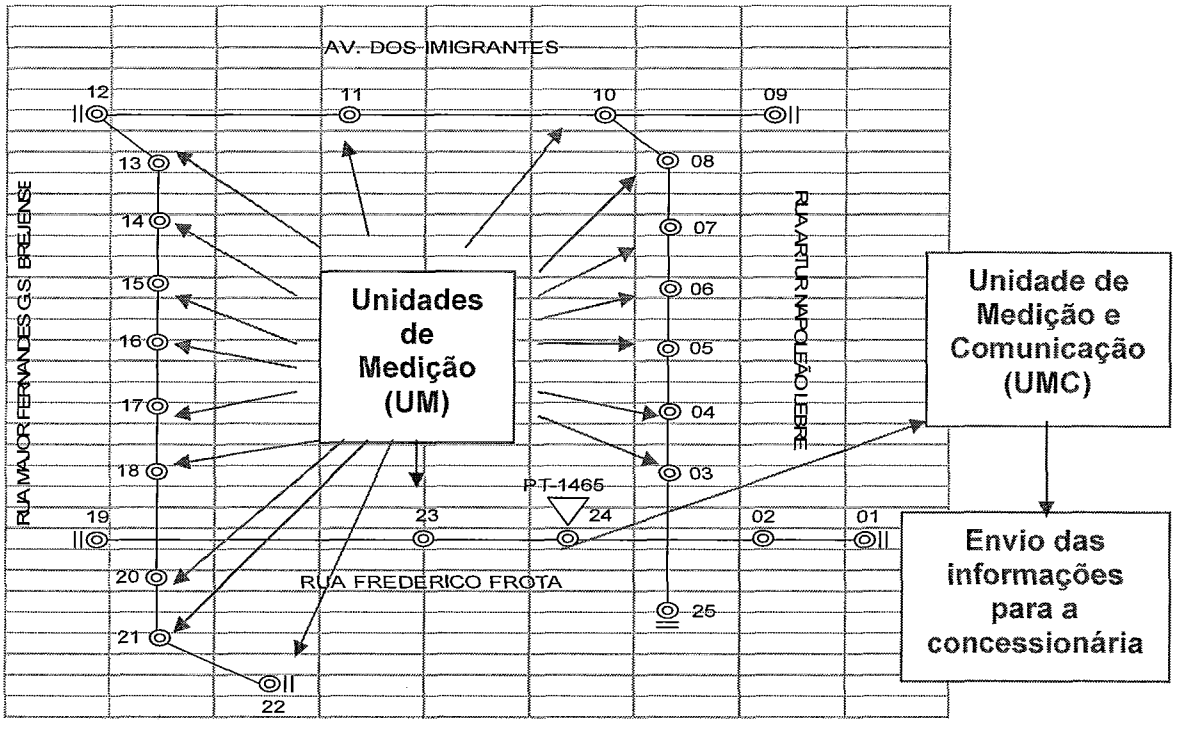


Figura 3 – Diagrama esquemático da arquitetura do SMRP

1.2.1 Componentes do sistema alvo de Medição:

Unidade de Medição: Cada poste possui uma caixa metálica chamada de Unidade de Medição (UM). Apresentam-se na Figura 4 os componentes de uma UM. Cada UM possui um medidor chamado de Módulo de Medição (MM) que é ligado diretamente a um consumidor e é responsável pelo consumo de energia elétrica desse consumidor. Junto com o MM, a UM ainda possui um módulo chamado de Módulo de Alimentação e Comunicação (MAC), é nele que são armazenados os dados de medição do consumidor. O MAC também cuida da transmissão de dados para as UM vizinhas. Os medidores são eletrônicos e têm a função de medir a energia ativa e reativa do consumidor. Além disso, também são capazes de realizar o corte e o religamento do consumidor remotamente, ou seja, não é necessário o envio de uma equipe ao campo para realizar estas operações.

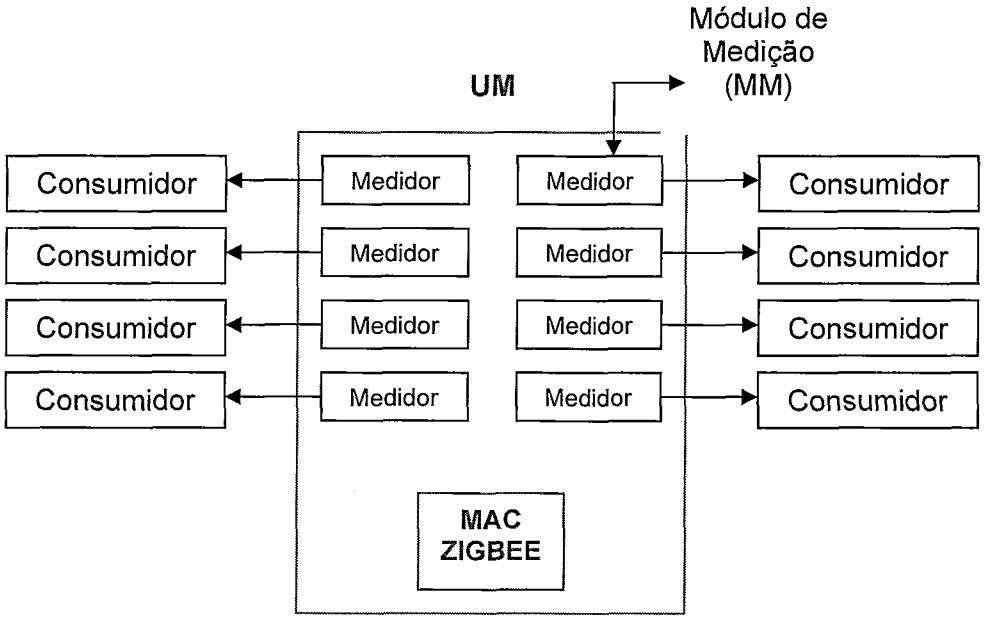


Figura 4 – Componentes da Unidade de Medição (UM)

A Figura 5 apresenta uma UM aberta usada no projeto.

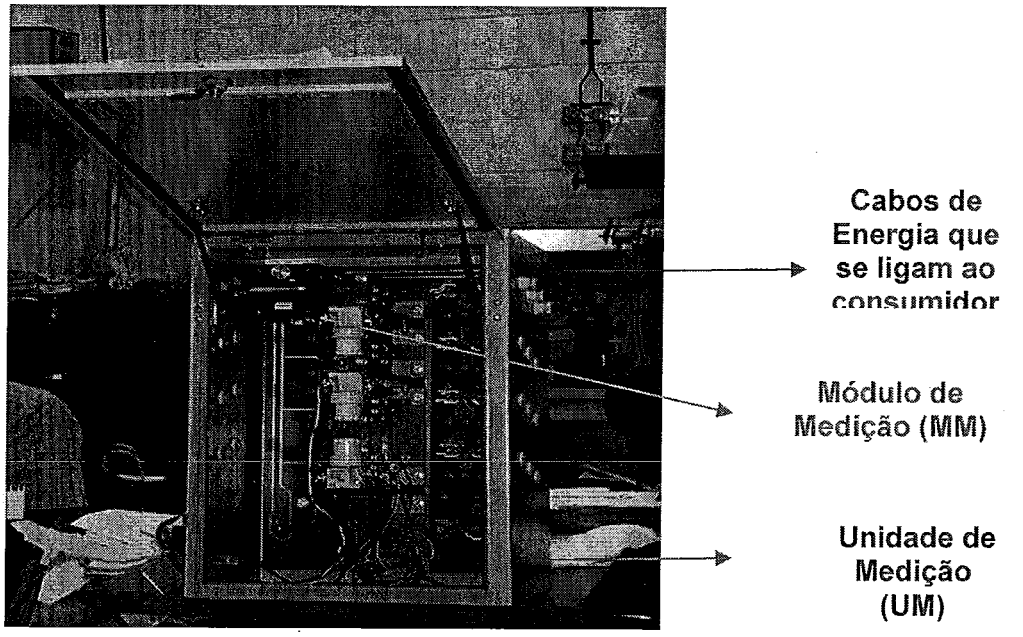


Figura 5 – Unidade de Medição aberta

Unidade de Medição e Comunicação (UMC): A UMC é a concentradora da rede. Apresenta-se na Figura 6 um diagrama esquemático da UMC. Observa-se que além do módulo MAC ela possui um transceptor de rádio para enviar as informações de medição para a concessionária.

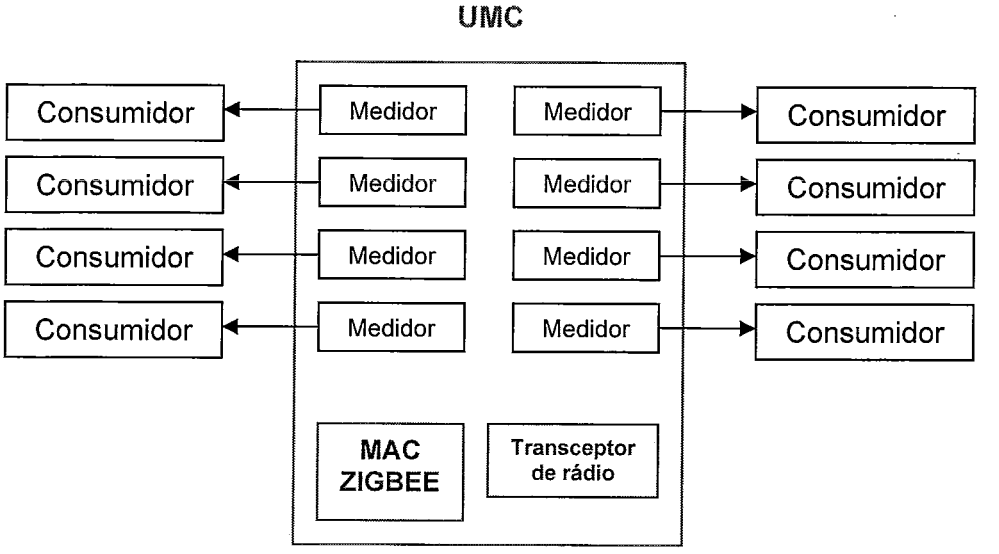


Figura 6 – Componentes da UMC

A Figura 7 mostra o caminho de como é realizada a medição do consumidor para a concessionária. A UM envia os dados de medição e faturamento do medidor em questão através do protocolo *ZigBee*, o MAC da UM armazena os dados e encaminha para o MAC da UM mais próxima, que esteja na rota da UMC, também através do protocolo de comunicação *ZigBee* e assim sucessivamente até chegar a UMC. A UMC se comunica com as UM através do MAC e se comunica com a concessionária através do seu transceptor de rádio. A UMC então, encaminha os dados de consumo a concessionária. Por esse caminho ser bidirecional (*full duplex*), a Figura 8 mostra o caminho inverso.

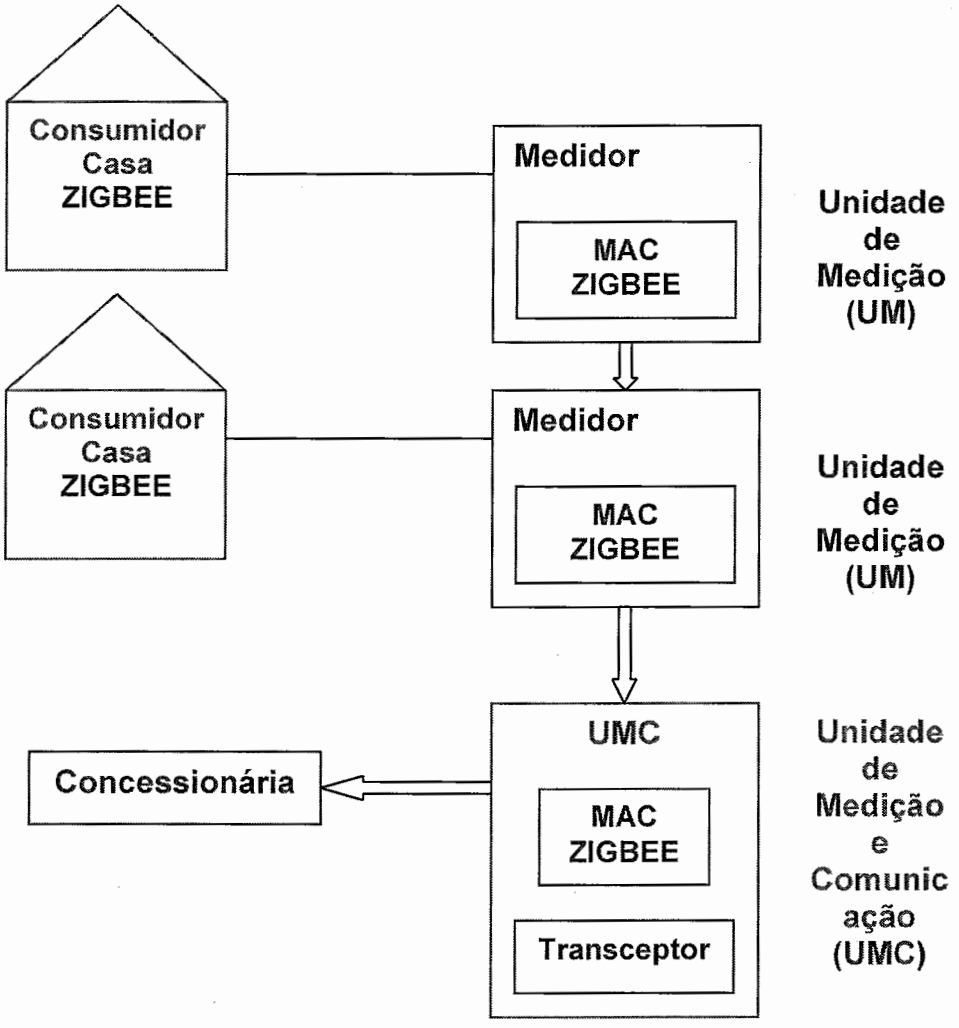


Figura 7 – Medição do consumidor para a concessionária

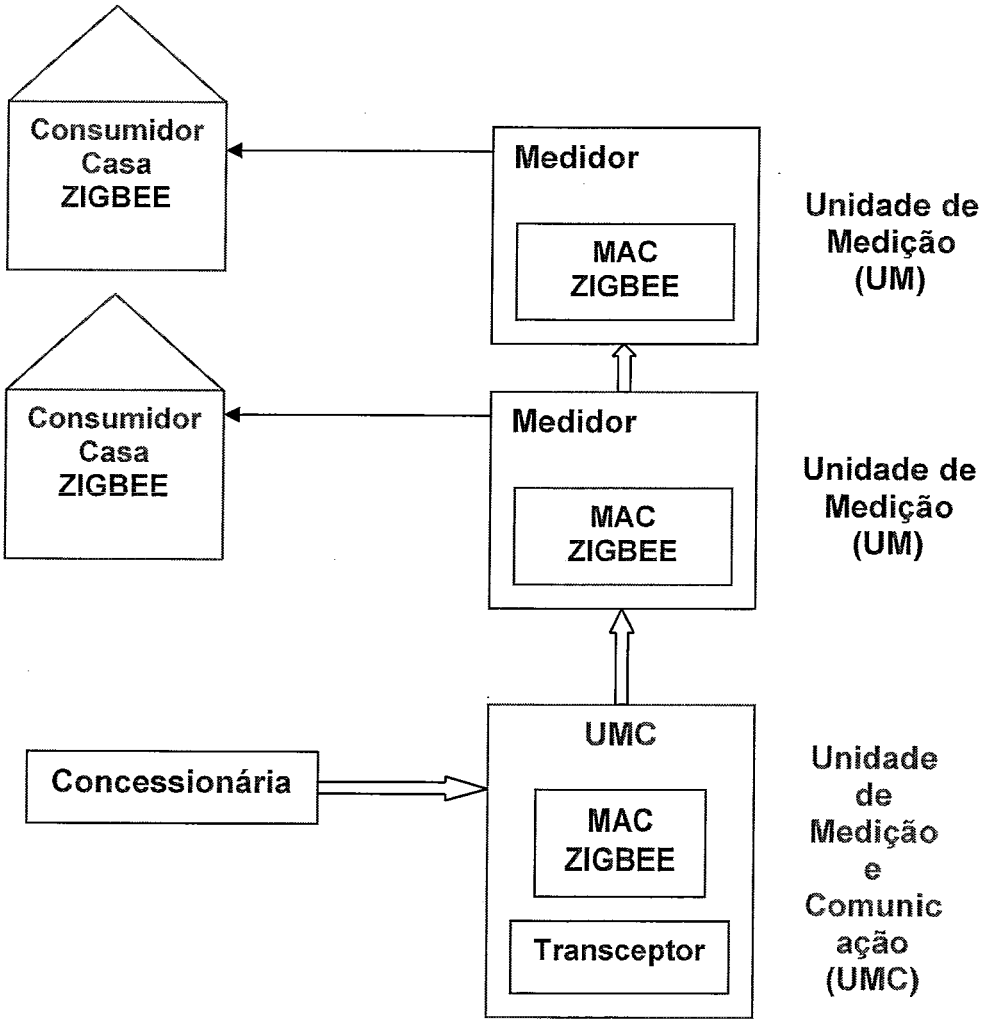


Figura 8 – Corte ou Religamento da Concessionária para o Consumidor

A concessionária envia para a UMC através do tranceptor de rádio o procedimento para o consumidor em questão (corte, religamento ou informações para o consumidor, por exemplo: mudança de tarifa), o MAC da UMC recebe o dado, armazena e o encaminha para o MAC da UM mais próxima e assim sucessivamente até chegar ao consumidor em questão.

Há ainda duas situações que podem ser utilizadas no SRMP. A Figura 9 apresenta o diagrama da UM nas duas situações: pode ser instalado um único equipamento de medição para atender a um máximo de oito consumidores ou pode ser instalado um único equipamento de medição para atender a apenas um consumidor.

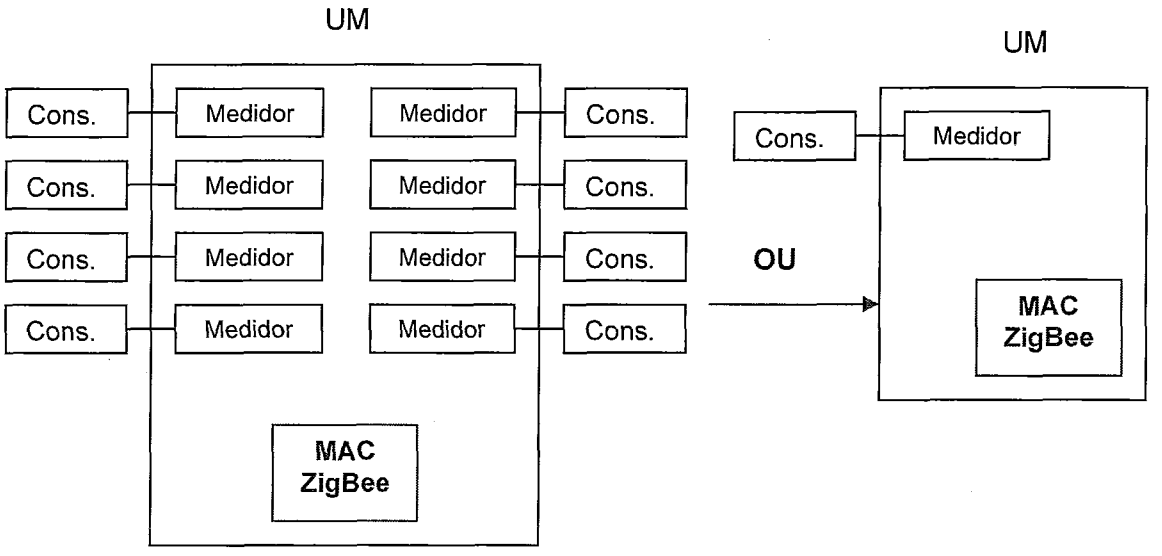


Figura 9 – Unidade de Medição (UM) utilizada em duas situações diferente

Pela dificuldade de visualização na simulação, para este trabalho foi utilizada a Unidade de Medição onde apenas um consumidor é ligado ao medidor. Caso não haja visada para as UM não terá para os consumidores também, ou seja, o número de consumidores não altera na conclusão deste trabalho. A topologia escolhida é necessária para qualquer processo de análise gráfica.

1.3 Alternativas de comunicação

Atualmente o SRMP está operando através da comunicação via *Power Line Communication* (PLC). *Os* PLC são utilizados pelas empresas de energia elétrica desde a década de 1920, pois transmitem dados e voz em banda larga pela rede de energia elétrica. Esses sistemas foram e ainda são utilizados para controle remoto e comunicações de voz. Os equipamentos são muito robustos e normalmente tem uma vida útil superior a trinta anos, porém os fabricantes estão deixando de produzi-lo por falta de demanda devido ao grande avanço da instalação de fibras ópticas, de redes wireless e principalmente devido ao barateamento dos sistemas de telecomunicações.

Uma das grandes desvantagens do PLC em relação ao protocolo *ZigBee* é que o PLC não é normalizado, ou seja, não existe um padrão que nem existe no *ZigBee* que é o padrão IEEE 802.15.4.

Quanto a confiabilidade, a instalação do PLC é incerta, pois ele pode não funcionar corretamente em todos os lugares principalmente quando os fabricantes são diferentes. Por exemplo, o fabricante de um PLC X pode não se comunicar com o fabricante de um PLC Y. Já no *Zigee*, o fabricante de um dispositivo *ZigBee* X poderá se comunicar com um dispositivo de um outro fabricante Y. Um dispositivo *ZigBit™ da Meshnetics* não terá dificuldades em se comunicar com um dispositivo do fabricante *Jennic* ou de quaisquer outros fabricantes.

Uma outra desvantagem do uso da PLC é em relação a interferência e aos ruídos, qualquer "ponto de energia" (um ponto de rede, onde, só é preciso plugar o equipamento de conectividade (que normalmente é um *modem*) na tomada e pode-se utilizar a rede de dados) pode se tornar um ponto de interferência, ou seja, todos os outros equipamentos que utilizam radiofreqüência, como receptores de rádio, telefones sem fio, alguns tipos de interfone e dependendo da situação, até televisores, podem sofrer interferência. Em alguns países existem movimentos contra a sua instalação.

Na rede elétrica, o PLC sofre interferências principalmente na parte da banda de rádio de onda média que é de 1,7 a 3 MHz e toda a onda curta que é de 3 a 30 MHz, eles ficam completamente prejudicadas e inutilizáveis. Outros equipamentos podem causar interferências em uma rede PLC como motores de escova e os dimmers de luz. Entre os motores domésticos, destacam-se as furadeiras elétricas. Recentemente, o custo de um PLC cotado na empresa *Yitran* [4] foi de $40,00. Por conta de todas essas desvantagens em relação ao uso do PLC, como alternativa, foi feito o estudo de duas tecnologias de comunicação sem fio do grupo 802.15 para ser utilizado nesse trabalho: *ZigBee* (802.15.4) e o *Bluetooth* (802.15.1) [5].

Por serem do mesmo padrão (802.15), eles têm algumas características em comum: operam na faixa de 2.4Ghz, modo *full duplex* na banda ISM e utilizam a freqüência de transmissão espectral (*Spread Sprectrum*). Porém também possuem algumas diferenças que foram essenciais para a escolha do padrão para ser utilizado neste trabalho. São elas:

- Pilha protocolar de implementação: a pilha do *ZigBee* é simplificada (código muito menor do que o código do *Bluetooth*), conduzindo a interfaces de baixo custo (*low cost)* [6];
- Possibilidade de suportar uma elevada densidade de nós por rede: *ZigBee* suporta um máximo de 65.535 dispositivos por cada coordenador de rede. No *Bluetooth* a rede *ad-hoc* é chamada de *Piconet*. Uma rede *Piconet* é formada por um dispositivo mestre e no máximo sete dispositivos escravos por rede;
- Topologias de rede: *ZigBee* suporta diversas topologias de rede como: estrela (*star*), malha (*mesh*) e árvore (*tree*) enquanto que no *Bluethooth* não existe uma topologia de rede pré determinada e muito menos um controle centralizado.
- Alcance: *ZigBee* 10 - 100m enquanto que o *Bluethooth* de 1 - 10m.
- Duração das Baterias: ZigBee de 100 a 1000 dias enquanto que o *Bluethooth* agüenta de 1 a 7 dias.
- Aplicação: *ZigBee* é mais apropriado para redes com muitos dispositivos, rede de sensores com pequenos pacotes de dados enquanto que o *Bluethooth* é mais apropriado para aplicações como sincronização de PCs, telefones celulares e PDAs.
- Segurança: *ZigBee* utiliza criptografia de 128 *bits* padrão *Advanced Encryption Standard* (AES) enquanto que o *Bluethooth* utiliza o algoritmo de criptografia E0 e SAFER+ de 64 ou 128 *bits* (evita escutas não autorizadas, mantendo a privacidade do canal de comunicação).
- Custo: Um kit de *ZigBee* (MC 13213) com conectores, antenas, 2.4Ghz, 16 canais, suporta as topologias *star, mesh e tree* da *Freescale* [7] , custa em torno de $ 4,10.

O *ZigBee* é um procotolo de sete camadas criado pela união do padrão IEEE 802.15.4 com a *ZigBee Alliance* [8]. Possui três topologias de rede (*star, mesh e tree*) e possui dispositivos para estruturar essas topologias (coordenadores, roteadores e end-devices). Esses dispositivos se dividem em FFD (*Full Function Device*) e RFD (*Reduce Function Device*). Um dispositivo FFD se comunica com todos os dispositivos da rede e um dispositivo RFD se comunica apenas com o dispositivo mais próximo a ele. O roteamento do *ZigBee* é feito de duas maneiras: *on-demand* e hierárquico (híbridos). O roteamento *sob-demand* só cria rotas quando estas são solicitadas pelo nó origem. Dois

dos protocolos de roteamento *on-demand* mais conhecidos são: *Ad Hoc On-Demand Distance Vector Routing* (AODV) [9,10] e *Dynamic Source Routing* (DSR) [11].

Uma vez que o *ZigBee* foi admitido como uma nova alternativa viável para comunicação entre as UM, faz-se necessário os estudos dos protocolos de roteamento para redes *Ad-hoc* no SRMP. Os protocolos de roteamento escolhidos são protocolos *on-demand*, ou seja, só criam rotas quando são solicitadas pelo nó origem. Dentre os protocolos reativos mais conhecidos estão o *Dynamic Source Routing* (DSR), *Ad Hoc On Demand Vector Routing* (AODV), *Temporally Ordered Routing Algorithm* (TORA) e *Associativity Based Routing Algorithm* (ABR).

No AODV, esses nós se comunicam entre si até chegar ao nó destino, possuem tabela de roteamento e utilizam o processo de descobrimento de rotas (inundam a rede com mensagens de roteamento) para encontrar um caminho e utilizam manutenção de rotas. O DSR possui roteamento pela fonte, ou seja, só o nó fonte sabe o caminho até o destino, ao invés da tabela de roteamento ele utiliza o *cachê* de rotas (local onde ficam armazenadas as rotas válidas ou não). Utiliza o processo de descobrimento de rotas e a manutenção de rotas, igual ao AODV.

1.4 Objetivos

O objetivo deste trabalho é apresentar os resultados do estudo dos protocolos de roteamento *on-demand, Ad Hoc On-Demand Distance Vector Routing* (AODV) e *Dynamic Source Routing* (DSR) voltados para o protocolo IEEE 802.15.4/*ZigBee* visando uma aplicação específica de medição: o SMRP. Foram escolhidos os protocolos de roteamento AODV e DSR por eles serem suportados pelo simulador de redes utilizado neste trabalho (NS-2), devido ao funcionamento dos mesmos serem adequados ao funcionamento do SRMP, pelo fácil entendimento do funcionamento de ambos e pelas inúmeras referências bibliográficas.

Existem protocolos reativos (*on-demand*), pró-ativos e híbridos. O *ZigBee* trabalha com os protocolos reativos e híbridos e o SMRP com os protocolos AODV ou DSR que são protocolos reativos. Ambos são protocolos *on-demand*, porém no AODV, os nós se comunicam entre si até encontrar um destino alcançavél, o mesmo acontece no SRMP.

1.5 Descrição do trabalho

Para estudar esses protocolos usar-se-á a topologia inicial de rede, a rota completa é composta por 24 UM, 24 consumidores e 1 UMC, como mostra a Figura 10.

Com o objetivo de induzir algumas falhas, foram simuladas algumas quebras de enlaces através da retirada de algumas UM e conseqüentemente a topologia foi alterada. Desta forma, foram simuladas sete estratégias para analisar o comportamento e o desempenho dos protocolos de roteamento. Para isso, foram observados dois parâmetros a serem analisados: pacotes recebidos com sucesso: constam da quantidade do número de pacotes que foram enviados e que chegaram ao destino com sucesso por unidade de tempo e pacotes perdidos: constam da quantidade do número de pacotes que foram enviados, porém não chegaram ao destino com sucesso.

Os protocolos de roteamento para redes *Ad-Hoc* que operam *on-demand* e que tenham a característica de múltiplos caminhos, tabela de roteamento e a troca de informação entre os nós como o AODV, são indicados para este tipo de topologia. Já o DSR utiliza roteamento pela fonte (somente o nó fonte conhece a rota completa até o nó destino), cachê em rota e múltiplas rotas para o mesmo destino, seu uso é menos indicado para este tipo de topologia de rede. De acordo com os resultados obtidos e analisados, conclui-se que o AODV foi o protocolo que apresentou o melhor desempenho nas diversas simulações realizadas.

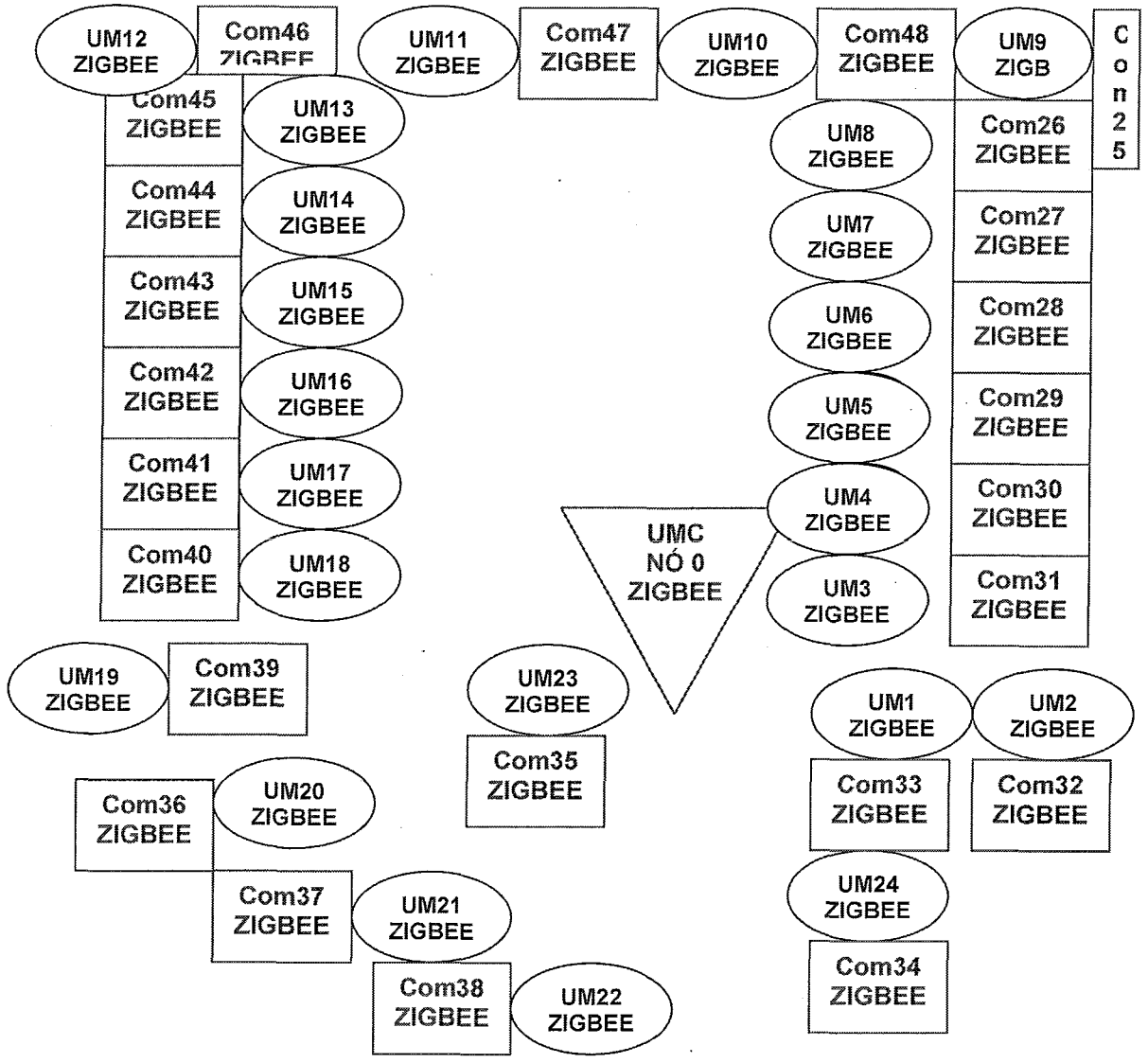


Figura 10 – Topologia inicial de rede

1.6 Trabalhos Relacionados

Há uma escassez de artigos que combinem os estudos de protocolos para redes *Ad-hoc AODV e DSR* utilizando o padrão 802.15.4/*ZigBee* simulando-os no NS-2. Têm-se apenas artigos que os tratam separadamente. Assim, para efeito dos estudos deste trabalho apresentam-se os seguintes trabalhos relacionados: em [12] avalia-se o comportamento dos protocolos de roteamento para redes ad-hoc (AODV e DSR) em um cenário que tenta retratar uma situação real de uma operação militar em um campo de batalha, onde nessas aplicações os nós se movem em grupo. Os protocolos de roteamento escolhidos foram o DSDV (pró-ativo), AODV e DSR (ambos *on-demand*). Para as simulações foi escolhido o simulador de redes *Network Simulator* (NS-2), porém foi gerado um modelo (*Mixed Way Point*) para atender as necessidades do trabalho utilizando a ferramenta *ScenGen* (gerador de cenários). Utilizou-se também o gerador de tráfego *Constant Bit Rate* (CBR). Os parâmetros analisados entre os três protocolos foram: taxa média de entrega, atraso médio (em segundos), pacotes de roteamento e pacotes entregues e os alcances também foram variados. Para este caso, o DSR foi o protocolo que apresentou um melhor desempenho em relação aos demais, já que ele é um protocolo sob demanda e que opera com múltiplas rotas.

Em [13] é feita uma comparação entre os protocolos de roteamento AODV e DSR em um cenário retangular de 1500m x 300m com 50 nós, as fontes de tráfego são de taxa de bits constante *Constant Bit Rate* (CBR). Os pares fonte-destino estão espalhados aleatoriamente pela rede. O número de pares fonte-destino e a taxa de transmissão de pacotes em cada par é variada para modificar a carga da rede. O modelo de mobilidade utiliza o *random waypoint*. Considera-se também que cada pacote inicia sua jornada a partir de um local aleatório para um destino aleatório com uma velocidade aleatória, uniformemente distribuída entre 0 e 20 m/s. Quando o destino é alcançado, um outro destino aleatório é escolhido após um tempo de pausa. A variação do tempo de pausa afeta a velocidade relativa dos nós. Cenários idênticos de mobilidade e tráfego são utilzados para os dois protocolos. Neste cenário, constatou-se que o AODV apresentou um melhor desempenho em cenários de alta carga da rede e de alta mobilidade dos nós.

Em [14,15] foi desenvolvido o suporte do padrão IEEE 802.15.4 para o NS-2 e foi feita várias experiências, entre elas: transmissão com e sem quadro de sinalização

(*beacon*); associação de dispositivos na rede; CSMA; transmissão de dados direta, indireta e reservada (GTS); colisões em função do tamanho do quadro de sincronismo (*superframe*). Os resultados deste estudo serviram de base neste trabalho.

Em [16] faz-se um estudo aprofundado do protocolo *ZigBee* utilizando o simulador *Network Simulator* (NS-2), compara-o com o protocolo de roteamento AODV e analisa a vazão e as taxas de entrega nas freqüências de 2.4Ghz e 902Ghz.

1.7 Estrutura do Trabalho

Este trabalho está estruturado da seguinte forma: Capítulo 1 é a Introdução que descreve os objetivos, a motivação e o Sistema de Medição para Redução de Perdas.

O Capítulo 2 descreve o protocolo *ZigBee* de uma maneira geral abordando o conceito, as características, as camadas, as topologias e o funcionamento.

O Capítulo 3 descreve os protocolos de roteamento para redes *ad-hoc*.

O Capítulo 4 apresenta o mapeamento do sistema alvo, o simulador utilizado, a metodologia experimental e a análise dos resultados obtidos.

O Capítulo 5 apresenta as conclusões obtidas com a realização do trabalho, as dificuldades encontradas e os trabalhos futuros.

No próximo capitulo será descrito o protocolo *ZigBee* de uma maneira geral abordando o conceito, as características, as camadas, as topologias e o seu funcionamento.

Capítulo II

ZigBee

2.1 Redes *Wireless*

Um número cada vez maior de usuários de redes de computadores tem optado por soluções baseadas em topologias sem fio (*wireless*), ao invés de redes com cabeamento convencional, especialmente quando se trata de ampliação ou melhoria de uma rede existente.

Os avanços recentes das tecnologias de redes sem fio possibilitaram o surgimento de várias alternativas e padrões de implementação mas até recentemente a grande maioria tinha como premissa principal prover um conjunto de protocolos que garantissem a qualidade para a transmissão de voz ou de dados com altas taxas de transferência, o que tornava os equipamentos bastante caros e pouco atraentes para outras aplicações mais simples.

As redes *wireless* também conhecidas como redes sem fio (padrão IEEE 802.11) foi uma das grandes novidades tecnológicas dos últimos anos. Existem atualmente quatro grandes grupos:

2.1.1 WPAN (*Wireless Personal Area Network*)

Neste grupo temos as tecnologias *wireless* de pequeno alcance (10 -100 metros). Para padronizar o desenvolvimento de tecnologias WPAN, o IEEE [17] estabeleceu o grupo de trabalho 802.15 para WPAN. Fazem parte deste padrão: *ZigBee* (802.15.4), *Bluetooth* (IEEE 802.15.1) e *Ultra-wide band* (UWB) [18] que faz parte do padrão IEEE 802.15.3. Esse grupo também é conhecido como LR-WPAN (LR - *Low Rate).*

O protocolo ZigBee da WPAN foi desenvolvido para se tornar uma alternativa de comunicação em redes que não necessitem de soluções mais complexas para seu controle, barateando assim os custos com a aquisição, instalação de equipamentos, manutenção e mão de obra. Trata-se de uma tecnologia relativamente simples, que utiliza

um protocolo de pacotes de dados com características específicas sendo projetado para oferecer flexibilidade quanto aos tipos de dispositivos que pode controlar. Veremos o protocolo *ZigBee* com detalhes mais a frente.

2.1.2 WLAN (*Wireless Local Area Network*)

As tecnologias WLAN permitem que os usuários estabeleçam conexões sem fio em uma área local (por exemplo: em um prédio corporativo ou de um campus ou em um espaço público, como um aeroporto). As WLAN podem ser usadas em escritórios temporários ou em outros espaços em que a instalação extensiva de cabos teria um custo muito elevado ou para complementar uma LAN existente de modo que os usuários possam trabalhar em diferentes locais em um prédio, em diferentes horários. Em 1997, o IEEE aprovou o padrão 802.11 para WLAN que especifica uma taxa de transferência de dados de 1 a 2 megabits por segundo (Mbps). As WLAN podem funcionar de duas maneiras distintas: WLAN de infra-estrutura e WLAN *Ad-Hoc*.

2.1.2.1 Redes infra- estruturadas (*Infrastructure Basic Service Set*)

As redes infra-estruturadas necessitam de um *Access point* (AP) para permitir a comunicação entre a rede sem fio e à rede convencional, estabelecendo a comunicação entre os diversos clientes.

Nas redes infra-estruturadas, toda comunicação entre os nós móveis é feita através de Estações de Suporte à Mobilidade (ESM). Neste caso, os nós móveis, mesmo próximos uns dos outros, estão impossibilitados de realizar qualquer tipo de comunicação direta entre si. As estações de suporte à mobilidade (ESM) podem estar conectadas a conectores (*gateways)* que permitem a comunicação entre os nós móveis e a parte fixa da rede [19, 20]. A Figura 11 mostra uma rede infra – estruturada:

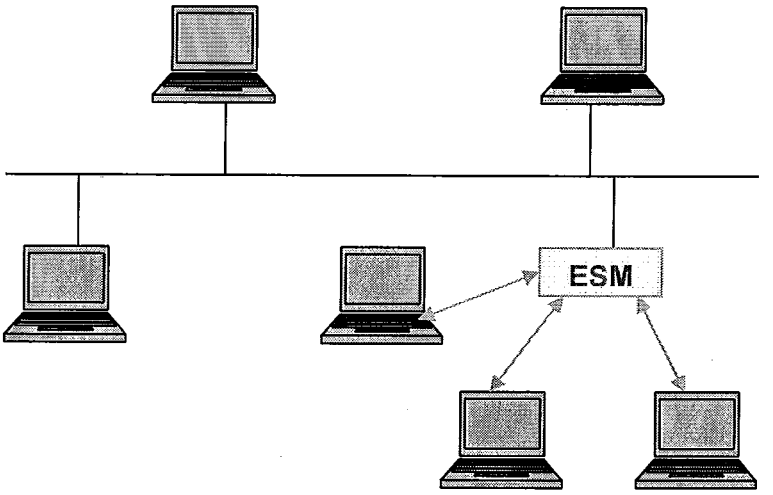


Figura 11 - Rede infra – estruturada

2.1.2.2 Redes *Ad-Hoc (Independent Basic Service Set* (IBSS))

A comunicação entre as estações de trabalho é estabelecida diretamente sem a necessidade de um *Access Point* (AP) e não precisa de uma rede física para conectar os nós. Em uma rede *ad-hoc*, todos os nós móveis são aptos a se comunicar diretamente entre si. Não existem pontos de acesso, não existem estações de suporte à mobilidade.

Os nós de uma rede *ad-hoc* podem se mover arbitrariamente, não há uma topologia pré-determinada. Deste modo, a topologia da rede muda freqüentemente e de forma imprevisível. A conectividade entre os nós móveis muda constantemente, exigindo uma constante adaptação e reconfiguração de rotas. São referenciadas pelo IEEE como MANET (*Mobile Ad-hoc NETwork*). O roteamento é a principal função da camada de rede de uma rede ad-hoc, pois consiste na determinação de uma rota ligando um dispositivo origem até um dispositivo destino na rede. Isso se torna muito mais difícil devido á mobilidade. A Figura 12 exemplifica uma rede *ad-hoc*:

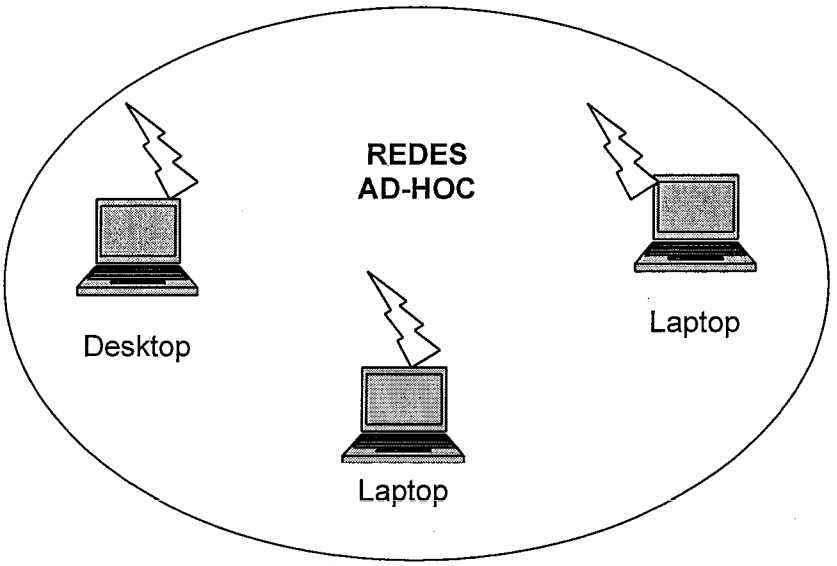


Figura 12- Redes ad-hoc

2.1.3 WMAN (Wireless Metropolitan Area Network)

As tecnologias WMAN permitem que os usuários estabeleçam conexões sem fio entre vários locais em uma área metropolitana (por exemplo: entre vários prédios de escritórios em uma cidade ou em um campus universitário) sem o custo elevado proveniente da instalação de cabos de cobre ou fibra e da concessão de linhas. As WMAN utilizam ondas de rádio ou luz infravermelha para transmissão de dados [21].

2.1.4. WWAN (Wireless Wide Area Network)

As tecnologias WWAN permitem que os usuários estabeleçam conexões sem fio em redes remotas privadas ou públicas. Essas conexões podem se sustentar através de grandes extensões geográficas, como cidades ou países através do uso de sites com várias antenas ou sistemas de satélite mantidos por provedores de serviços sem fio. As tecnologias WWAN atuais são conhecidas como sistemas de segunda geração (2G).

2.2 Protocolo *ZigBee*

O nome *ZigBee* vem do ziguezague das abelhas (*Bee*). O nome foi criado a partir da analogia entre o funcionamento de uma Rede *Mesh* e o modo como as abelhas trabalham e se locomovem. As abelhas que vivem em colméia voam em Zig...Zag e dessa forma, durante um vôo a trabalho em busca de néctar, trocam informações com outros membros da colméia sobre distância, direção e localização de onde encontrar alimentos. Uma malha (*Mesh*) *ZigBee* dispõe de vários caminhos possíveis entre cada nó da rede para a passagem da informação, assim, é possível eliminar falhas se um nó estiver inoperante simplesmente mudando o percurso da informação [22].

ZigBee é um protocolo que já foi chamado de *Home RF Lite* quando a tecnologia era da empresa *Philips*. O *ZigBee* foi homologado em 2003 pelo IEEE sob o número 802.15.4 e é utilizado para comunicação de redes sem fio.

O ZigBee surgiu através de uma Aliança chamada *ZigBee Alliance*. A *ZigBee Alliance* e o padrão IEEE trabalham em conjunto para desenvolver um protocolo capaz de possibilitar um controle seguro e de baixo custo em redes sem fio. Cada um aprimora-o da sua maneira. A utilização do padrão 802.15.4 é gratuita. Para ser *participants* da *ZigBee Alliance* é necessário o pagamento de U$ 9,500/ano, para ser *promoter* é necessário o pagamento de $ 40,00/ano e para ser *adopter*, $ 3,50/ano. É uma licença de uso onde as empresas pagam para participar das pesquisas e geração de novos segmentos. A *ZigBee Alliance* é uma aliança constituída por mais de 200 empresas oriundas de mais de 20 países distintos, na qual se integram também especialistas da área de telecomunicações e semicondutores. São elas: *Morotola (Freescale), Eaton, Siemens, Philips, Bosh, Analog Device, Samsung, Texas Instruments, Microchip* etc. Dessas empresas, a *Freescale, Philips, Siemens, Samsung, Texas Instruments* etc são *Promoters*. *Atmel, Eaton, Daintree Networks, Jennic, Meshnetics* etc são *Participants*. *CROW Electronic Engineering, Comverge, Inc.* etc são *Adopters*.

A arquitetura do protocolo *ZigBee* é estruturada em sete camadas segundo o modelo de referência *Open Systems Interconnection* (OSI). As camadas *Médium Access Control* (MAC) e *Physical Layer* (PHY) seguem a definição do padrão IEEE 802.15.4. As Camadas de Rede (NWK), Suporte a Aplicação (APS) e Aplicação (APL) são definidas pela *ZigBee Alliance* como mostra a Figura 13.

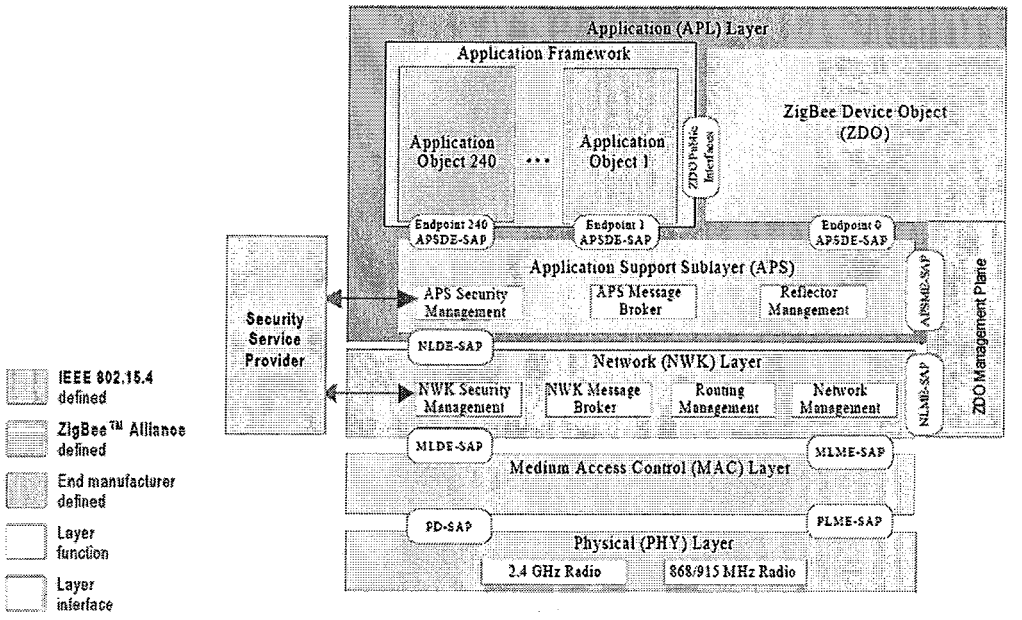


Figura 13 - Arquitetura do ZigBee

Cada camada executa serviços específicos ao dispor da camada superior. Analisando a Figura acima, percebemos algumas interfaces entre uma camada e outra. Essas conexões da camada superior com a camada abaixo são conhecidas como um SAP (*Service Access Point*) e cada SAP suportam um número de interfaces de serviço para ativar a funcionalidade que se pretende solicitar [23]. Por exemplo: as interfaces da camada física (PHY) que fazem ligação com a camada MAC são as entidades de dados que fornecem dados para o serviço de transmissão e a entidade de gerenciamento que fornecem informação para todos os outros serviços conhecidos como: *Physical Layer Data – Service Access Point* (PD-SAP) e *Physical Layer Management Entity – Service Access Point* (PLME-SAP), onde PD significa *PHY* Dados e PLME significa *PHY* Gerenciamento. Assim como as interfaces que fazem a ligação da camada MAC e da camada de rede (NWK) são conhecidas como: *Médium Access Control Layer Management Entity – Service Access Point* (MLME-SAP) e *Médium Access Control Layer Data Entity – Service Access Point* (MLDE-SAP), onde MLME é o gerenciamento da camada MAC e MLDE é referente aos dados da camada MAC e assim sucessivamente.

O protocolo *ZigBee* possui as seguintes características abaixo [24]:

- Alcance: 10 - 100m;
- Consumo de energia baixo (*low power*);
- Pilha protocolar de implementação simplificada (código de tamanho reduzido) conduzindo a interfaces de baixo custo *low cost*;
- Bateria: boa durabilidade;
- Utilização: simples;
- Taxa de Transferência de dados: 20Kbps a 250Kbps;
- Velocidades de conexão: 10Kbps e 115Kbps;
- Exelente imunidade contra interferência;
- Possibilidade de suportar uma elevada densidade de nós por coordenador de rede (num máximo de 65535 dispositivos por cada coordenador) e múltiplos "coordenadores de rede" podem ser ligados para suportar redes extremamente grandes;
- Admite diferentes topologias da rede: estrela (*star*), malha (*mesh*) ou árvore (*cluster tree*), permitindo o estabelecimento de redes de nós *ad-hoc*;
- Possui dois estados de operação: ativo: quando há envio ou recepção de dados ou *sleep* (estado de dormência) consumindo um mínimo de energia;
- Possui um tempo de ligação à rede menor que os outros protocolos e apresenta maior rapidez na passagem do modo sleep á ativo, o *ZigBee* apresenta também uma latência baixa *low latency*;
- Possui dois modos de operação da rede: *beacon* e *non-beacon;*
- O padrão *ZigBee* foi criado para economizar ao máximo energia. Com isso, é possível criar dispositivos sensores remotos alimentados com pilhas ou baterias comuns, que durarão meses ou mesmo anos sem precisarem ser substituídas;
- Elevada segurança com recurso a 128bits padrão *Advanced Encryption Standard (AES);*

2.2.1 Camadas que fazem parte do padrão 802.15.4

Fazem parte desse padrão as camadas: Camada Física (PHY) e a Camada de Controle de Acesso ao Meio (MAC).

2.2.1.1 Camada Física (PHY)

A camada física é responsável pela ativação e desativação do transceptor de rádio, seleção do canal de freqüência e transmissão e recepção de dados. Usam as faixas de freqüência *Industrial Scientific Medical* (ISM), ou seja, não requerem licença para funcionamento. São elas: 2.4GHz ISM, 16 canais (Global); 915MHz ISM: 10 canais (America do Norte) e 868MHz: 1 canal (Europa). Nesse contexto, a taxa de transferência dos dados é de até 250kbps na frequência de 2.4Ghz, operando com 16 canais, 40kbps na frequência de 915Mhz operando com 10 canais e 20kbps na frequência de 868Mhz operando com um canal.

Em termos de modulação é utilizado *Offset quadrature phaseshift keying* (O-QPSK) para a banda dos 2.4Ghz e *Binary phase shift keying* (BPSK) para os 915Mhz ou 868Mhz.

A camada física utiliza uma técnica conhecida como radio *Spread-Spectrum* (Técnica de Espalhamento Espectral) o qual se divide em: *Direct Sequence Spread Spectrum* (DSSS) e *Frequency Hopping Spread Spectrum* (FHSS) [25].

DSSS é um esquema de modulação *spread-spectrum* de seqüência direta utilizado pelo *ZigBee* enquanto que o FHSS é um esquema de modulação spread-spectrum por saltos de frequencia. O DSSS é que gera um padrão redundante de bits para cada bit transmitido. O padrão de bits, chamado chip ou código de chip, permite aos receptores filtrar sinais que não utilizam o mesmo padrão incluindo ruídos ou interferências. O código de chip cumpre duas funções principais: na primeira, ele identifica os dados para que o receptor possa reconhecê-los como pertencentes a determinado transmissor. O transmissor gera o código de chip e apenas os receptores que conhecem o código são capazes de decifrar os dados. Na segunda, o código de chip distribui os dados pela largura de banda disponível. Os chips maiores exigem maior largura de banda, mas permite maior probabilidade de recuperação dos dados originais. Ainda que um ou mais bits do chip sejam danificados durante a transmissão, a tecnologia incorporada no rádio recupera os dados originais usando técnicas estatísticas sem necessidade de retransmissão. Os receptores não desejados em banda estreita ignoram os sinais de DSSS, considerando-os como ruídos de potência baixa em banda larga.

2.2.1.2 Camada MAC (Médium Access Control)

É a camada responsável pela transmissão dos dados. Utiliza mecanismos de prevenção de colisão como o CSMA/CA para o qual efetua comunicações com a camada inferior (Camada Física). Além disso, especifica o tipo de dispositivos permitidos na rede (topologias de rede) e faz a sincronização e a transmissão de *beacons* (sinalização), permitindo a confiabilidade da operação.

2.2.1.2.1 Topologias de Rede do *ZigBee*

Numa rede *ZigBee* são identificados dois tipos de dispositivos: FFD e RFD

FFD (*Full Function Device*): São dispositivos mais complexos e precisam de um *hardware* mais potente para a implantação da pilha de protocolos, conseqüentemente, consomem mais energia. Numa topologia de rede *ZigBee* eles podem assumir o papel de coordenador, roteador ou até mesmo de um *end-device*. Comunica-se com todos os nós da rede.

RFD (*Reduce Function Device*): São dispositivos mais simples, onde sua pilha de protocolo pode ser implementada usando os mínimos recursos possíveis de *hardware*. Só podem se comunicar com dispositivos FFD (coordenador ou roteador). Numa topologia de rede *ZigBee* eles assumem o papel de *end-device.*

No padrão *ZigBee* existem três classes de dispositivos lógicos: coordenador, roteador e *end-device*, que definem a rede:

Coordenador: Também chamado de *ZigBee Coordinator* (ZC), tem as seguintes funções:

- É o responsável pela rede, pela entrada de algum dispositivo novo atribuindo a ele um novo endereço e pela manutenção dela;
- Existe somente um coordenador para cada rede;
- Um coordenador pode ter no máximo 65.535 (2^{16}) nós ligados a ele;
- São dispositivos de função completa (FFD).

Roteador - Também chamado de *ZigBee Router* (ZR), tem as seguintes funções dentro da rede:

- Não é usado na topologia em estrela, pois nessa topologia não há roteamento.
- Agem como dispositivo intermediário, roteando dados para outros dispositivos;
- Podem ser FFD ou RFD (*Reduce Function Device*)

End-Device - Também chamado de *ZigBee End-Device* (ZED), tem as seguintes funções:

- Requer menos memória, pois não precisa armazenar informações de roteamento, é mais barato que um roteador e que um coordenador *ZigBee*.
- Geralmente são RFD mas podem ser FFD também caso a topologia seja em estrela (*star*).

Em uma rede *ZigBee* [26] os dispositivos podem permanecer por longos períodos sem se comunicar com outro dispositivo, seu tempo de acesso conectado é de 30 ms. Por essas características que a tecnologia do *ZigBee* é muito econômica em relação ao consumo de energia podendo durar muito mais tempo que outros dispositivos de comunicação sem fio.

Os *ZigBees* são iguais em relação as características, frequencias, taxas de transmissão, camadas etc. A única coisa que difere é que cada fabricante tem a sua própria topologia e a sua própria aparência, ou seja, o *ZigBee* varia de fabricante para fabricante, isso significa que cada *kit* de desenvolvimento é diferente um do outro, porém o fabricante de um dispositivo *ZigBee* X poderá se comunicar perfeitamente com um dispositivo de um outro fabricante ZigBee Y. Por exemplo: um dispositivo *ZigBit da Meshnetics* não terá dificuldades em se comunicar com um dispositivo do fabricante *Jennic* ou de quaisquer outros fabricantes.

Como a topologia do SMRP é grande em relação ao padrão e como são muitos dispositivos, é sugerida pela maioria dos fabricantes a topologia em Rede *Mesh*. A maioria dos fabricantes trabalha com as topologias *Mesh e Tree* (Árvore), onde os dispositivos têm a mesma aparência.

2.2.1.2.2 Transmissão de dados do *ZigBee*

A transmissão de dados é feita de três maneiras [27]: dados são enviados do coordenador para os dispositivos via *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA), dados são enviados do dispositivo para o coordenador via *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) e dados são enviados do dispositivo para o dispositivo via *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA). O funcionamento do CSMA/CA pode ser resumido da seguinte maneira:

- Caso o meio esteja livre por um intervalo de tempo maior que o período de 50 ms, denominado de DIFS (*Distribution Coordination Function Interframe Space*), então, a estação pode transmitir;
- Caso o meio esteja ocupado, a estação executa o procedimento de *backoff*, ou seja, deve esperar o meio ficar desocupado para então poder transmitir;
- As estações que não transmitiram devem esperar até a primeira estação terminar de transmitir e só então poderão tentar transmitir de novo.

O CSMA/CA tem como objetivo evitar que aconteçam colisões, já que as mesmas são difíceis de serem detectadas em ambientes sem fio. O CSMA/CA ocorre na camada física também.

2.2.1.2.3 Modos de Operação de uma rede *ZigBee*

Há dois tipos de modo de operação numa rede *ZigBee*: *beacons* e *non-beacons*.

Modo *Beacon:* *Beacons* são sinalizações que são utilizadas para sincronizar os dispositivos na rede. Essa sinalização é feita de tempo em tempo para sincronizar os dispositivos (roteadores) que não foram localizados na rede (caso a rede seja dinâmica) e avisá-los de que haverá uma transmissão de dados. O coordenador desativa o modo *beacon* para modo *non-beacon*, por *software*. O intervalo entre os *beacons* pode variar entre os 15.36 ms e 251.65s, para uma taxa de transmissão de 250kbit/s.

Os outros dispositivos da rede só precisam estar ativos no momento da sinalização, mas esses dispositivos devem ser configurados para perceber o período em

que ocorrerá esta sinalização, pois no modo *beacon* a maioria dos dispositivos permanecem dormindo (*sleep*). Nesse modo, o consumo de energia é o mínimo possível.

Modo *Non-Beacon*: Nesse modo a maioria dos dispositivos de rede permanece sempre com seus receptores ativos, consumindo mais energia, isso porque, eles podem receber a sinalização a qualquer momento. É importante notar que nesse modo, os dispositivos devem ser alimentados com fontes de energia mais potentes e duradouras que pilhas ou baterias comuns.

A Figura 14 mostra a transmissão de dados do coordenador para o dispositivo com o modo de operação *beacon* ativo, onde o coordenador armazena os dados para que o dispositivo faça o contato através de um *data request*. Quando o dispositivo receber um *acknowledge* (quando o *data request* é recebido com sucesso), ele envia os dados através do CSMA/CA ao dispositivo e este responde com um *acknowledge* para avisar que o pacote foi recebido com sucesso, como veremos na Figura 14.

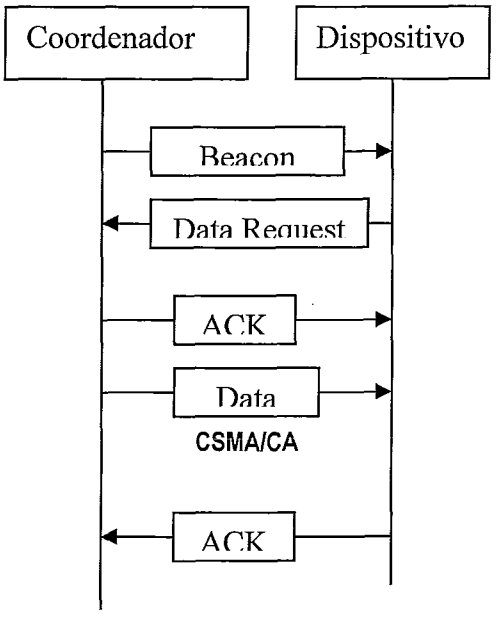


Figura 14 – Transmissão de dados do coordenador para o dispositivo

A Figura 15 mostra a transmissão de dados do dispositivo para o coodenador com no modo *beacon*, seguindo o mesmo padrão da Figura anterior.

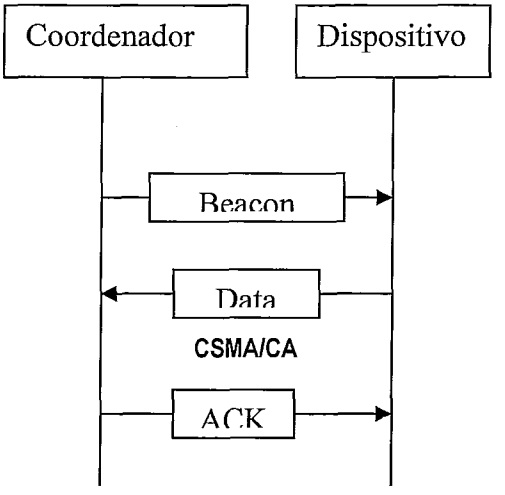


Figura 15- Transmissão de dados do dispositivo para o coordenador

A Figura 16 mostra a transmissão de dados de um dispositivo para outro dispositivo no modo *non-beacon* seguindo o mesmo padrão das Figuras anteriores:

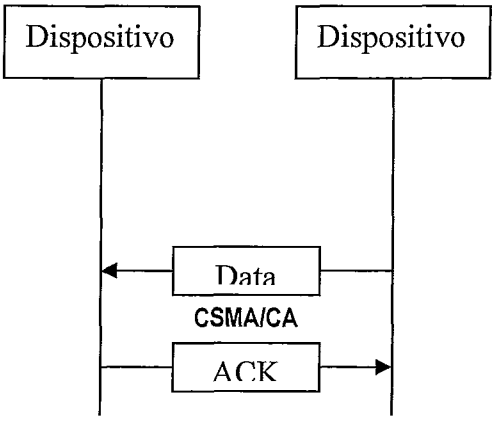


Figura 16 – Transmissão de dados de um dispositivo para um dispositivo

Para transmitir o *beacon* mostrado nas Figuras acima, o coordenador utiliza um quadro chamado de *beacon frame*. A Figura 17 mostra a estrutura de um *beacon frame*:

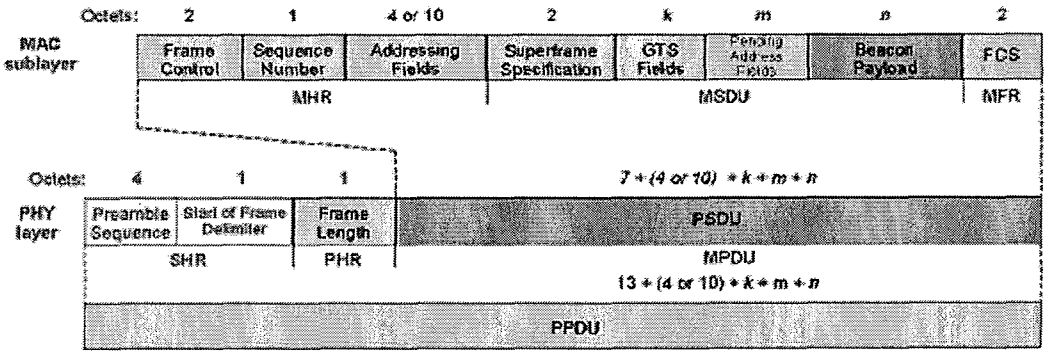


Figura 17- *Beacon Frame*

O *beacon frame* é dividido em três partes na Camada Mac*: MAC sub-layer service data unit* (MSDU) que são os dados, provenientes da camada superior; *MAC Header* (MHR) que contém: um campo de controle de 2 Bytes (FC - *Frame Control*) e 1 Byte para número de seqüência e de 4 a 20 bytes para campo de endereçamento; *MAC footer* (MFR) é composto de 16 bits *Frame Check Sequence* (FCS). A união do MSDU, do MHR e do MFR, forma o MAC protocol data unit (MPDU). O *Médium Access Control PHY Data Unit* (MPDU) é então passado para a camada PHY formando o *PHY service data unit* (PSDU), que são os dados do comprimento variável vindos da camada MAC.

O PSDU é formado pelo *Start of frame Delimiter* (SFD) que permite ao dispositivo receptor sincronizar com o feixe de bits, através de 4 bytes correspondentes ao campo *Preamble Sequence* (PS) e um byte no campo SFD e pelo *PHY header* (PHR) que tem um campo de 1 Byte (FL - *Frame Length*) que contém a informação do comprimento em bytes do quadro PSDU. A união do SHR, PHR e PSDU forma o *PHY protocol data unit* (PPDU). O MPDU e o PPDU correspondem a transmissão e a recepção de beacons.

Para a transmissão dos dados é utilizado o *Data Frame*, como mostra a Figura 18:

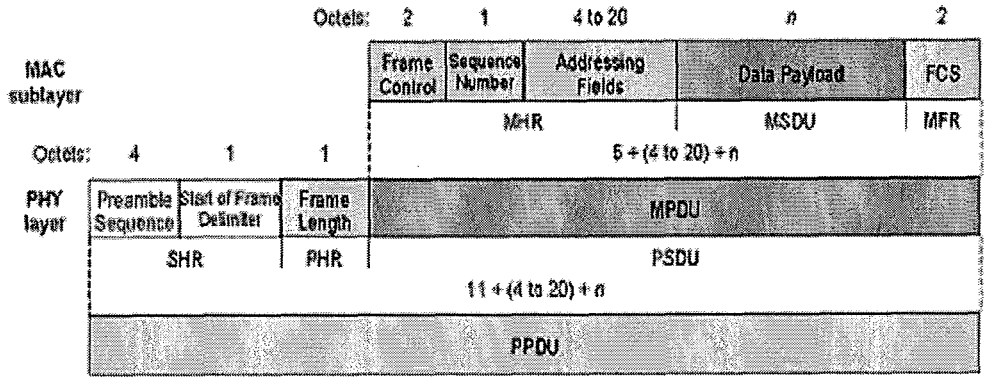


Figura 18- *Data Frame*

O *Medium access control sub-layer service data unit* (MSDU) é formado pelo *Data Payload* que é o campo onde serão transmitidos os dados. A união do: MHR com o MSDU e com o MFR formam o *MAC protocol data unit* (MPDU). O MPDU é passado para a camada PHY e é chamado de *PHY Service Data Unit* (PSDU). A união do SHR com o PHR e com o PSDU forma o *PHY protocol data unit* (PPDU). Logo, o MPDU E O PPDU correspondem a transmissão e recepção de dados.

Para a confirmação e recepção bem sucedida de um pacote, é usado o *Acknowledge frame*, como mostra a Figura 19:

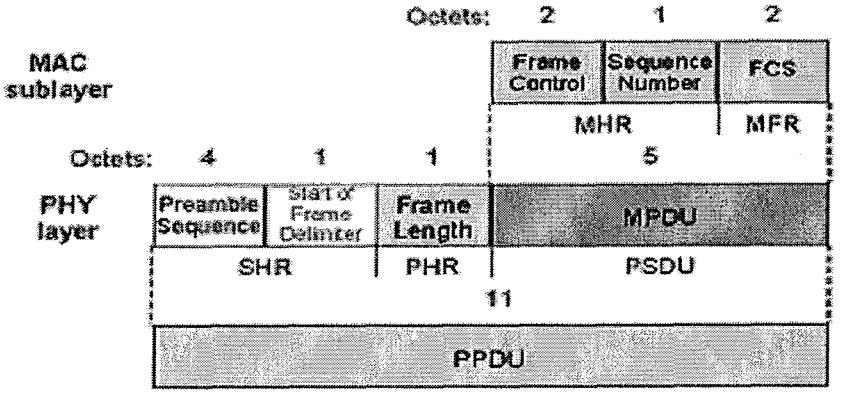


Figura 19 – *Acknowledge Frame*

A união do MHR com o MFR forma o *MAC protocol data unit* (MPDU) e a união do SHR com o PHR e com PSDU formam o *PHY protocol data unit* (PPDU). O MPDU e o PPDU correspondem a transmissão e a recepção de *acknowledges*.

2.2.2 Camadas que fazem parte da *ZigBee Alliance*

Fazem parte da *ZigBee Alliance: Security Sevice Provider* (SSP) que fornece os serviços de segurança em cada nível é usado pelas camadas MAC, NWK e APS onde a camada responsável pela comunicação dos dados é responsável por codificá-lo quando envia e autenticá-lo quando recebe. A camada MAC, por exemplo, utiliza o padrão *Advanced Encryption Standard* (AES) como seu algoritmo de criptografia descrevendo uma variedade de rotinas de segurança; Camada de Aplicação (APL), que contém: *a sub-layer Aplication Support Sublayer* (APS), o *ZigBee Device Object* (ZDO), *Application Framework* (AF) e a Camada de rede (NWK).

Uma das camadas que exerce grande responsabilidade pelo funcionamento do *ZigBee* é o *ZigBee Device Object* (ZDO). Ele é responsável pelo papel do dispositivo na rede (*Gerência de Discovery*), se o dispositivo é coordenador ou não. É responsável também pelas chaves de segurança na rede (*Gerência de Security*) e pela união de dois ou mais dispositivos considerando suas necessidades e definindo os serviços que eles oferecem (*Gerência de Binding*). Além de tudo, o ZDO tem o seu próprio perfil, chamado de *ZigBee Device Profile* (ZDP). É o ZDP que contém os serviços para a descoberta do dispositivo na rede.

2.2.2.1 Camada de Rede (NWK)

A camada NWK *ZigBee* inclui mecanismos usados na conexão e desconexão de dispositivos numa rede, de aplicação de segurança aos frames e roteamento para seus destinos além de incluir a descoberta e a manutenção das rotas entre dispositivos envolvidos na rede. O coordenador da camada NWK é responsável por iniciar uma nova rede sempre que apropriado e assinalar endereços para os novos dispositivos associados.

Cada dispositivo mantém uma Tabela para todos os dispositivos da rede armazenarem suas informações de roteamento. Essa Tabela é chamada de Tabela de Roteamento (*Routing Table*). Cada movimentação que os dispositivos fazem dentro da rede, é armazenada nesta Tabela. A Tabela de roteamento do *ZigBee* possui três campos:

- Endereço destino com 2 bytes;
- Status (é o status do roteador e possui uma Tabela com alguns valores para saber qual dos valores corresponde ao status do momento) com 3 bits;
- Próximo salto com 2 bytes.

Para o processo de descobrimento de rotas, ele utiliza o mesmo processo utilizado no AODV e no DSR (Inundação).

A camada de rede (NWK) suporta três topologias: Estrela (*Star*), Árvore (*Tree*) e Malha (*Mesh*). A Figura 20 mostra a topologia em Estrela (Star):

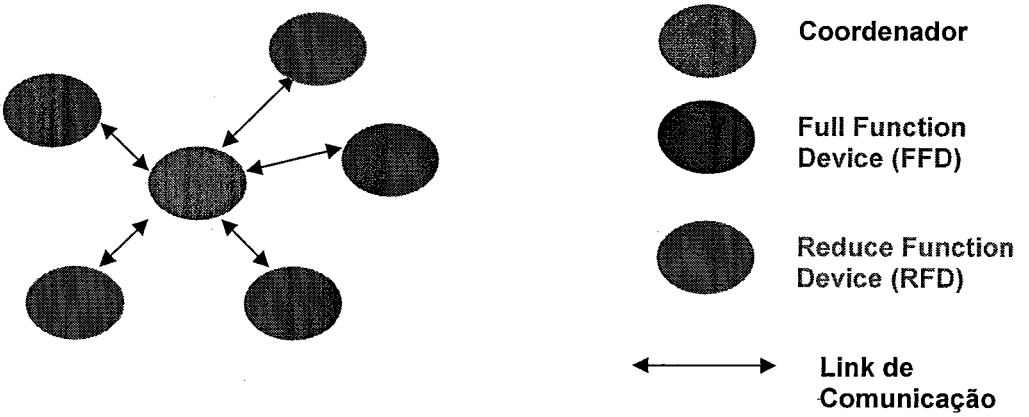


Figura 20 – Topologia de Rede em Estrela (*Star*)

Na topologia de rede *Star*, os dispositivos não se comunicam entre si apenas com o coordenador, todos são RFD e geralmente são todos *end-devices*. O único FFD desta topologia é o coordenador que se comunica com todos os dispositivos e todos se comunicam com ele.

A topologia em Arvore (*Tree*) é formada por alguns FFD e RFD conectados ao coordenador e a partir dos FFD surgem os RFD, como mostra a Figura 21. A vantagem desse tipo de topologia é que pode ser usada para estender o alcance geográfico da rede. Os FFD se comunicam entre si e com o coordenador e os RFD se comunicam apenas com o coordenador e com os FFD mais próximos a ele.

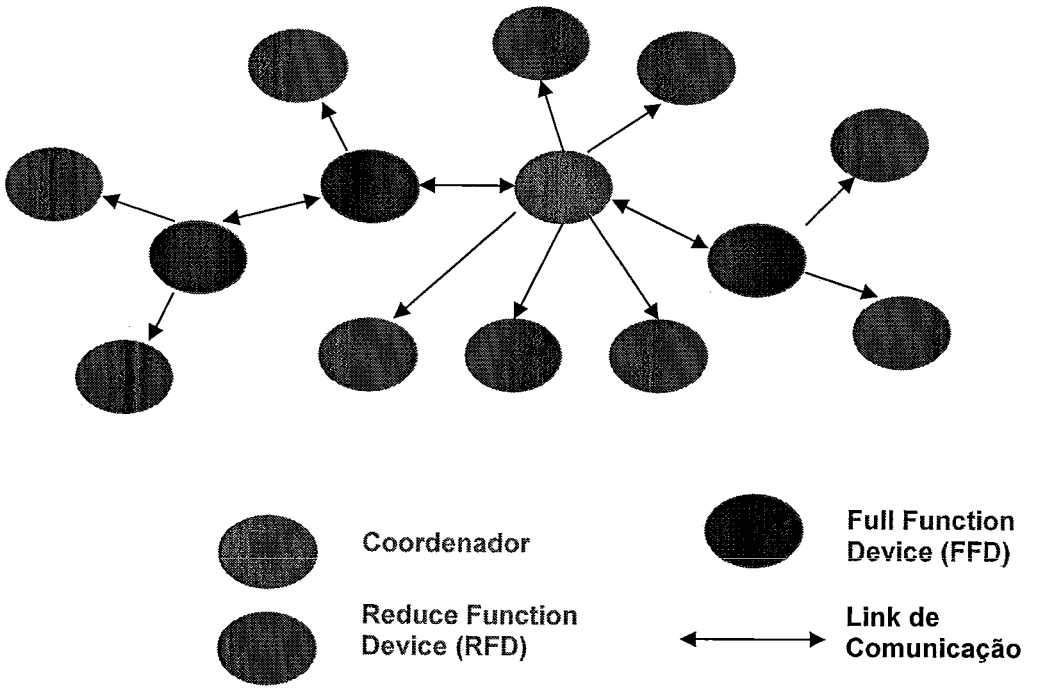


Figura 21 – Topologia em Árvore (*Tree*)

A topologia em malha, como pode ser vista na Figura 22, é a topologia em estrela (*star*) com alterações. Os RFD (*end-devices*) da topologia em estrela passam a ser FFD (roteadores) na topologia *Mesh* e estes possuem seus RFD (*end-devices*). A vantagem desta topologia consiste na confiabilidade e vazão da rede através do uso de múltiplos caminhos.

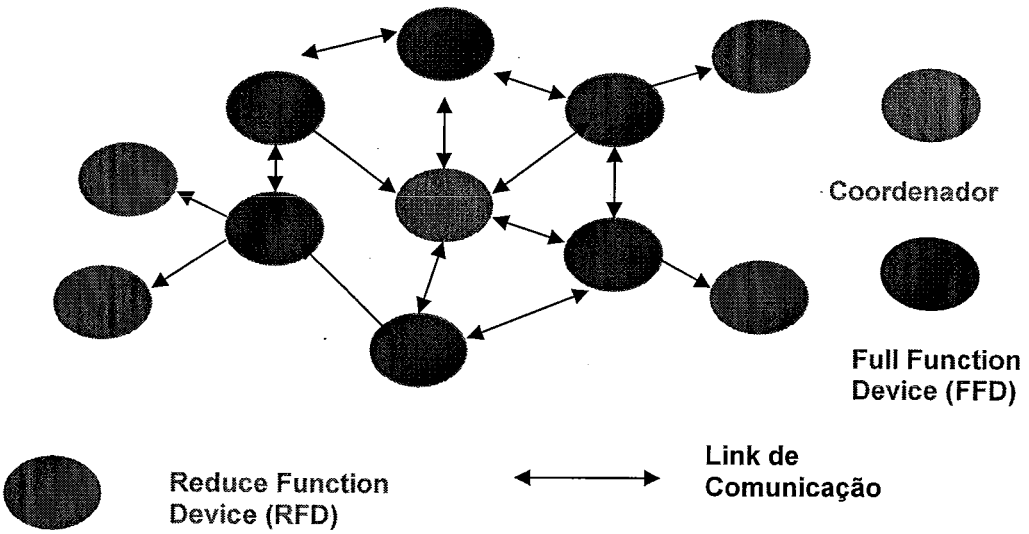


Figura 22 – Topologia em Mallha (*Mesh*)

2.2.2.1.1 Formas de endereçamento do *ZigBee*

O *ZigBee* tem duas formas de endereçamento do rede: *Unicast* e *Broadcast.*

O *Unicast* envia uma mensagem a um único nó, ou seja, somente o dispositivo destino receberá a mensagem. Já o *broadcast* é o envio de um nó a múltiplos nós, ou seja, qualquer radio que estiver na rede com o RX (recepção) habilitado, irá receber uma mensagem. Todos os dispositivos da rede são capazes de iniciar uma transmissão *broadcast* (de um para todos), mas essa comunicação é apenas entre dispositivos da mesma rede. O padrão *ZigBee* não suporta *broadcast* entre redes distintas.

2.2.2.1.2 Formação de uma rede *ZigBee*

Uma nova rede é estabelecida primeiramente pelo coordenador. O coordenador faz uma busca por alguns canais (na frequencia de 2.4Ghz ele possui 16 canais). Ao fim da busca, os canais escolhidos são colocados na ordem decrescente dos níveis de energia (os canais de níveis mais baixos são excluídos). Então o coordenador faz outra busca em cada canal procurando por dispositivos ou redes *ZigBee*. Com essa busca o coordenador faz a escolha do melhor canal para criar uma a rede. Após isso, o coordenador atribui a cada dispositivo novo que se conectar a rede, um endereço de rede (NWK) de 16 bits e um número lógico. Ao final, o coordenador libera a conexão para outros dispositivos ingressarem na rede. Este endereço de rede de 16 bits serve para eles serem localizados na rede, se houver mobilidade na rede e para se comunicarem uns com os outros e é enviado por broadcast. Esses endereços são atribuídos ou por um coordenador, usando um algoritmo de árvore estruturada como veremos na Figura 23. Iniciado o coordenador, este procurará por um outro coordenador que opere no mesmo canal, ou seja, na mesma freqüência.

Uma vez que uma rede nova é estabelecida, os roteadores e os end-devices estão aptos a juntar-se a rede. (Quando a rede é formada é possível que devido às mudanças físicas mais de uma rede apareça havendo então um conflito de coordenadores. Nesta situação, um coordenador pode resolver este conflito automaticamente por software e um dos coordenadores mudaria o seu canal. O coordenador afetado instruiria todos seus dispositivos para fazer as mudanças necessárias).

A Figura 23 mostra os níveis de uma árvore estruturada, ou seja, de como é formada uma rede *ZigBee*. Chama-se de dispositivo pai, o dispositivo que esta no nível zero (Profundidade =0). E chama-se de dispositivo filho, o dispositivo que esta abaixo deste e assim sucessivamente, ou seja, essa escala é feita em níveis.

O coordenador é sempre o nível zero da rede, ou seja, é sempre o nível mais alto da rede conhecido como *stack profile*. Na topologia de rede do SMRP, os roteadores estão no nível um e os *end-devices* no nível dois da rede. Ou seja, os *end-devices* como o próprio nome já diz estão no final da rede. Logo acima dele, está o roteador como sendo seu pai e acima do roteador, o coordenador como sendo seu pai ou então como mostra a Figura 23, acima do *end-device* temos um outro roteador como sendo seu pai também. Neste caso, o roteador pai terá obrigatoriamente que ser um FFD.

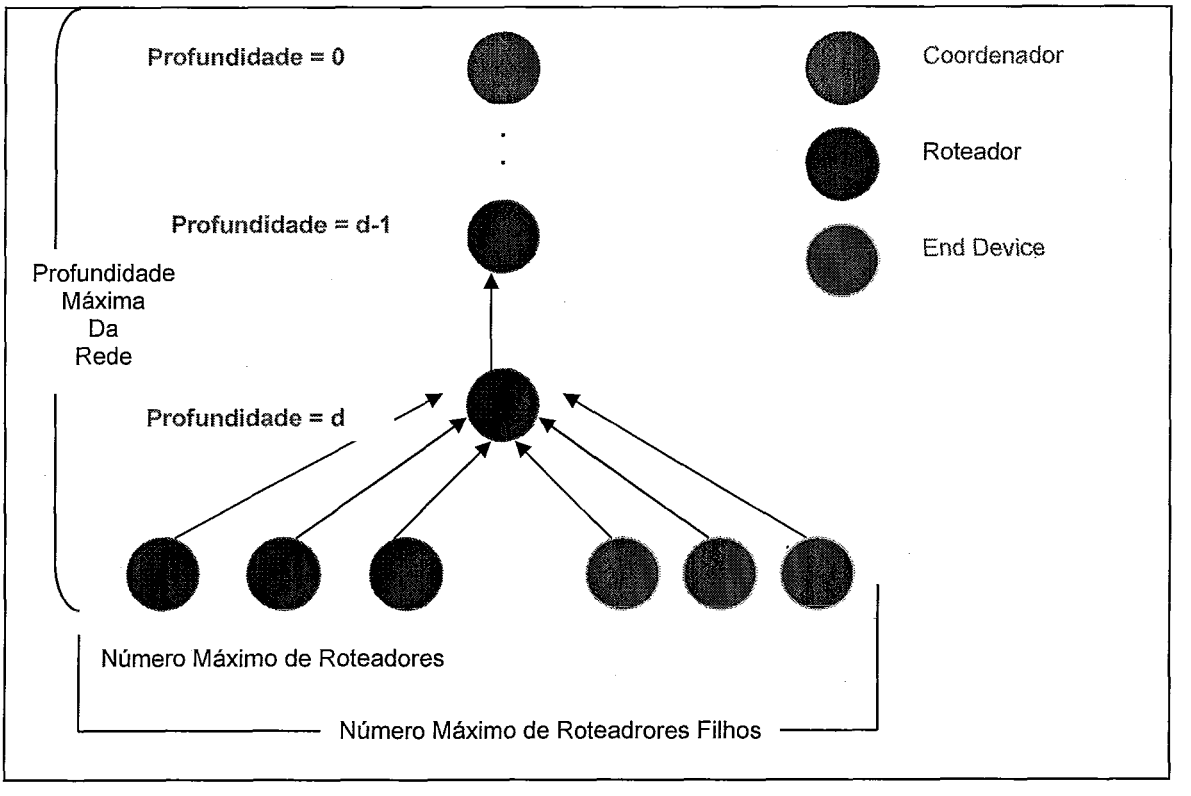


Figura 23 – Níveis de profundidade de uma rede ZigBee

O *stack profile* (o coordenador) é formado por uma série de parâmetros e informações, estas incluem definições de: profundidade máxima da rede (Lm), número máximo de filhos de um roteador, ou seja, número de roteadores possíveis em uma

profundidade (Cm), número máximo de roteadores filhos, ou seja, número máximo de *end-devices* (Rm). Estes parâmetros determinam o formato da rede. E através destes parâmetros é feito um cálculo para saber se a rede comporta ou não novos dispositivos.

$$Cskip(d) = \begin{cases} 1 + Cm.(Lm - d - 1), \\ \dfrac{1 + Cm - Rm - Cm.Rm^{Lm-d-1}}{1 - Rm} \end{cases}$$

Onde Rm =1

Se Cskip(d) > 0: O coordenador permite a entrada de novos dispositivos na rede e os atribue novos endereços;
Se Cskip(d) = 0: O coordenador não aceita novos dispositivos.

O coordenador calcula por software o Cskip(d) para cada dispositivo que entra na rede, ate que esta se complete. Estes atributos da árvore lógica podem ser usados para determinar o lugar de um dispositivo na rede. Isto simplifica a tarefa de roteamento. Como opção, a especificação *ZigBee* também permite que a atribuição de endereços seja realizada pela camada de aplicação.

2.2.2.1.3 Roteamento *ZigBee*

O *ZigBee* executa dois tipos de roteamentos distintos: roteamento hierárquico ou em árvore (*Tree Routing*) e roteamento *on-demand* (*Mesh Routing*) que é semelhante ao protocolo de roteamento AODV.

O tipo de protocolo que atua no roteamento hierárquico ou em árvore é chamado de protocolo híbrido, ou seja, parte do funcionamento ocorre como um protocolo pró-ativo e a outra parte como um protocolo reativo.

O roteamento *on-demand* só cria rotas quando estas são solicitadas pelo nó origem. Um dos algoritmos de roteamento *on-demand* mais conhecidos é o AODV. Este tipo de roteamento é também chamado de *ZigBee Mesh Routing*.

O próximo capítulo apresenta os protocolos de roteamento para redes ad-hoc.

Capítulo 3

Protocolos de roteamento para redes Ad-hoc

3.1 Protocolos de roteamento

No contexto dos protocolos de roteamento, algumas blibliografias se referem a eles como "algoritmos de roteamento" outras como "protocolos de roteamento". O fato é que existe uma diferença entre eles. Os protocolos de roteamento são responsáveis por encontrar, estabelecer, atualizar as Tabelas de roteamento e divulgar rotas entre dois nós que desejam se comunicar, eles implementam um ou mais algoritmos de roteamento.
Para exemplificar os algoritmos de roteamento, temos: *Distance Vector, Link State, Flooding, Shortest Path First* (SFP) etc. Já os protocolos de roteamento: RIP, OSPF e BGP para redes fixas e AODV, DSR, DSDV para redes móveis. Os protocolos voltados para redes fixas são ineficientes para redes móveis [28]. Neste trabalho será utilizada a nomenclatura Protocolos de Roteamento.

Em uma rede *ad-hoc*, se a comunicação direta entre dois nós não for possível, informações de roteamento deverão ser utilizadas para que se possa descobrir o melhor caminho possível entre o nó origem e o nó destino. Neste tipo de rede todos os nós atuam como roteadores não havendo nenhum nó especial responsável exclusivamente pelo roteamento. Cada nó utilizando algoritmos próprios troca informações e decide se um determinado destino é alcançável. Havendo mais de um caminho, decide-se pelo melhor de acordo com uma métrica pré-estabelecida.

3.2 Classificação dos protocolos de roteamento

Os protocolos de roteamento em redes *ad-hoc* são classificados em três categorias: *On-demand* ou reativo, *Table-driven* ou pró-ativo e híbrido.

Protocolos *On-Demand* ou Reativo: Estes protocolos não ficam continuamente enviando informações da topologia da rede e não ficam atualizando suas Tabelas de roteamento á todo momento. Uma rota para um determinado destino só é descoberta quando for necessário enviar um pacote para este. Dessa forma, só há gasto de energia quando a descoberta de rotas for necessária.

A dinâmica da topologia das redes *ad-hoc* é outro fator que contribui para a maior eficiência dos protocolos reativos em relação aos pró-ativos. Em uma rede com intensa movimentação não é válido descobrir uma rota que não será utilizada em instantes, já que esta rota possui grandes chances de já ter sido modificada no momento do seu uso. A desvantagem dos protocolos reativos é o maior atraso no envio de pacotes, já que se a rota do destino do pacote não for conhecida, o procedimento de descoberta de rota deve ser realizado.

Faz parte desse grupo os protocolos: *Ad Hoc On-Demand Distance Vector Routing* (AODV), *Dynamic Source Routing* (DSR), *Lightweight Mobile Routing* (LMR), *Temporally Ordered Routing Algorithm* (TORA), *Associativity-Based Routing* (ABR) e *Signal Stability Routing* (SSR). Atualmente o AODV e o DSR são os mais eficientes e os mais citados na literatura.

Protocolos Table Driven ou Pró-Ativo: Nos protocolos pró-ativos cada nó possui á todo momento informações em sua Tabela de roteamento referentes a todos os possíveis destinos como ocorre nas redes cabeadas. Dessa forma, quando um pacote deve ser enviado a um determinado nó de destino a rota já é conhecida e pode ser imediatamente utilizada.

A vantagem de utilizar um protocolo pró-ativo é ter uma Tabela de roteamento constantemente atualizada tendo assim a rota disponível a qualquer momento. A grande desvantagem dessa classe de protocolos é o custo para manter as Tabelas atualizadas devido à troca de mensagens de controle que ocupam parte da capacidade de transmissão das redes. Apesar destes protocolos de roteamento terem sido projetados para o mesmo tipo de rede, as suas características são bem distintas.

Nesta classificação estão incluídos os protocolos: *Destination-Sequenced Distance-Vector Routing* (DSDV), *Wireless Routing Protocol* (WRP) e *Clusterhead Gateway Switch Routing (*CGSR).

Protocolos Híbridos: Combinam as características dos protocolos pró-ativos e reativos. Os protocolos desse tipo estabelecem uma zona onde ele vai atuar como pró-ativo, a partir do limite dessa zona ele passa a atuar como reativo, fazendo um *flood* (inundação

de uma grande quantidade de pacotes de atualização para descobrir qual rota utilizar para enviar a informação). Esses protocolos são adequados para redes *Ad-hoc* com muitos nós para poder estabelecer uma zona onde se tem um conhecimento parcial da topologia da rede e caso necessite enviar alguma informação para um nó mais distante este protocolo atuaria como um protocolo *on- demand.* Nesta classificação está incluído o protocolo: *Zone Routing Protocol* (ZRP).

3.3 AODV (Ad hoc On-demand Distance Vector)

O AODV foi projetado para o uso em redes *ad-hoc* que possuam desde dezenas até milhares de nós móveis. Supõe-se que todos estes nós confiam uns nos outros.

O objetivo principal do protocolo é se adaptar rápida e dinamicamente às variações das condições dos enlaces da rede, descobrindo rotas de forma a se evitar o desperdício de banda e minimizar o uso de memória e processamento nos nós que atuam como roteadores. Cada nó, troca informações e decide se um determinado destino é alcançável. Havendo mais de um caminho, já que o protocolo AODV é utilizado em uma situação em que se tenham múltiplos caminhos, decide-se pelo melhor de acordo com uma métrica pré-estabelecida.

3.3.1 Mensagens de roteamento

- *Route Request* - RREQ (Requisição de rota)
- *Route Reply* - RREP (Resposta á requisição de rota)
    - Caminho inverso
- *Route Error* - RERR (Aviso de queda de enlace)

3.3.2 Funcionamento do AODV

Quando o nó fonte não sabe o caminho (rota) do nó destino, mas ele sabe quem é o nó destino, é iniciado o processo de descoberta de rota. Neste processo, o nó fonte envia uma mensagem de RREQ para todos os seus nós vizinhos (*broadcast*), ou seja, o nó fonte envia para todas as rotas que tiverem na rede e que tiverem contato com o nó fonte.

Cada nó possui dois contadores: um número de seqüência e um identificador de *broadcast*. Se um nó fonte precisa se comunicar com um nó destino para o qual ele não possui entrada em sua Tabela de roteamento, ele incrementa o seu número de seqüência e o seu identificador de *broadcast* e envia mensagens RREQ por *broadcast* para toda a rede. Cada mensagem RREQ contém: o endereço da fonte e do destino, o número de seqüência da fonte, o último número de seqüência do destino conhecido pela fonte, o identificador de *broadcast* e o contador de saltos, que inicialmente vale zero. É importante ressaltar que uma mensagem RREQ só pode ser identificada através do endereço da fonte e do identificador de *broadcast*, que é incrementado a cada nova requisição de rota enviada.

Conforme os nós vizinhos forem recebendo seus RREQ, eles vão enviando para o nó que lhes enviou mensagens unicast de RREP fazendo assim o caminho inverso.

As mensagens RREP contém: o endereço da fonte e do destino, o número de seqüência do destino, o contador de saltos, incrementado em uma unidade a cada salto e o tempo de vida para a rota, isto é, o tempo em que se considera a rota ativa. Se os nós não receberem a confirmação dos vizinhos (mensagens de RREP) é porque houve um RERR. O nó que receber um RERR deve informar aos seus nós vizinhos (*broadcast* ou *unicast*).

Quando o pacote chegar ao seu destino através dos RREQ enviados, estabelece-se então um caminho (rota) do nó fonte ao nó destino. Se tiverem vários caminhos (rotas), permanece a rota que tiver mais proximidade entre o nó fonte e o nó destino.

Quando o nó fonte recebe a notificação de queda de um enlace (RERR), ele decidirá se reinicia ou não um novo processo de descobrimento de rota. O AODV informa a todos os nós que usam a rota que ocorreu a falha, tornando-a inválida. A informação da queda de enlace é propagada a todos os demais nós que façam parte deste enlace em alguma rota ativa. Uma rota ativa é uma entrada "recente o suficiente" na Tabela de roteamento com uma métrica finita. Somente rotas ativas podem ser usadas para o envio de pacotes. Na Tabela de roteamento, existem também entradas com rotas não ativas, isto é, rotas inválidas, que serão apagadas após um *timeout*.

O protocolo oferece duas maneiras para se detectar a queda de um enlace. A primeira utiliza um mecanismo de detecção de vizinhança, que pode ser implementado pelo próprio AODV. Neste mecanismo, mensagens de *broadcast* local chamadas de *hello* são enviadas periodicamente para se confirmar á conectividade local entre vizinhos. O não recebimento de uma mensagem de *hello* de um vizinho durante um período de tempo indica a queda de um enlace. A segunda maneira para se detectar a queda de um enlace é através de informações provenientes da camada de acesso ao meio, o que acarreta em economia de banda. No caso do IEEE 802.11, podem utilizar os pacotes de reconhecimento positivo (*Acknowledgement - ACK*) da camada de enlace. Se um nó tenta enviar um determinado número de vezes um pacote a outro nó e não recebe em seguida a confirmação através de uma mensagem ACK, pode-se concluir que o enlace caiu.

O funcionamento é melhor visuzalizado na Figura 24 [29]:

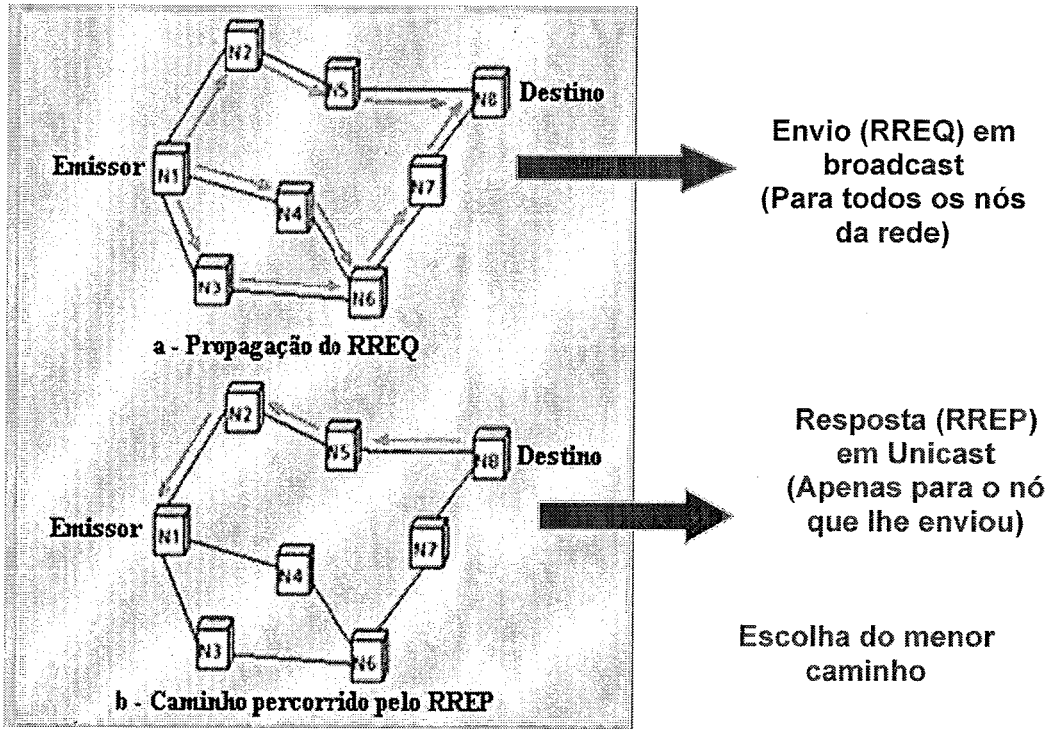


Figura 24 - Descoberta de Rota do protocolo AODV

3.3.3 Tabela de Roteamento

Cada nó constrói a sua Tabela de roteamento a partir da troca das mensagens RREQ, RREP e RERR. Esta Tabela contém os seguintes campos: endereço IP do destino, número de seqüência do destino, indicador de validade do número de seqüência do destino, indicador de validade de uma rota (válida, inválida, reparável, sendo reparada), interface de rede, contador de saltos (número de saltos necessários para alcançar o destino), próximo salto, lista de predecessores, tempo de vida da rota (tempo de expiração da rota). Cada entrada da Tabela de roteamento de um nó tem de incluir a última informação disponível sobre o número de seqüência (o protocolo utiliza números de seqüência crescentes para mostrar o quão recente é uma rota) de 32 bits associado ao endereço IP do nó de destino, para o qual a entrada da Tabela é mantida. O número de seqüência do destino, como é chamado, é atualizado sempre que um nó recebe uma nova informação sobre este a partir de uma mensagem de roteamento relacionada ao destino. O algoritmo do AODV depende que cada nó da rede mantenha o seu número de seqüência do destino para garantir a ausência de *loops* em todas as rotas que utilizam este nó. Todas as mensagens de roteamento contêm números de seqüência [30].

Um nó incrementa o seu número de seqüência em duas situações: imediatamente antes de iniciar um processo de descobrimento de rota, o que previne conflitos com caminhos reversos previamente estabelecidos e imediatamente antes de enviar um RREP em resposta a um RREQ.

Para assegurar que uma informação relativa a um destino não está desatualizada, o nó compara o número de seqüência que ele possui, em uma entrada da Tabela relativa ao destino com o número presente na mensagem que ele acaba de receber. Subtrai-se o número armazenado do número recebido na mensagem. Se o resultado for menor do que zero, a mensagem é descartada.

A outra situação em que um nó pode mudar o número de seqüência do destino, em uma de suas entradas da Tabela de roteamento é em resposta a quebra ou a expiração do tempo de vida do enlace utilizado como um próximo salto para o destino. Quando isto ocorre, para cada destino que necessita do enlace afetado para ser alcançado o nó incrementa o número de seqüência e marca a rota como inválida. Sempre que a mensagem mais recente chegar a um nó que possui uma rota marcada como

inválida para este destino, o nó deve atualizar a sua Tabela de roteamento de acordo com a informação contida na mensagem.

3.4 DSR (Dynamic Source Routing)

3.4.1 Funcionamento do DSR

A característica principal do DSR é a utilização do roteamento pela fonte, ou seja, a fonte conhece a rota completa, salto a salto, para o destino e os cabeçalhos dos pacotes de dados carregam esta seqüência de nós. Estas rotas são armazenadas em um *cachê* de rotas onde ficam armazenadas as últimas rotas.

Quando uma estação necessita de uma rota para um destino, ela faz uma consulta nas informações em *cachê*. O DSR permite que cada nó mantenha múltiplas rotas para o mesmo destino. O DSR consiste de dois mecanismos: descoberta de rotas e manutenção de rotas.

Quando um nó precisa enviar um pacote para outro nó, o nó de origem verifica se possui uma rota para o nó de destino em seu *cachê*. Caso a rota não exista, inicia um processo de descoberta de rotas para encontrar dinamicamente uma rota para o destino. Quando esta estação recebe o pacote, se ela não for o destinatário do pacote, ela simplesmente retransmite para a próxima estação identificada. É importante citar que cada entrada em (*cachê*) esta associada a um período de expiração, tal que o seu término implica na remoção da rota.

Destaca-se que todos os nós ao receber uma mensagem operam, muitas vezes, de maneira promíscua (escutam o meio) e lêem informações do pacote para atualizarem seu *cachê de rotas* com informações recentes. Uma das vantagens da utilização deste algoritmo consiste em minimizar o *overhead. Overhead* é geralmente considerado qualquer processamento ou armazenamento em excesso, seja de tempo de computação, de memória, de largura de banda ou qualquer outro recurso que seja requerido para ser utilizado ou gasto para executar uma determinada tarefa. Como consequência pode piorar o desempenho do aparelho que sofreu o *overhead.*

3.4.2 Mensagens de Roteamento

- *Route Request* - RREQ: Requisição de rota. Contém o *Route Record* e o *Request ID*;
- *Route Reply* - RREP: Resposta á requisição de rota
- *Route error* - RERR: Queda de enlace

3.4.3 Protocolo de descobrimento de rotas

O protocolo de descobrimento de rotas no DSR é similar ao utilizado pelo AODV, entretanto o DSR não depende de nenhuma atividade periódica ou baseada em temporizadores. Ele utiliza ao máximo o *cachê* de rotas disponível em cada nó e mantém múltiplas rotas para um determinado destino. Este protocolo permite que um nó descubra dinamicamente uma rota para qualquer outro nó em uma rede *Ad hoc*, mesmo que não esteja no seu raio de alcance, pois este mecanismo permite o uso de caminhos formados por nós intermediários.

Um nó inicia o processo de descobrimento de rotas enviando por *broadcast* (difusão) um *route request* que será recebido por todos os nós vizinhos. O *route request* identifica o remetente e o destinatário da descoberta de rota. Desta forma, quando o remetente alcança o destinatário, ele inicia o procedimento de *route reply*. Cada *route request* possui um *route record*, no qual registra todos os nós por onde o pacote passou durante o período de descoberta de rotas e também contém um *request id* único, caracterizado por um número em sequência. Para detectar o recebimento de requisições de rotas duplicadas, os nós atualizam suas listas a cada requisição de rota recebida.

3.4.4 Procedimento de Manutenção de Rotas

O procedimento de manutenção de rotas monitora todas as rotas a partir dos nós que fazem parte da rede. Portanto, quando um nó repassa um pacote ele aguarda um reconhecimento (*Acknowledge*) do próximo nó registrado na rota. Na ocorrência da não confirmação do reconhecimento ou caso haja alteração na topologia da rede que possa comprometer o uso das rotas, o nó envia para o remetente um pacote de erro de rota, conhecido como *route error.*

Neste pacote está contido o endereço da estação que identificou a falha na rota e o endereço da estação que não confirmou o reconhecimento. Assim, o remetente remove todas as rotas que ele utilizava neste enlace.

3.5. Comparação entre o AODV e o DSR:

A Tabela 1 mostra uma comparação dos dois protocolos citados anteriormente:

AODV	Tabela de Roteamento	Nós intermediários tem iniciativa de se comunicarem entre si	Somente rotas ativas podem ser usadas para o envio de pacotes	Rotas inválidas são apagadas por timeout (temporizador)	Possuem número de seqüência (atualização)	Mensagens *Hello*
DSR	*Cachê* de Rotas	Nós intermediários não tem a iniciativa de se comunicarem entre si	Podem enviar pacotes com rotas antigas	Não tem temporizador	Não possuem número de sequencia	Roteamento pela fonte

Tabela 1 – Comparação AODV e DSR

O próximo capítulo apresenta a modelagem e o mapeamento do sistema alvo, o simulador utilizado, a metodologia experimental, as estratégias apresentadas, os gráficos e os resultados obtidos.

Capítulo 4

Modelando e Simulando o Sistema Alvo

4.1 Simulação

A realização de estudos práticos visando a avaliação dos protocolos de roteamento para redes *Ad-hoc* e o estudo do *ZigBee* baseados em uma rede real, apresentam dificuldades de grande complexidade e alto custo. Por isso, serão utilizadas as técnicas de simulação. Estas são utilizadas com muita freqüência principalmente pela possibilidade de testar cenários variados incluindo o comportamento dos protocolos, as novas tecnologias e o efeito de diferentes topologias de rede.

A simulação é utilizada em geral quando o sistema a ser avaliado não está disponível, antes de colocar a rede baseada em *ZigBee* em funcionamento no local onde está instalado o SMRP. Se a performance corresponder ás expectativas, será realizada a instalação no local. As simulações foram executadas em uma máquina Pentium 4, CPU 2.20Ghz, 256 Mb de memória RAM, executando o sistema operacional *Linux Slackware*.

4.2 Ferramentas utilizadas para a Simulação

As ferramentas utilizadas para a simulação se destacam tanto pela sua importância no atual contexto de redes como pela abrangência, flexibilidade, quantidade de referências e grau de aceitação acadêmica. Na Tabela 2, são descritas as características mais marcantes de algumas ferramentas de simulação disponíveis.

Ferramenta	Código Aberto	Linguagem de Programação	Sistema Operacional	Padrões Suportados
NS-2	sim	C++/Otcl	Linux / Windows	AODV, DSR,TORA,DSDV, 802.15.4, 802..11
Glomosim	sim	C e Parsec	Linux / Windows	802.15.4
OPNET	sim	Nesc, C, C++	Linux / Windows	802.11, 802.16, CSMA/CA e GTS com beacon
Omnet	sim	Nesc, C, C++	Linux / Windows	Redes ad-hoc, TCP/IP, 802.11, IPV6

Tabela 2 – Ferramentas de Simulação

Na escolha da ferramenta de simulação mais adequada aos objetivos deste trabalho foi levado em consideração alguns fatores como:

1. Código da ferramenta: código-aberto, *software* livre ou não;
2. Linguagem de programação: linguagens de programação utilizadas pela ferramenta;
3. Sistemas operacionais usados: *UNIX, Sun Solaris, IBM AIX, FreeBSD, GNU/Linux, MS-Windows* etc.
4. Padrões suportados: refere-se à topologia e aos componentes da rede;

Foi escolhido para este trabalho o simulador de redes [31] *Network Simulator* (NS-2) por ele ser gratuito e ter código fonte aberto, utilizar a linguagem de programação em script Otcl, utilizar tanto o *Windows* quanto o *Linux* e ser compatível com os protocolos de roteamento AODV e DSR.

4.3 *Network Simulator* (NS-2)

O *Network Simulator* (NS-2) é um simulador de redes baseado em eventos discretos direcionado para pesquisa em redes de computadores. Fornece um suporte amplo para avaliação de desempenho de funções como *Unicast* ou *Multicast* e todos os principais protocolos *IP, TCP, UDP, RTP, RTCP, FTP, HTTP* em taxas de tráfego constantes *(Constant Bit Rate* (CBR)). O simulador funciona tanto para redes fixas como para redes *wireless* em geral. Esta plataforma é considerada como um padrão de fato e sua validação é comprovada e aceita por órgãos de padronização como o *National*

Institute of Standards and Technology (NIST) [32] e agências de governo como a *Defense Advanced Research Projects Agency* (DARPA).

Sua primeira versão surgiu em 1989. Esta versão foi oriunda do *software* REAL *Network Simulator*. A partir de 1995 este *software* passou a ter o apoio do DARPA resultante de um projeto conhecido como *Virtual InterNetwork Testbed* (VINT) e hoje está sob a licença de software livre (GPL) [33]. O seu desenvolvimento atualmente continua pleno e é realizado por diversos pesquisadores ao redor do mundo. Além da DARPA, compõem esse projeto a USC/ISI, Xerox PARC, LBNL e a Universidade de *Berkeley*.

Network Simulator se encontra atualmente em sua versão 2, por este motivo ele é conhecido como NS-2. É um *software* desenvolvido na Universidade de *Berkeley* e escrito em C++ que utiliza um interpretador OTcl (Tcl-*Tool Command Language)* voltado para objetos como linguagem de modelagem dos cenários. A linguagem Tcl (*Tool Command Language*) foi desenvolvida por *John Ousterhout* no final da década de 1980. Seu objetivo era desenvolver um bom interpretador e uma biblioteca que pudessem ser reutilizados de diversas maneiras. Com isso ele teria: linguagens simples, genéricas e extensíveis. Algum tempo depois o criador da Tcl desenvolveu a biblioteca gráfica Tk como forma de prover um conjunto de recursos gráficos e reutilizáveis. A integração entre Tcl e Tk se tornou tão popular que ambas passaram a ser distribuídas e referenciadas a um mesmo pacote denominado Tcl/Tk [34]. O módulo OTcl, desenvolvido no *Massachusetts Institute of Technology* (MIT) é uma extensão das linguagens Tcl/Tk para a programação orientada a objetos. O simulador suporta uma hierarquia de classes em C++ e uma hierarquia similar no interpretador [35].

Uma grande vantagem do NS-2 reside no fato de ele ser totalmente gratuito e com código fonte aberto o que permite ao usuário proceder os ajustes que julgar necessários. Devido a esse fato, o NS-2 inclui contribuições substanciais de outros pesquisadores incluindo códigos de wireless para todos os padrões, roteamento etc [36].

O NS-2 possui duas formas para analisar os resultados da simulação: a primeira é o *trace file* que contém toda a comunicação realizada entre os nós. A segunda é um visualizador gráfico para animações (*Network Animator* - NAM*)* onde é visualizada a topologia de rede que foi programada no script em Otcl. Essas duas ferramentas de

análise serão detalhadas mais a frente. O NS-2 inclui também uma ferramenta de plotagem, o *xgraph* e vários tipos de geradores de tráfego como: *Constant Bit Rate* (CBR) utilizado para simular tráfego de voz; *File Transfer Protocol* (FTP) para gerar tráfego correspondente a aplicações de transferência de arquivos [37].

Nesse trabalho foi utilizada a versão NS 2.31 (versão mais atual) rodando na distribuição Linux. Para o padrão 802.15.4/*ZigBee* funcionar corretamente é necessário utilizar um módulo MAC. Foi necessário rodar esse módulo dentro do NS-2, sendo este gratuito [38].

Para utilizar o NS-2 no *Windows* é necessária a instalação de um emulador chamado *Cygwin* [39]. No Anexo B deste trabalho encontram-se maiores informações sobre o *Cygwin*, como instalá-lo e sites para *downloads*.

4.3.1 Ferramentas de Análises do NS-2

4.3.1.1 Trace File

Uma simulação baseada em *traces* é a que tem como entrada um registro que contém eventos ordenados no tempo observados em um sistema real. Esses registros são chamados de *trace file*. Uma característica importante é a credibilidade. Um *trace* tem uma maior credibilidade do que informações geradas randomicamente através de alguma distribuição. Um dos principais problemas dos *traces* é o tamanho. Os *traces* são geralmente seqüências longas e exigem um considerável tempo computacional para serem processados. Processada a simulação e a geração do NAM, o NS-2 processa automaticamente um *trace file*, é um arquivo com a terminação ".tr". A partir daí, inicia-se uma das fases mais importantes: a análise dos resultados. Afinal, estes dados serão utilizados na elaboração dos gráficos que servirão de suporte a este trabalho.

O formato wireless do *trace file* [40] é o mesmo para os dois protocolos de roteamento (AODV e DSR). Algumas linhas deste formato são mostradas na Figura 25.

```
s 20.553760000 _15_ MAC  --- 0 CM7 8 [0 ffffffff f 0]
s 20.574912000 _0_  MAC  --- 0 BCN 12 [0 ffffffff 0 0]
s 20.578030123 _5_  RTR  --- 3 DSR 32 [0 0 0 0] ------- [5:255 6:255 32 0] 1 [1 2] [0 2 0
0->16]
s 20.580295123 _5_  MAC  --- 3 DSR 39 [0 ffffffff 5 800] ------- [5:255 6:255 32 0] 1 [1 2]
s 20.580800000 _12_ MAC  --- 0 CM7 8 [0 ffffffff c 0]
s 20.611040000 _9_  MAC  --- 0 CM7 8 [0 ffffffff 9 0]
s 20.611040000 _5_  MAC  --- 0 CM7 8 [0 ffffffff 5 0]
r 21.500000000 _5_  RTR  --- 135 CBR 80 [0 0 0 0] ------- [5:0 6:0 32 0] [5] 0 0
```

Figura 25 – Formato do *Trace File*

- O primeiro campo da Figura 25 diz respeito ao evento ocorrido:
- Pode ser um descarte de pacote (D) onde D é *dropped*, ou seja, pacotes perdidos; uma transmissão de pacote (s) onde s é *send* de pacotes enviados ou um recebimento de pacote (r) onde r é *receive* de pacotes recebidos;
- O campo seguinte corresponde ao tempo da simulação onde o evento ocorreu, no caso da Figura 25, a primeira linha tem como tempo 20.553760000s;
- O terceiro campo é referente ao disposivito (nó) onde o evento ocorreu, na Figura 25, na primeira linha ocorreu um envio de pacote de dados, no tempo de 20.553s no nó 15.
- O campo consecutivo diz respeito a um evento de nível de rede, podem ser: colisões MAC, RTR etc.
- Os três traçinhos (---) correspondem a uma série de *flags* relacionados a notificação antecipada de congestionamento, mas normalmente não são utilizados.
- O número seguinte aos flags é a identificação do fluxo de cada aplicação.
- O próximo campo diz respeito ao tipo dos pacotes (CM7, ACK), tipos de tráfego (CBR) e aos protocolos de roteamento (no caso o DSR);
- Os números seguintes aos protocolos de roteamento referem-se ao tamanho do pacote (em bytes);
- Os próximos campos em hexadecimal em negrito na segunda linha da Figura 25 correspondem respectivamente ao: tempo para o envio de dados [0 0 0 0], destino do endereço MAC [5:255 6:255 32 0] , fonte do endereço MAC [1 2] e o tipo (se é ARP,IP) [0 2 0 0->16].

Para este trabalho, dos *traces files* analisados foram tirados alguns valores para serem analisados graficamente. Por exemplo, no script "Pacotes perdidos"; a linha 0 do *trace file* corresponde ao tempo 27.2s, a linha 50 corresponde ao tempo 36.9s, a linha 100 corresponde a 46.0s e assim sucessivamente até completar todas as linhas inteiras que o *trace file* "pacotes perdidos" possue. Para isso foi necessário separar os *traces* em três grupos: pacotes perdidos (as linhas que possuem um D de *dropped* na frente), pacotes recebidos (as linhas que possuem um r de *receive* na frente) e pacotes enviados (as linhas que possuem um s de *send* na frente) isso para ambos os protocolos de roteamento. Os arquivos de *trace* foram separados e visualizados num editor de arquivos chamado *Crimson Editor* [41]. Esses scripts estão disponíveis no anexo A.

4.3.1.2 Network Animator (NAM)

O NAM [42] é uma ferramenta utilizada para visualizar as simulações criadas no *Network Simulator*. A primeira etapa para utilizar o *NAM* é produzir o arquivo de traço onde o usuário cria/desenha o seu próprio estudo de caso como mostra a Figura 26.

O arquivo de traço contém a informação da topologia, como por exemplo: estações, ligações, assim como traços do pacote. Para iniciá-lo basta digitar "nam" na tela do Linux, segundos depois aparecerá a tela do NAM. Nesta tela, clica-se em *File → NAM Editor*. Aparece então a tela da direita, uma tela lisa sem nenhuma ilustração onde poderemos criar/desenhar a topologia de rede utilizando a barra de ferramentas da tela, como mostra a Figura 26.

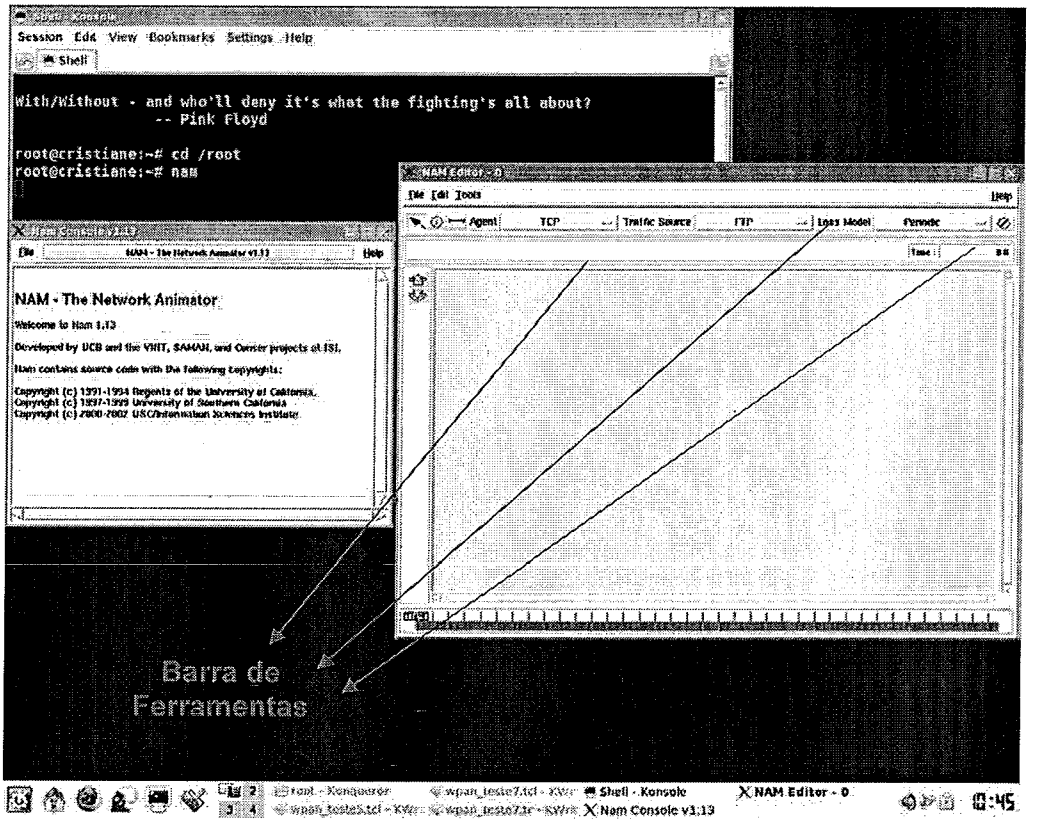


Figura 26 - Tela do NAM onde é criado o arquivo de traço

O arquivo de traço como mostra na Figura 27 [43], é gerado pelo próprio simulador e quando encerra a simulação este arquivo está pronto para a animação pelo *NAM*.

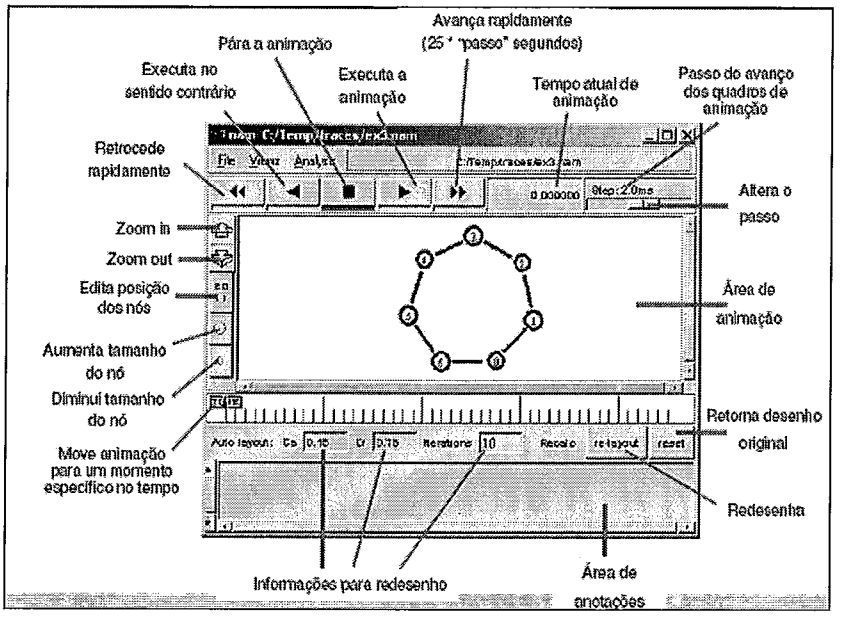


Figura 27 – Tela do NAM com o arquivo de traço pronto

A simulação pode ser escrita em qualquer editor de texto. Os arquivos devem ser salvos no computador com a extensão "*.tcl*". Para se executar a simulação faz-se necessária a digitação do comando: $ ns <nome-do-arquivo>.tcl

A Figura 28 mostra a simulação sendo feita, ou seja, mostrará o NS-2 escaneando todos os nós nos devidos tempos e fazendo as comunicações necessárias de acordo com o que foi programado no script "arquivo.tcl". Assim que a simulação chegar no tempo de 99.999680s como mostrado na Figura 28 é porque a simulação chegou ao fim, pois foi programado no script "arquivo.tcl" (vide Anexo A.2) que ele parasse de simular em 100s, a partir daí surge a tela de inicialização do NAM e logo depois o aparecimento da topologia de rede.

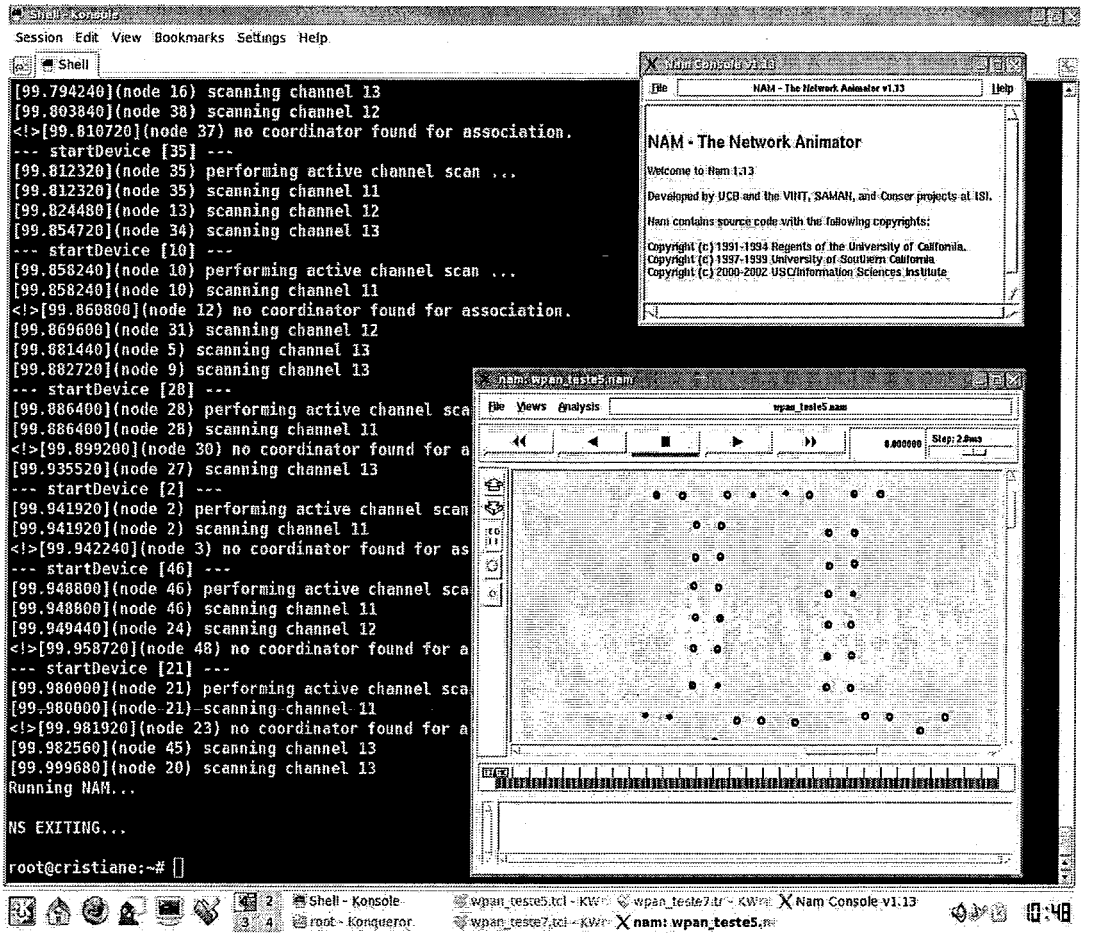


Figura 28 – Tela do NS-2, do NAM e da topologia de rede

4.4 Mapeamento do Sistema Alvo no NS-2

Mapeou-se todas as informações de uma rede *ZigBee* juntamente com as informações do SRMP, utilizando ambos os protocolos de roteamento. Assim foi programado em Otcl e simulados no NS-2. Para que esse mapeamento fosse necessário, utilizou-se um modelo detalhado de simulação visando demonstrar o desempenho dos dois protocolos.

Como os nós (roteadores) estão sempre se comunicando e recebendo informações da UMC e esta da concessionária, para visualizar melhor o funcionamento do NS-2 e o funcionamento da topologia de rede no NAM, foram programadas propositalmente duas transmissões de dados: uma do lado direito da topologia de rede que é efetuada em 20.5s, entre a UMC e a UM6 (consumidor 28) e a outra do lado esquerdo da topologia de rede que é efetuada em 20.8s, entre a UMC e a UM15 (consumidor 43).

- **Topologia de Rede:** O SMRP se assemelha com a topologia de rede *Mesh*, pois os coordenadores são dispositivos FFD (*Full Function Device*) e se comunicam diretamente com os roteadores que também são dispositivos FFD e estes se comunicam com os end-devices que são dispositivos RFD (*Reduce Function Device*). As topologias em *Tree* e *Mesh* são bem parecidas, porém o que as diferencia é que na topologia em *Tree* os coordenadores se comunicam diretamente com os *end-devices*, o que não é o caso deste trabalho.

- **Dispositivos de Rede:** No SMRP, a UMC é a concentradora da rede, as UM os dispositivos que roteam as mensagens e os consumidores são os dispositivos finais, donde se conclui que: a UMC é o coordenador (nó zero), as UM são os roteadores e os consumidores são os end-devices;

- **Formação da rede:** Como na topologia de rede, há apenas um coordenador (uma única UMC) e como a topologia de rede a priori já está completa (a não ser que haja uma mudança na geografia do local a qual o projeto está instalado) não será necessária á formação de novas redes *ZigBee* nem a entrada de novos dispositivos nessa rede;

- **Profundidade da Rede:** O coordenador (UMC) é o *stack profile* permanecendo no nível zero, os roteadores (UM) no nível um e os *end-devices* (consumidores) no nível dois de acordo com os níveis de profundidade da rede;

- **Cálculo para saber se a rede comporta ou não mais dispositivos:** Não foi calculado o valor de Cskip(d), pois a topologia de rede já está completa e a priori não entrarão mais dispositivos;

- **Formas de endereçamento:**
  - *Unicast:* Somente um dispositivo da rede é capaz de iniciar uma transmissão *unicast*, somente um destino receberá a mensagem. Essa comunicação é feita de UM para UM ou então de UMC para a UM destinatária ou ainda de UM para o seu consumidor.
  - *Broadcast*: Todos os dispositivos da rede (UMC, UM) são capazes de iniciar uma transmissão broadcast (de um para todos), mas essa comunicação é apenas entre dispositivos da mesma rede. O padrão *ZigBee* não suporta *broadcast* entre redes distintas (o que não é o caso deste trabalho);

- **Transmissão de Dados:** Duas formas de transmissão de dados são usadas, utilizando a técnica CSMA/CA:
  - Do coordenador (UMC) para os dispositivos (UM);
  - Dos dispositivos (UM) para o coordenador (UMC);
  - Do dispositivo (UM) para o dispositivo (UM).

- **FFD e RFD:** O coordenador (UMC) e todas as UM foram programados na linguagem *script* Otcl como sendo FFD, ou seja, as UM se comunicam entre si e com a UMC que é o coordenador da rede; Todos os consumidores foram programados como sendo RFD, ou seja, eles não se comunicam entre si, apenas se comunicam com a UM mais próxima e esta se comunica com a UMC;

- **Rede:** A rede *ZigBee* suporta 255 nós ativos por "coordenador de rede", o SRMP possui 49 nós no total (rota completa). O *ZigBee* suporta múltiplos "coordenadores

de rede" eles podem ser ligados para suportar redes extremamente grandes. Neste trabalho é utilizado apenas um coordenador para suportar toda a rede;

- **Modo *non- beacon*:** O coordenador (UMC) foi programado (por *software*) em modo *non-beacon*, ou seja, não há uma sinalização (*beacon*) antes da transmissão. Essa sinalização é feita para sincronizar os dispositivos e para alterá-los do modo *sleep* para o modo ativo. Os dados são transmitidos da UMC (coordenador) passando pelas UM (roteadores) e chegando até o consumidor (*end-device*) em questão. O método utilizado para essa transferência de dados é o CSMA/CA.

- **Coordenador (UMC)**: Não há necessidade de uma Tabela de rotas para a UMC armazenar suas rotas, os próprios protocolos de roteamento já possuem suas próprias Tabelas de roteamento;

- **Protocolos de Roteamento:** Utilizamos o AODV e o DSR, já citados anteriormente, pois o protocolo *ZigBee* tem dois tipos de roteamento: *on-demand* (*Mesh Routing*) e híbridos (*Tree Routing*). Os protocolos pró-ativos não são nem *Mesh* e nem *Tree*. Os híbridos não são compatíveis com o simulador de redes utilizado. Pelo fato de estarmos trabalhando com uma rede *Mesh* e com o simulador de redes (NS-2), foram escolhidos dois protocolos reativos.

No anexo A.2, há uma detalhada explicação do *script* em Otcl que foi utilizado para demonstrar os itens acima.

4.3 Metodologia Experimental

Com o objetivo de induzir algumas falhas, foram simuladas algumas quebras de enlaces através da retirada de alguns dispositivos e, conseqüentemente, a topologia foi alterada. Essas falhas podem ser decorrentes da não comunicação dos postes de energia elétrica podendo haver a quebra de um *link* onde uma UM e um consumidor parem de se comunicar devido a uma possível interferência na rede. Essa interferência pode ser causada ou por nós ocultos (uma UM instalada no poste em uma das esquinas torna-se

um nó oculto, isso acontece quando uma UM deixa de ter a visada da UM seguinte) ou pela construção de um prédio alto ou por árvores fazendo com que isso diminua a visibilidade entre os postes.

Foi utilizada a metodologia de retirada gradativa de dispositivos com o intuito de aumentar as dificuldades passo a passo para analisar o comportamento dos protocolos em cada caso. Cada caso de retirada de dispositivos denominar-se-á Estratégia.

A Tabela 3 mostra as estratégias, as rotas a que pertencem, os dispositivos que foram retirados de cada uma e a definição de cada uma.

Estratégias	**Rotas**	**Dispositivos retirados**		**Definição**
		UM	**Consumidor**	
1ª Estratégia	Rota Completa			Situação Real do SMRP
2ª Estratégia	Rota Incompleta - 1quebra	UM4	Cons30	
3ª Estratégia	Rota Incompleta - 2quebra	UM4 **UM13**	Cons30 **Cons45**	Isola uma rua
4ª Estratégia	Rota Incompleta - 3quebra	UM4 UM13 **UM10**	Cons30 Cons45 **Cons48**	
5ª Estratégia	Rota Incompleta - 4quebra	UM4 UM13 UM10 **UM6**	Cons30 Cons45 Cons48 **Cons28**	Elimina a UM da primeira transmissão de dados
6ª Estratégia	Rota Incompleta - 5quebra	UM4 UM13 UM10 UM6 **UM15**	Cons30 Cons45 Cons48 Cons28 **Cons43**	Elimina duas UM das duas transmissões de dados
7ª Estratégia	Rota Completa			Caso a UMC falhe

Tabela 3 – Estratégias

Todas as estratégias foram simuladas para ambos os protocolos de roteamento. Para isso foram utilizados os seguintes parâmetros do simulador (NS-2):

- Tamanho da rede: Inicialmente com 49 nós;
- Área: 80 m x 80 m;

- Fontes de tráfego: *Constant Bit Rate* (CBR), *File Transfer Protocol (FTP) e Poisson;*
- Modelo de propagação de radio [44]: *Two Ray Ground* (considera que as antenas, transmissora e receptora, estejam localizadas sobre uma superfície plana, sem que haja obstáculos entre elas);
- Tipo de Interface de Rede: Mac/802_15_4;
- Freqüência: 2.4GHz;
- Protocolos de Roteamento: AODV e DSR (reativos);
- Tempo de simulação: 100s.

Na topologia de rede utilizada (SMRP) foram feitas todas as retiradas possíveis já que o alcance máximo do protocolo *ZigBee* na rede é de 100m e os postes de distribuição de energia elétrica distam de 40m. Se houver a retirada de dois dispositivos seguidos em uma mesma rota, os protocolos terão que alterar o caminho (rota).

Para a análise gráfica foram utilizados dois parâmetros, baseados no seguinte critério: o protocolo que perder uma menor quantidade de pacotes e o protocolo que receber uma maior quantidade de pacotes terá o melhor desempenho.
Pacotes recebidos com sucesso: Constam da quantidade do número de pacotes que foram enviados e que chegaram ao destino com sucesso por unidade de tempo;
Quantidade de pacotes perdidos: Constam da quantidade do número de pacotes que foram enviados, porém não chegaram ao destino com sucesso.

1° Estratégia

A 1° Estratégia é a situação real do SRMP denominada Rota Completa. A Figura 29 corresponde a topologia de rede utilizada nas simulações da Rota Completa no NS-2 visualizada na tela do *Network Animator* (NAM).

No protocolo AODV, as rotas são programadas nas Tabelas de roteamento, a UMC e cada UM possuem uma Tabela e todas as UM se comunicam entre si. Já no DSR,

essas rotas ficam armazenadas no *cachê* de rotas e só quem tem o conhecimento de uma rota (do nó fonte ao nó destino) é o nó fonte (UMC).

Tanto no AODV quanto no DSR, quando o nó fonte não conhece o caminho para o nó destino, ele inicia um processo de descobrimento de rotas. Este processo também é chamado de Inundação e ocorre quando o nó fonte envia mensagens de requisições de rota (RREQ). Essas mensagens são enviadas por *broadcast*, ou seja, todos os nós da rede as recebem.

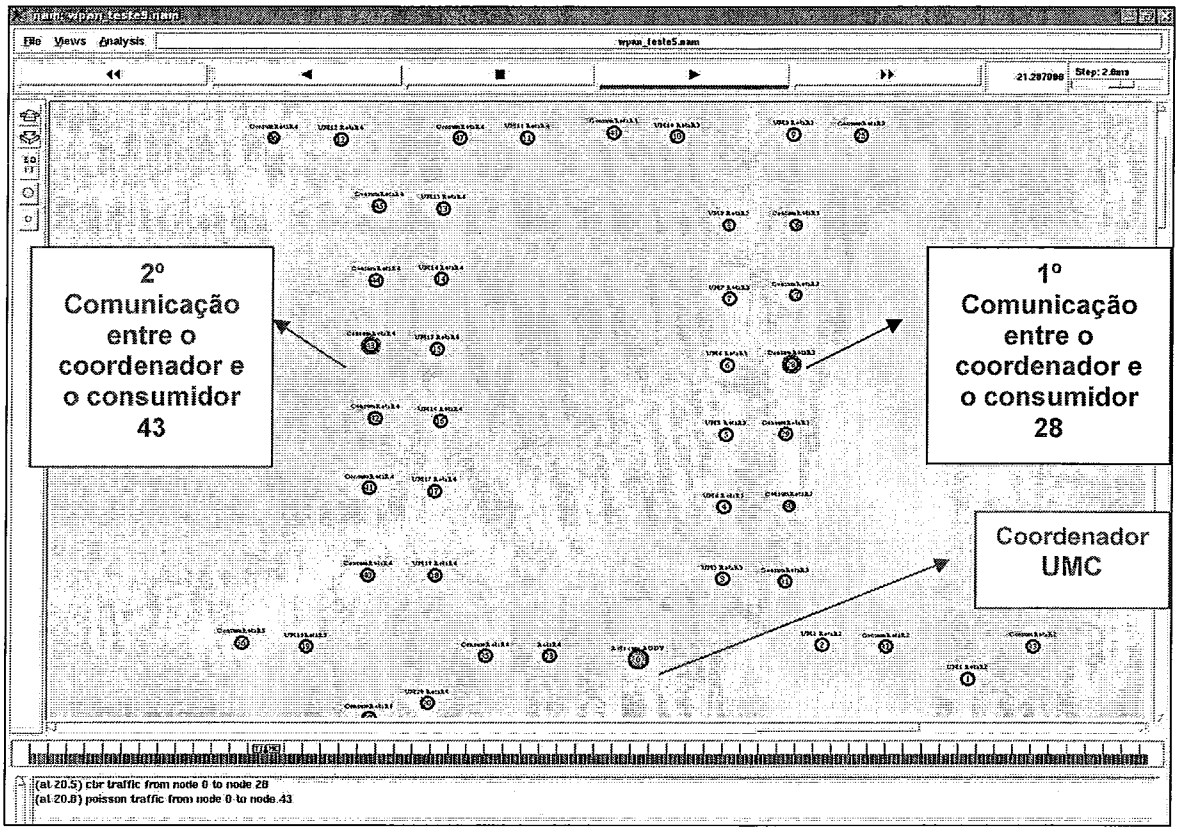


Figura 29 – Topologia de Rede utilizada na simulação da Rota Completa

2º Estratégia

A Rota Incompleta é idêntica a rota completa da 1ª estratégia com as mesmas comunicações. Porém para mostrar a quebra de um link decorrente de alguma falha na topologia de rede, foi retirada uma UM (UM4) e um consumidor (cons30) denominada Rota Incompleta – 1 quebra. Isso foi feito para analisar o comportamento e o desempenho dos protocolos utilizados faltando dois nós da rede.

A escolha das UM retiradas foi feita em ordem crescente da dificuldade, foi tirada a UM4 e o consumidor 30 justamente porque eles estão na rota do caminho que o protocolo fará para comunicar o coordenador (UMC) com a UM6 (consumidor 28), que é a 1º comunicação programada para o simulador executar.

Como as UM distam 40m uma da outra, mesmo com a retirada da UM4, a UM3 se comunica diretamente com a UM5.

A Figura 30 mostra a topologia de rede utilizada nas simulações da Rota Incompleta – 1 quebra:

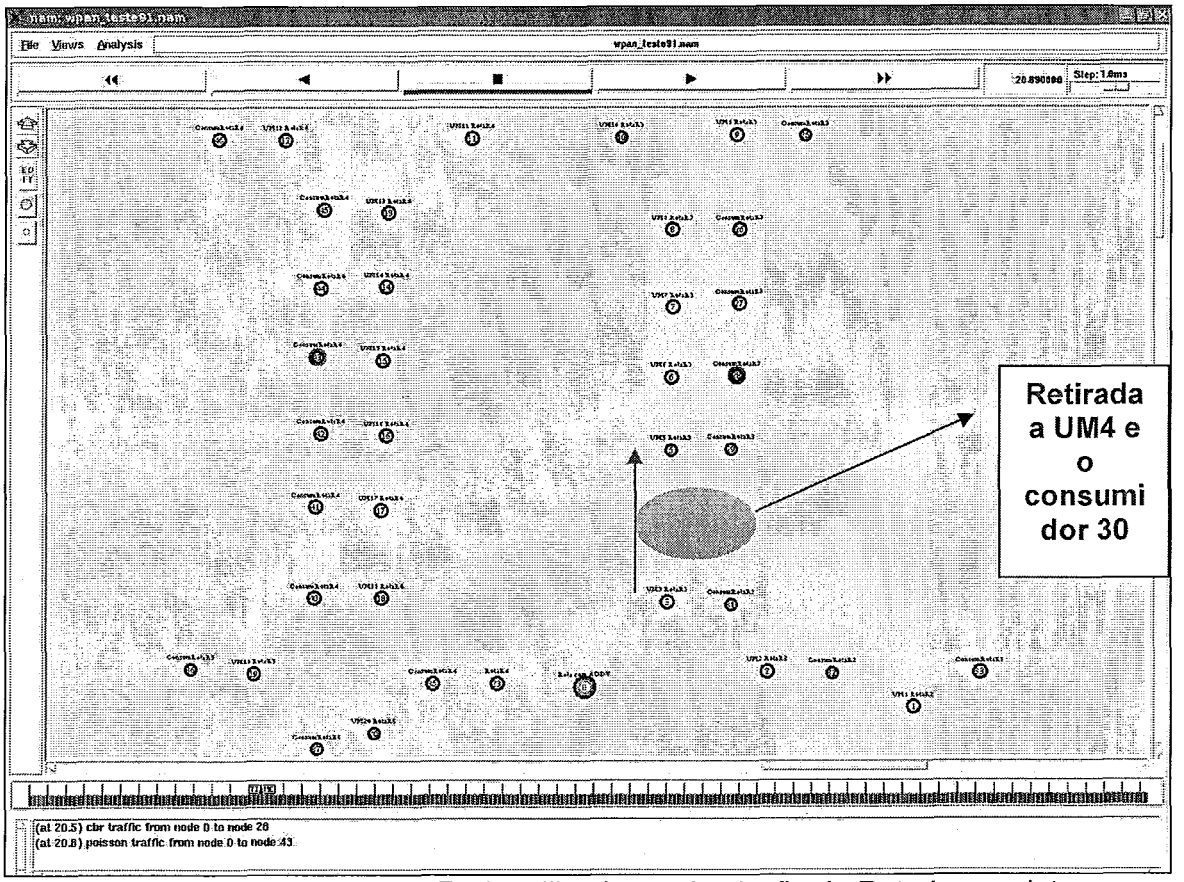


Figura 30 – Topologia de Rede utilizada na simulação da Rota Incompleta – 1 quebra

3° Estratégia

É idêntica a rota acima, porém com menos duas UM (UM4 e UM13) e menos dois consumidores (30 e 45) chamaremos de Rota Incompleta – 2 quebras.

Como estamos analisando uma quadra formada por quatro ruas, os nós foram escolhidos de uma maneira que as UM e os consumidores de uma determinada rua ficassem isolados do restante da quadra. Pela análise da simulação em questão, o comportamento dos nós que foram isolados, não ficaram fora da rota, pois como as UM distam 40m uma da outra, mesmo com a retirada da UM4, a UM3 se comunica diretamente com a UM5 e mesma com a retirada da UM13, a UM12 se comunica diretamente com a UM14.

A Figura 31 mostra a topologia de rede utilizada nas simulações da Rota Incompleta – 2 quebras:

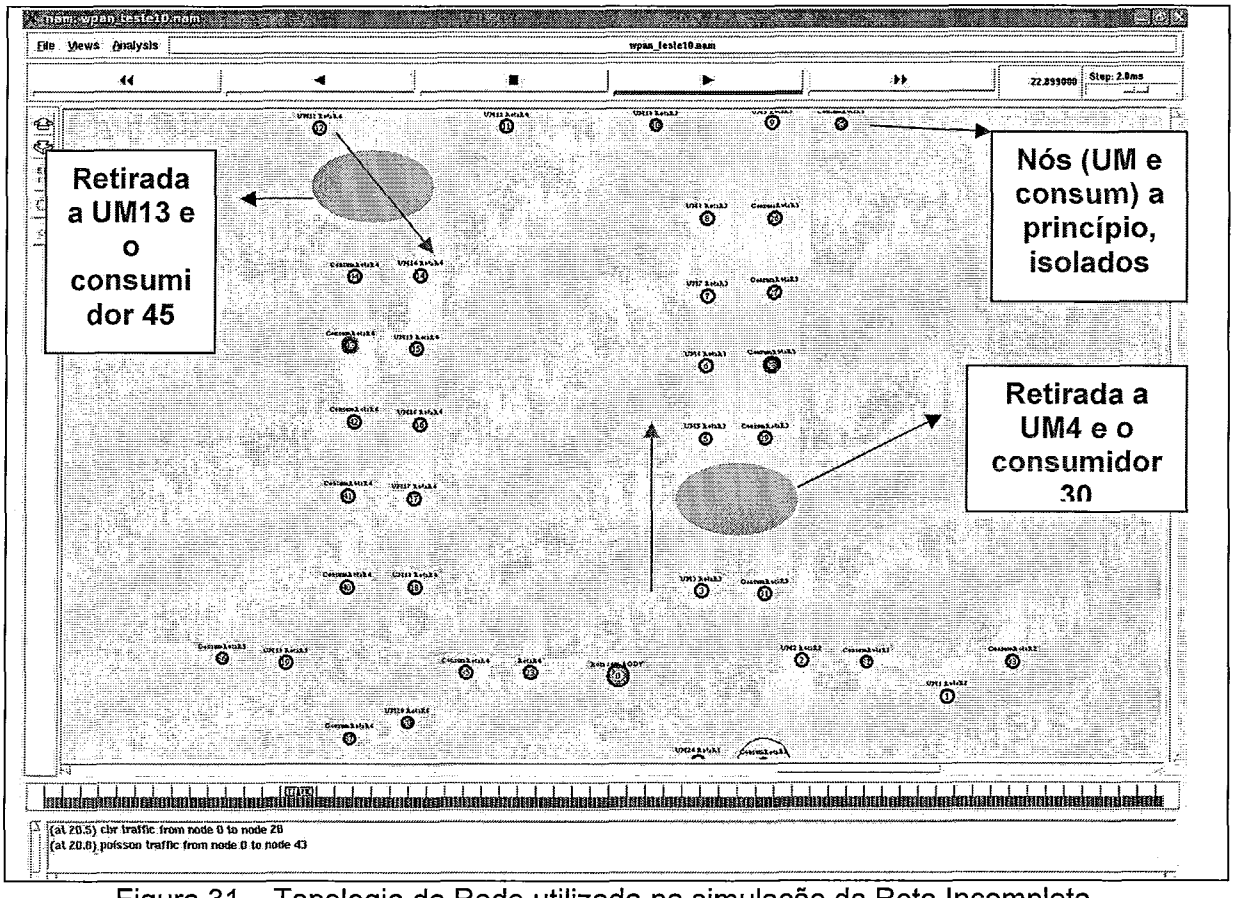


Figura 31 – Topologia de Rede utilizada na simulação da Rota Incompleta – 2 quebras

4º Estratégia

É idêntica a rota acima, porém com menos três UM (UM4, UM13, UM10) e menos três consumidores 30, 45, 48 respectivamente. Denominamos de Rota Incompleta - 3 quebras.

Mesmo sem a UM4, a UM3 se comunica diretamente com a UM5. Mesmo sem a UM13, a UM12 se comunica diretamente com a UM14 e mesmo sem a UM10, a UM9 se comunica diretamente com a UM12.

A Figura 32 mostra a topologia de rede utilizada nas simulações da Rota Incompleta – 3 quebras.

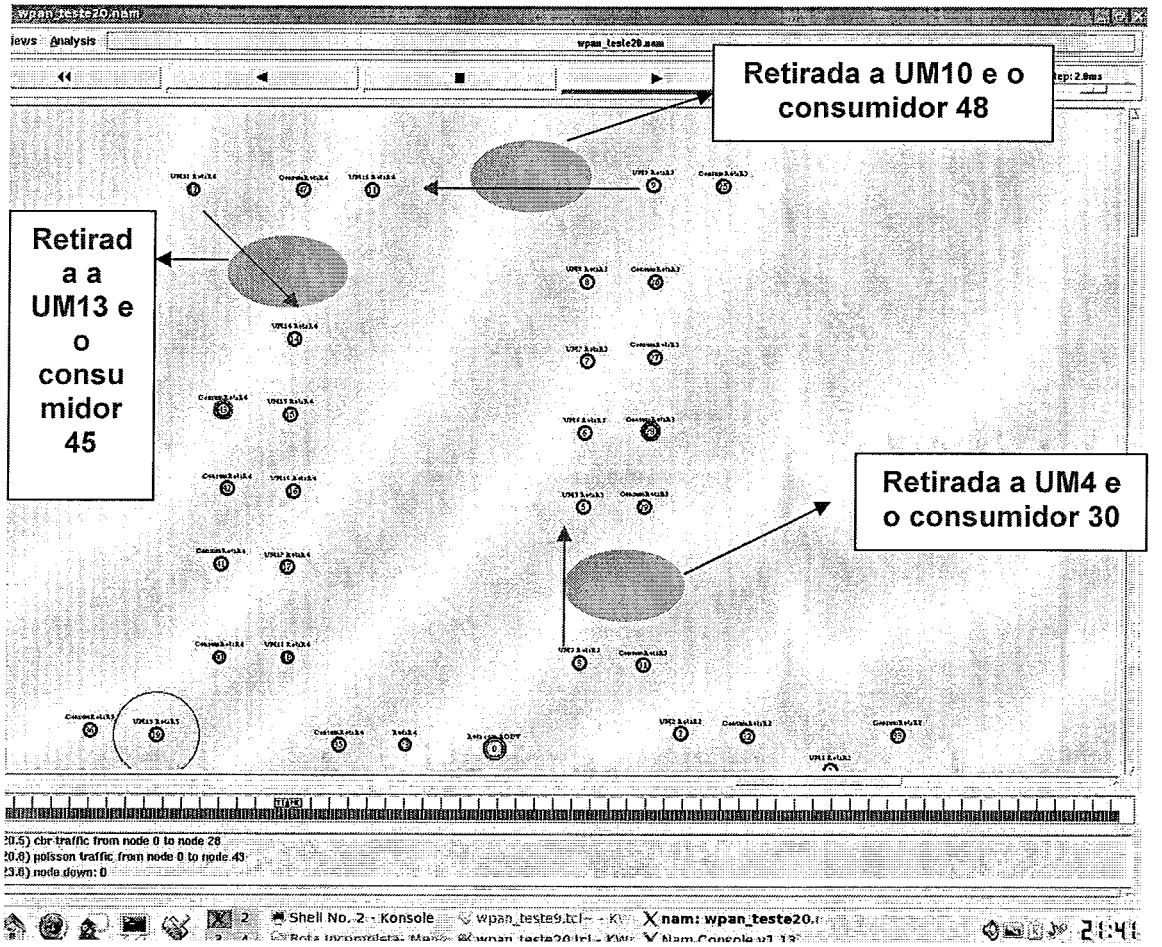


Figura 32 - Topologia de Rede utilizada na simulação da Rota Incompleta – 3 quebras

5º Estratégia

É idêntica a rota acima, porém com menos quatro UM: UM4, UM13, UM10 e UM6 e menos quatro consumidores 30, 45, 48 e 28 respectivamente. Chamaremos de Rota Incompleta - 4 quebras.

Mesmo sem a UM4, a UM3 se comunica diretamente com a UM5. Mesmo sem a UM13, a UM12 se comunica diretamente com a UM14, mesmo sem a UM10, a UM9 se comunica diretamente com a UM12 e mesmo sem a UM6, a UM5 se comunica com a UM7. Porém, a UM6 (consumidor 28) faz parte da primeira comunicação em que a UMC se comunica com o consumidor 28 em 20.5s.

Mesmo sem a UM que faz a primeira comunicação, a UMC estando em funcionamento e utilizando o protocolo de roteamento AODV, ela armazena e guarda na sua Tabela de roteamento as ultimas informações da UM6 e obriga a todas as UM intermediárias a se atualizarem também, ou seja, quando a UM6 voltar ao normal, ela atualizará seus dados. Caso a UMC esteja utilizando o protocolo de roteamento DSR, este guardará as últimas informações no seu *cachê* de rotas, sendo que essas informações poderão ser atualizadas ou não, já que neste caso, as UM intermediárias não se comunicam entre si.

O restante das UM da rede, caso estejam usando o AODV também guardam suas informações em suas Tabelas de roteamento para quando houver a normalidade da rede, essas Tabelas serem atualizadas com os valores recentes e serem encaminhadas a UMC. Caso esteja utilizando o protocolo de roteamento DSR, pode ser que as UM intermediárias não atualizem suas informações de roteamento, devido ao fato de somente a UMC (fonte) ter acesso a essas informações e dos nós intermediários não se comunicarem entre si.

A Figura 33 mostra a topologia de rede utilizada nas simulações da Rota Incompleta – 4 quebras.

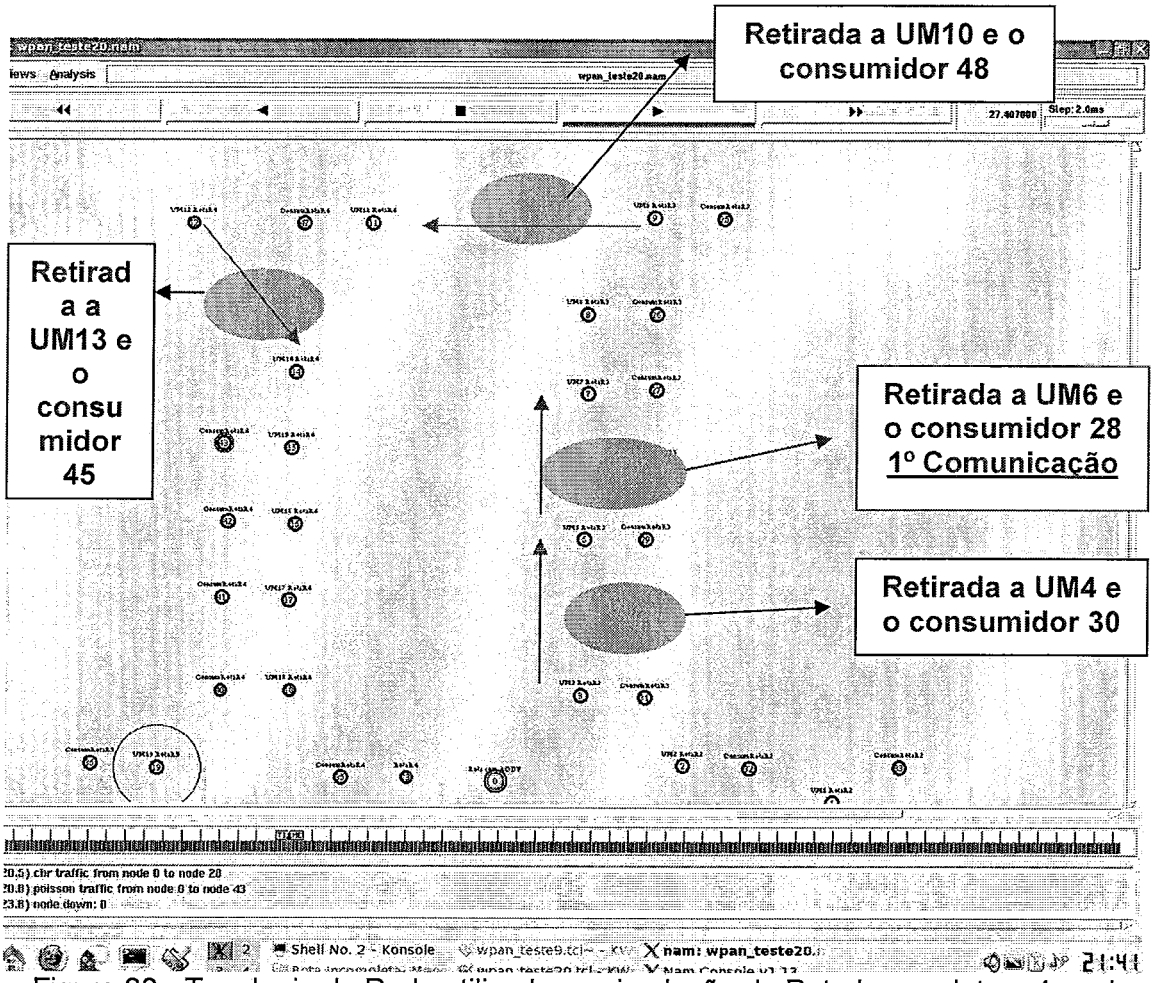


Figura 33 - Topologia de Rede utilizada na simulação da Rota Incompleta – 4 quebras

6º Estratégia

É idêntica a rota acima, porém com menos cinco UM: UM4, UM13, UM10 e UM6, e UM15 e menos cinco consumidores 30, 45, 48, 28 e 43 respectivamente. Sendo que a UM6 (consumidor 28) e a UM15 (consumidor 43) que nesta estratégia foram eliminadas fazem parte da primeira e segunda comunicação, respectivamente. Na primeira comunicação a UMC se comunica com o consumidor 28 em 20.5s e na segunda comunicação a UMC se comunica com o consumidor 43 em 20.8s. Denominanos de Rota Incompleta – 5 quebras.

Mesmo sem a UM4, a UM3 se comunica diretamente com a UM5, mesmo sem a UM13, a UM12 se comunica diretamente com a UM14, mesmo sem a UM10, a UM9 se comunica diretamente com a UM11 e mesmo sem a UM15, a UM14 se comunica

diretamente com a UM16. A UMC, estando em funcionamento é feito o mesmo procedimento da 5ª estratégia para ambos os protocolos de roteamento.

A Figura 34 mostra a topologia de rede utilizada nas simulações da Rota Incompleta – 5 quebras:

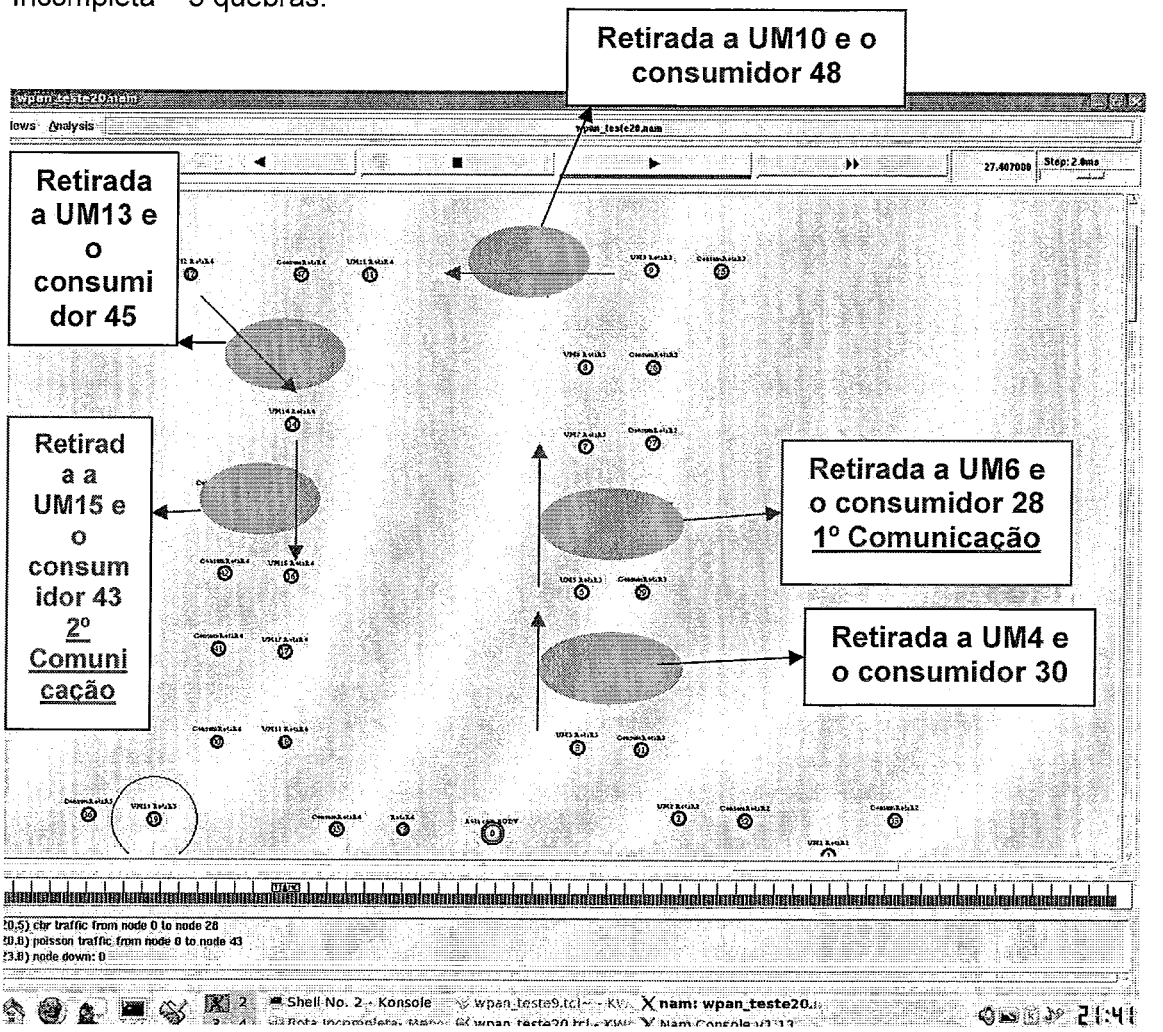


Figura 34 - Topologia de Rede utilizada na simulação da Rota Incompleta – 5 quebras

7º Estratégia

Denominada de Rota Completa - Caso a UMC deixe de funcionar. Se o link com a UMC cair devido a algum motivo nada na rede funcionará já que é ela que é a coordenadora de rede e é dela que partem todas as comunicações, apenas os roteadores que são FFD continuarão se comunicando entre si e resgatando a informação de seus consumidores. Para recuperar os dados da UMC, será necessário algum técnico ou engenheiro ir até o local munido de um palm top ou de um lap top, ao inves de subir no poste, essa pessoa insere um palm top na rede e consegue se comunicar com os demais nós da rede, reiniciando assim a UMC, como mostra a Figura 37.

Este palm top corresponde a um outro nó (nó 50) que entraria na rede no momento que a UMC deixasse de funcionar e ele seria um nó FFD para assim se comunicar com todas as UM e com a UMC.

A Figura 35 mostra a topologia de rede utilizada nas simulações da Rota Completa caso a UMC deixe de funcionar.

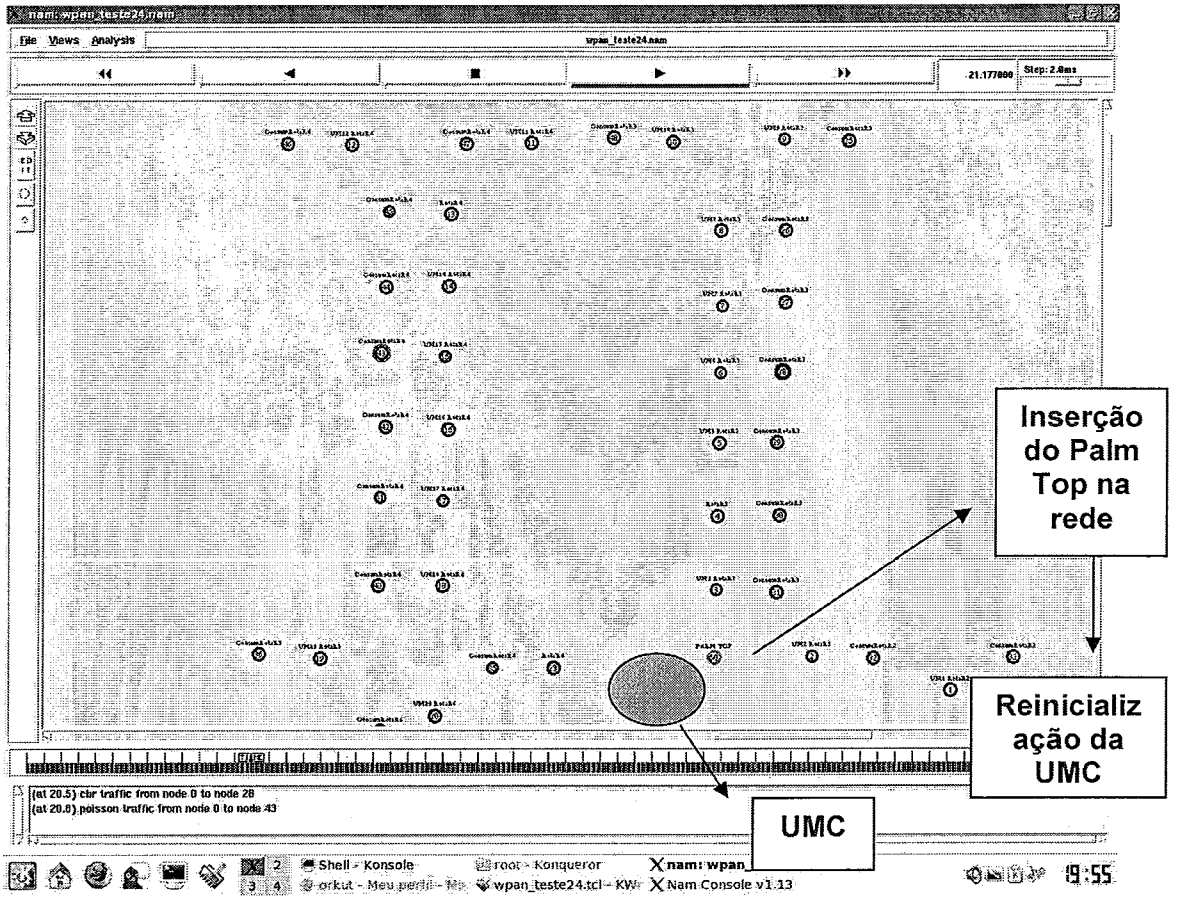


Figura 35 - Topologia de Rede utilizada na simulação da Rota Incompleta – Caso a UMC deixe de funcionar

4.4 Resultados Obtidos

Abaixo são citadas algumas ferramentas de geração de gráficos conhecidas e utilizadas:

- ***XGraph*:** É uma ferramenta que vem junto com a distribuição do NS-2. Roda nas plataformas*: Windows, Unix, Linux, FreeBSD, Solaris* [45];
- ***Trace graph*:** É uma ferramenta gratuita, é executada no NS-2. Roda em *Windows, Linux, UNIX e MAC OS Systems*. Porém para se usar o *Tracegrap*h, exige-se o *software Matlab* (que não é gratuito) para versão *Windows* e de algumas bibliotecas do *Matlab* para a versão *Linux*, mas estas são disponibilizadas juntamente com o *Tracegraph* [46];

- ***GNUPlot***: É uma ferramenta gratuita que roda no *Linux, DOS, Windows, Macintosh OS*, é interativo, pode-se plotar gráficos em 2D e 3D além de ser capaz de fazer pontos, linhas, pontos e linhas, barras, superfícies [47, 48, 49].

Para este trabalho, foi escolhida a ferramenta *GNUPlot* devido a facilidade de utilização, por ser uma ferramenta gratuita, executada no sistema operacional *Windows* e por ter inúmeras referências sobre ele. Os *scripts* utilizados podem ser encontrados no Anexo A.1.

As simulações no NS-2, os arquivos de *trace (trace file)* e o aplicativo *GNUPlot* resultaram na construção dos gráficos a seguir. Primeiramente analisar-se-ão os gráficos da quantidade dos pacotes perdidos com os protocolos de roteamento (AODV e DSR) para as sete estratégias e as Tabelas com as devidas referências relativas (%). Posteriormente será feita a análise dos gráficos dos pacotes recebidos com sucesso utilizando os mesmos critérios acima.

1ª Estratégia: Pacotes perdidos com a Rota Completa

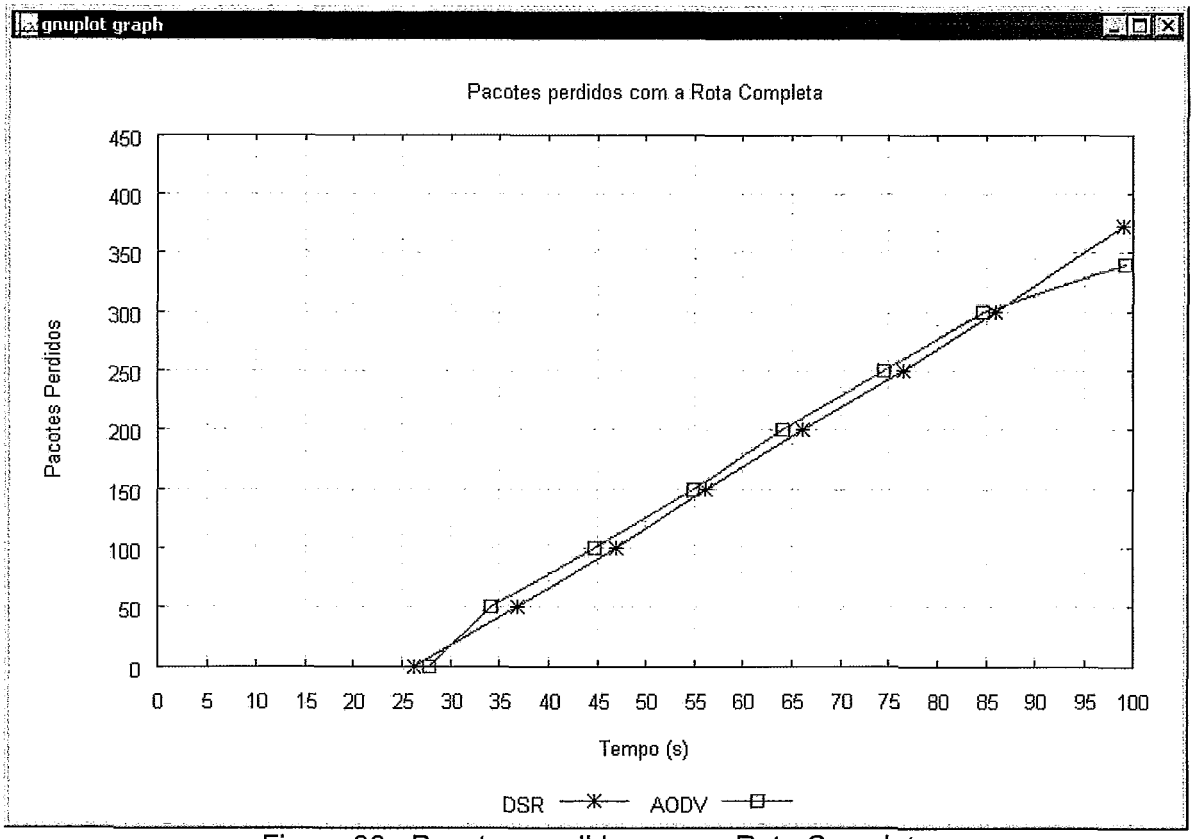


Figura 36 - Pacotes perdidos com a Rota Completa

Para todas as estratégias, as porcentagens dos pacotes perdidos foram feitas com base no AODV utilizando o critério: pacotes perdidos no DSR menos os pacotes perdidos no AODV, divididos pelo número de pacotes perdidos no AODV, multiplicado por 100. Esse cálculo resultou a porcentagem mostrada na Tabela 4, onde é mostrada a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,1	99,0
Pacotes Perdidos	340	373
Diferença Relativa (%)	10%	

Tabela 4 – Diferença Relativa dos Pacotes perdidos com a Rota Completa

Analisando o gráfico e posteriormente a Tabela 4, conclui-se que na estratégia da Rota Completa, o AODV perdeu 10% a menos que o DSR.

2ª Estratégia – Pacotes perdidos com a Rota Incompleta – 1 quebra

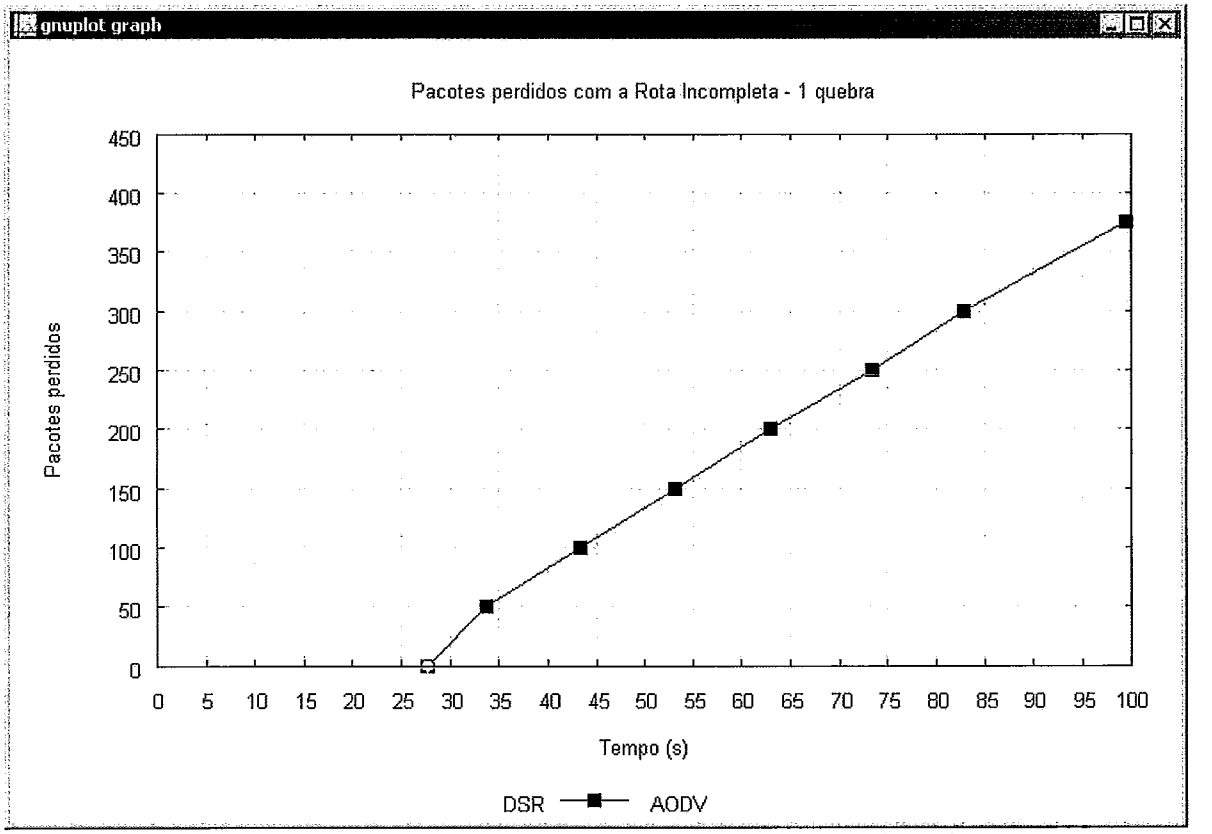


Figura 37 - Pacotes perdidos na Rota Incompleta – 1 quebra

A Tabela 5 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,2	99,5
Pacotes Perdidos	350	375
Diferença Relativa (%)	7%	

Tabela 5 – Diferença Relativa dos Pacotes perdidos com a Rota Incompleta – 1 quebra

Analisando o gráfico e posteriormente a Tabela 5, conclui-se que na estratégia da Rota Incompleta – 1 quebra, o AODV perdeu 7% a menos que o DSR.

3ª Estratégia – Pacotes perdidos com a Rota Incompleta – 2 quebras

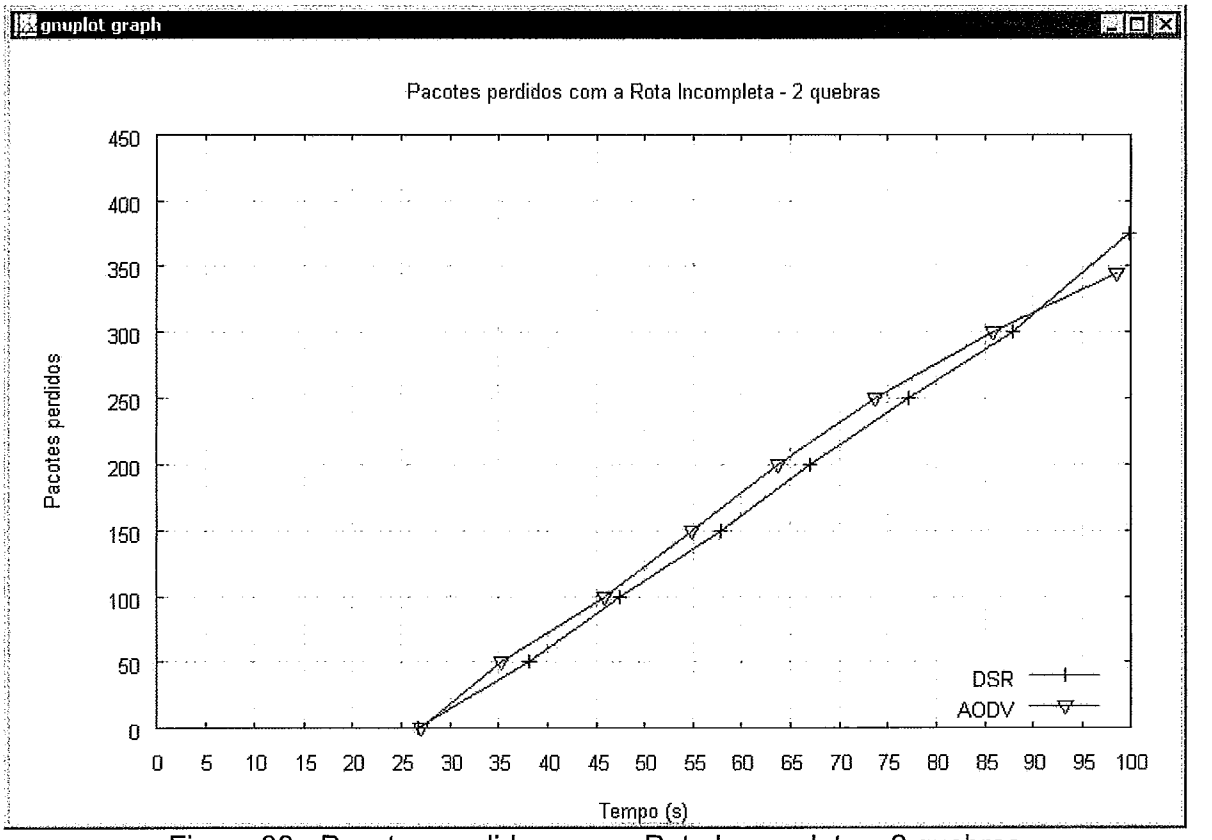


Figura 38 - Pacotes perdidos com a Rota Incompleta – 2 quebras

A Tabela 6 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	98,5	99,8
Pacotes Perdidos	345	375
Diferença Relativa (%)	8%	

Tabela 6 – Diferença Relativa dos Pacotes perdidos com a Rota Incompleta – 2 quebras

Analisando o gráfico e posteriormente a Tabela 6, conclui-se que na estratégia da Rota Incompleta - 2 quebras, o AODV perdeu 8% a menos que o DSR.

4ª Estratégia: Pacotes perdidos com a Rota Incompleta – 3 quebras

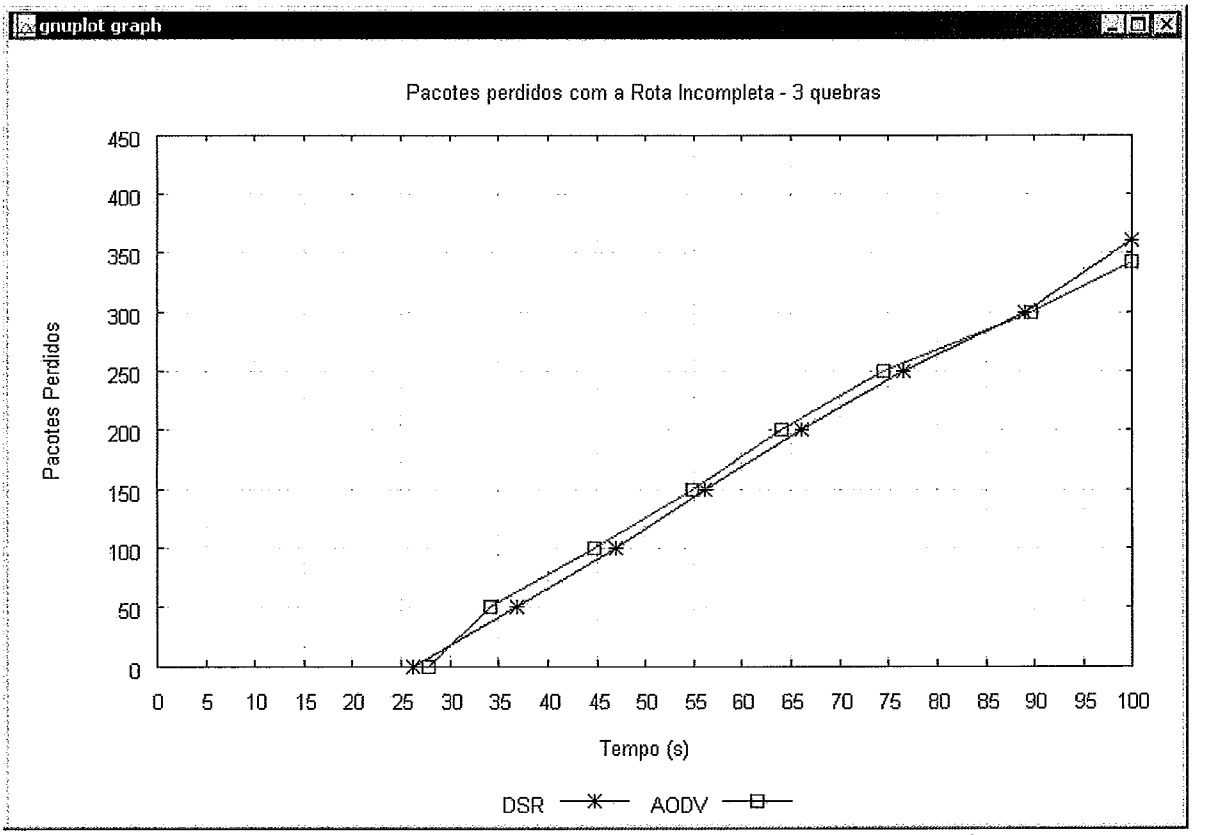


Figura 39 – Pacotes perdidos com a Rota Incompleta – 3 quebras

A Tabela 7 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,9	99,9
Pacotes Perdidos	343	361
Diferença Relativa (%)	5%	

Tabela 7 – Diferença Relativa dos Pacotes perdidos com a Rota Incompleta – 3 quebras

Analisando o gráfico e posteriormente a Tabela 7, conclui-se que na estratégia da Rota Incompleta - 3 quebras, o AODV perdeu 5% a menos que o DSR.

5ª Estratégia – Pacotes com a Rota Incompleta – 4 quebras

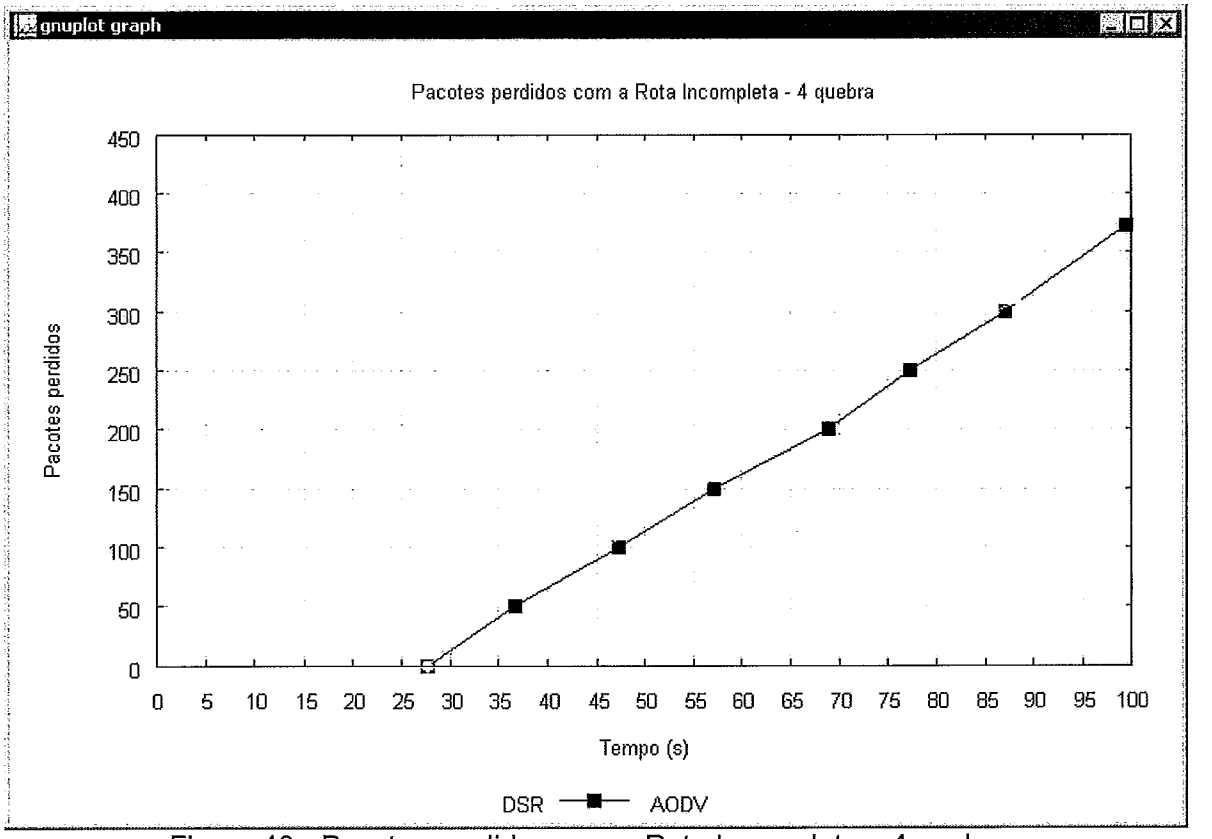


Figura 40 - Pacotes perdidos com a Rota Incompleta – 4 quebras

A Tabela 8 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,3	99,5
Pacotes Perdidos	340	373
Diferença Relativa (%)	9%	

Tabela 8 – Diferença Relativa dos Pacotes perdidos com a Rota Incompleta – 4 quebra

Analisando o gráfico e posteriormente a Tabela 8, conclui-se que na estratégia da Rota Incompleta - 4 quebras, o AODV perdeu 9% a menos que o DSR.

6ª Estratégia – Pacotes perdidos com a Rota Incompleta – 5 quebras

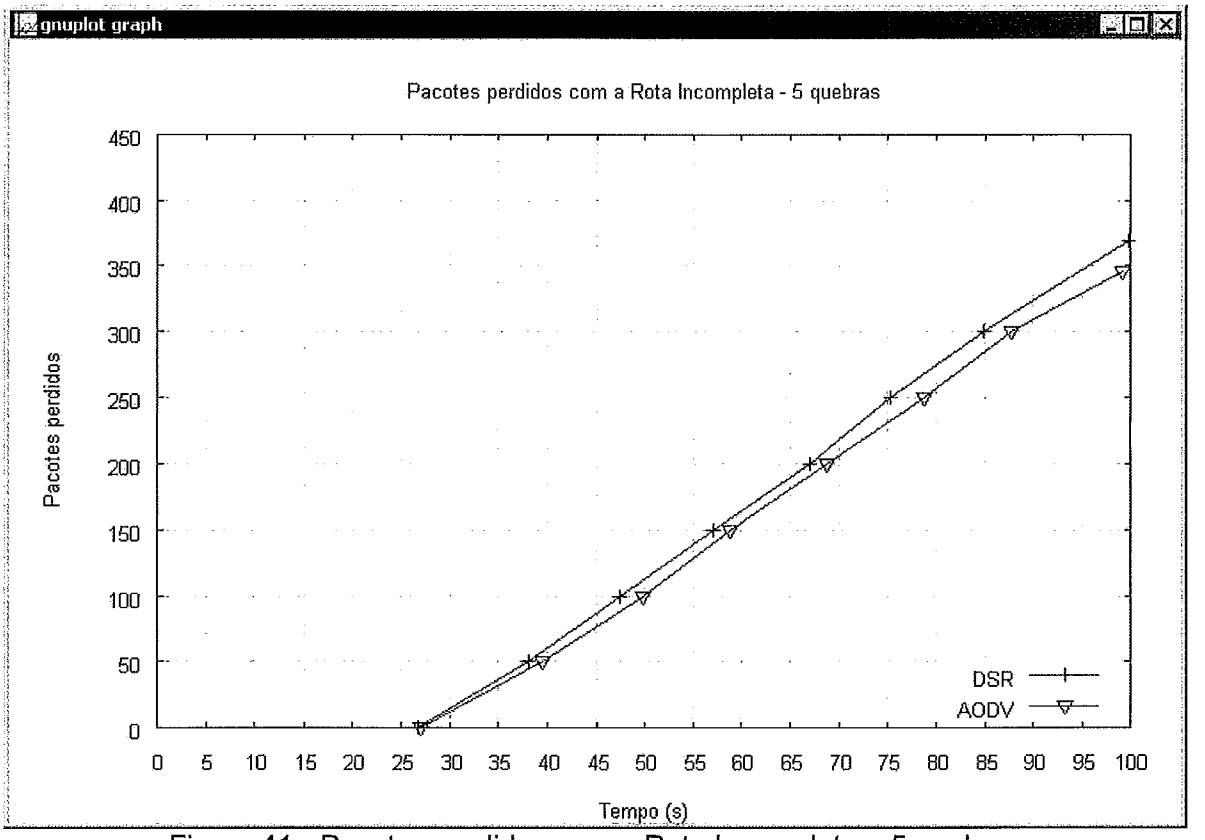


Figura 41 - Pacotes perdidos com a Rota Incompleta – 5 quebras

A Tabela 9 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,1	99,8
Pacotes Perdidos	346	369
Diferença Relativa (%)	6%	

Tabela 9 – Diferença Relativa dos Pacotes perdidos com a Rota Incompleta – 5 quebras

Analisando o gráfico e posteriormente a Tabela 9, conclui-se que na estratégia da Rota Incompleta - 5 quebras, o AODV perdeu 6% a menos que o DSR.

7ª Estratégia – Pacotes perdidos caso a UMC deixe de funcionar

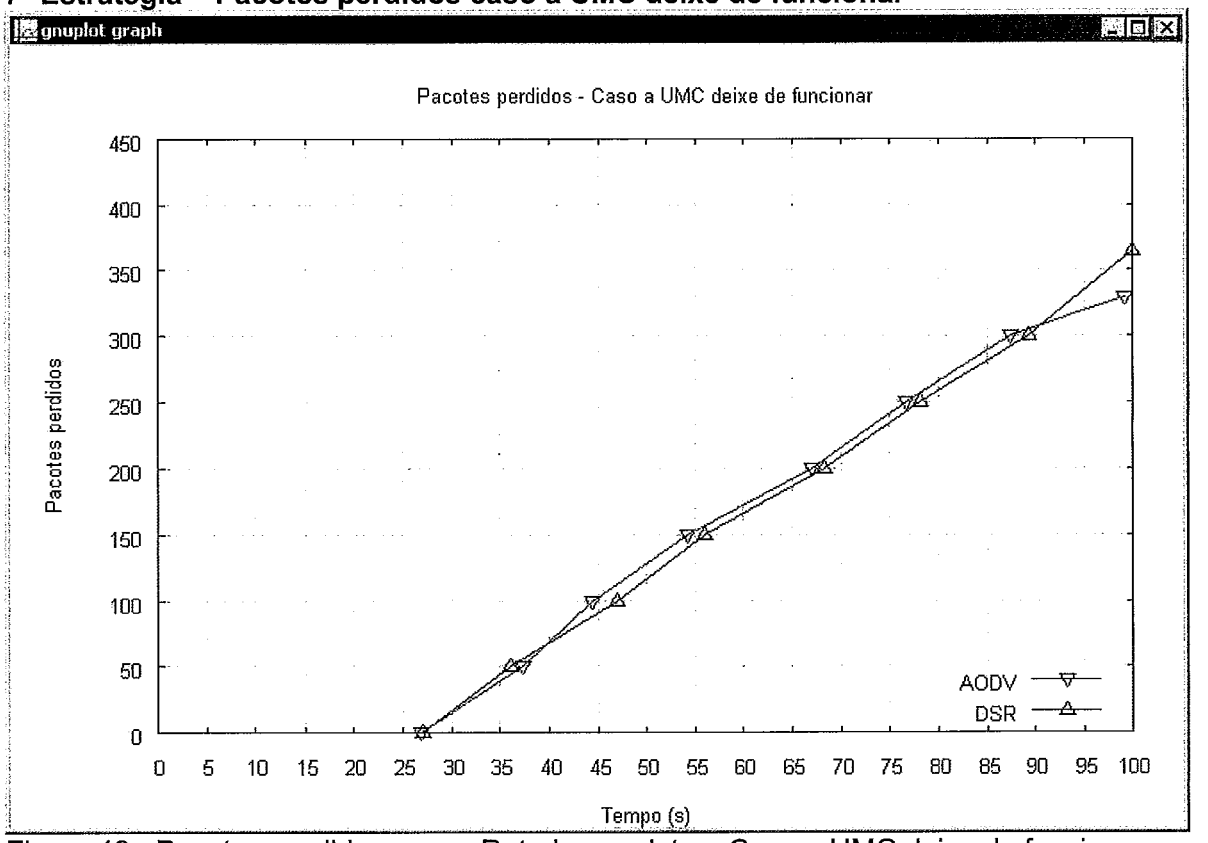


Figura 42 - Pacotes perdidos com a Rota Incompleta – Caso a UMC deixe de funcionar

A Tabela 10 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,1	99,9
Pacotes Perdidos	330	365
Diferença Relativa (%)	9%	

Tabela 10 - Diferença Relativa dos Pacotes perdidos caso a UMC deixe de funcionar

Analisando o gráfico e posteriormente a Tabela 10, conclui-se que na estratégia da Rota Completa - Caso a UMC deixe de funcionar, o AODV perdeu 9% a menos que o DSR.

Abaixo são mostrados os gráficos dos pacotes recebidos com sucesso com os protocolos de roteamento (AODV e DSR) para as sete estratégias. Para estas estratégias, as porcentagens dos pacotes recebidos com sucesso foram feitas com base no AODV utilizando o critério: Pacotes recebidos com sucesso no AODV menos os pacotes recebidos com sucesso no DSR, divididos pelo número de pacotes recebidos com sucesso no AODV, multiplicado por 100. Esse cálculo resultou a porcentagem mostrada nas Tabelas a seguir.

1ª Estratégia: Pacotes recebidos com sucesso com a Rota Completa

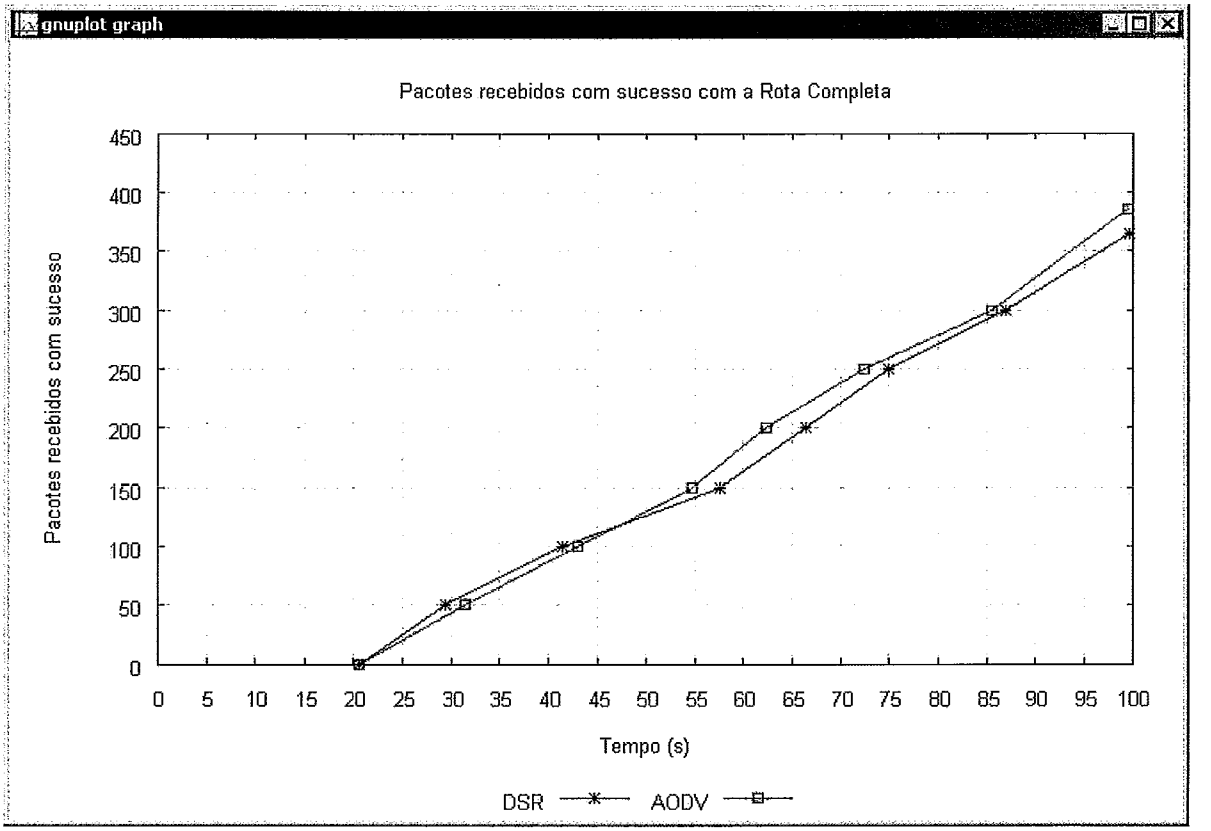


Figura 43 - Pacotes Recebidos com sucesso na Rota Completa

A Tabela 11 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,5	99,7
Pacotes Recebidos	386	365
Diferença Relativa (%)	5%	

Tabela 11 – Diferença Relativa dos Pacotes recebidos com sucesso com a Rota Completa

Analisando o gráfico e posteriormente a Tabela 11, conclui-se que na estratégia da Rota Completa, o AODV recebeu com sucesso 5% a mais que o DSR.

2ª Estratégia – Pacotes recebidos com sucesso com a Rota Incompleta – 1 quebra

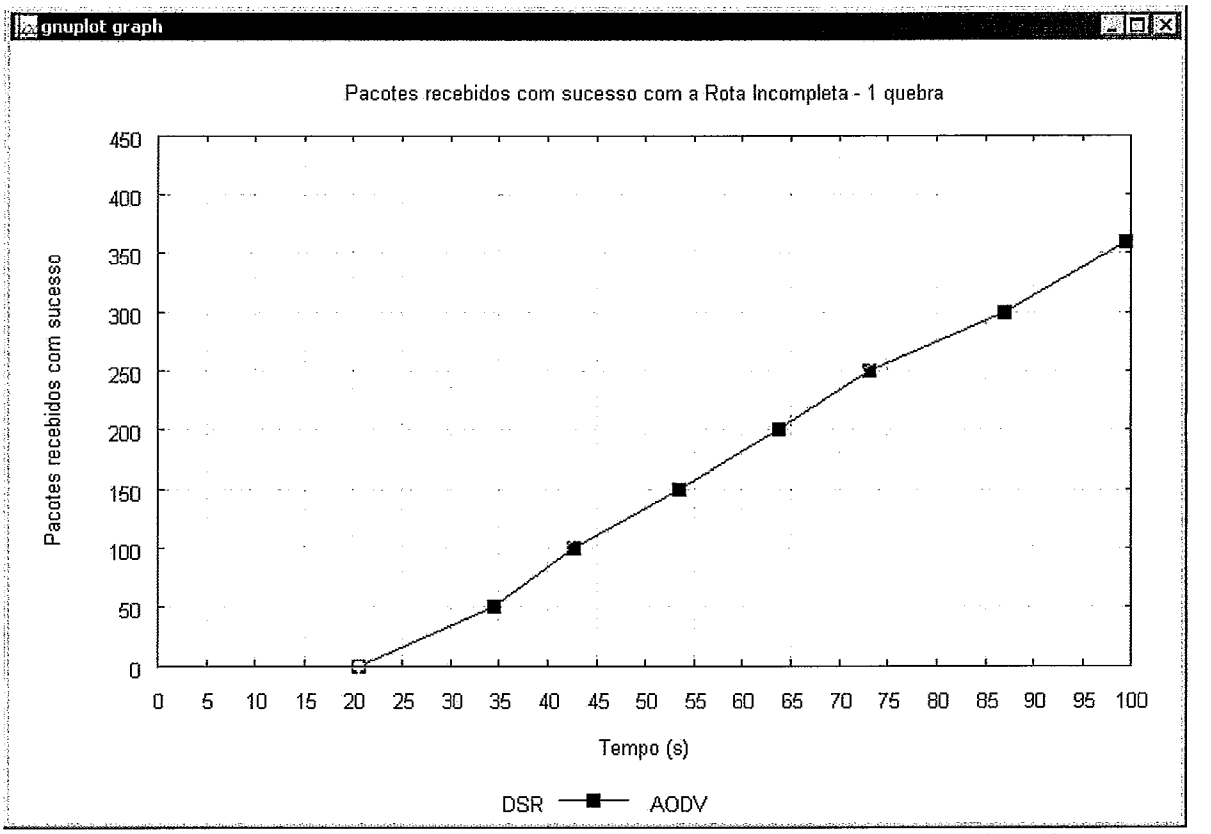


Figura 44 - Pacotes Recebidos com sucesso na Rota Incompleta – 1 quebra

A Tabela 12 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,9	99,5
Pacotes Recebidos	387	360
Diferença Relativa (%)	6%	

Tabela 12 – Diferença Relativa dos Pacotes recebidos com sucesso com a Rota Incompleta – 1 quebra

Analisando o gráfico e posteriormente a Tabela 12, conclui-se que na estratégia da Rota Incompleta – 1quebra, o AODV recebeu com sucesso 6% a mais que o DSR.

3ª Estratégia – Pacotes recebidos com sucesso com a Rota Incompleta – 2 quebras

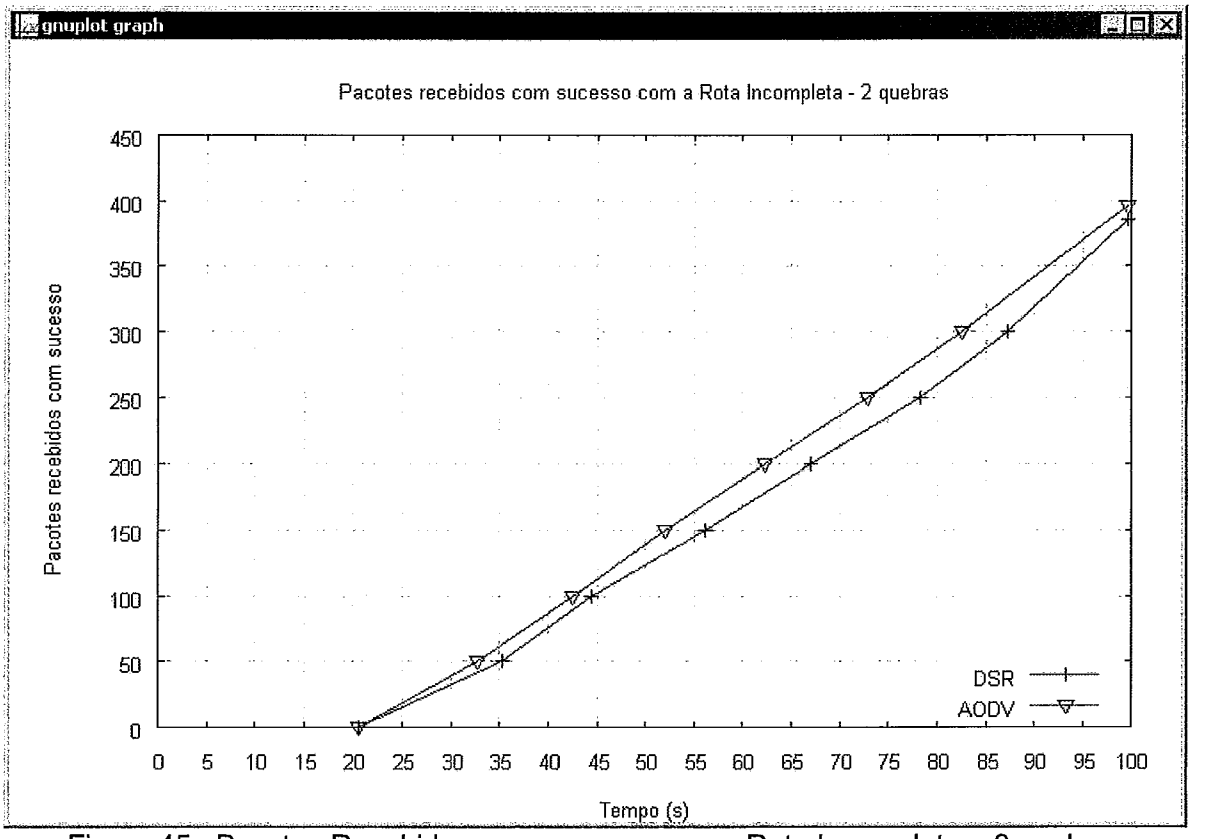


Figura 45 –Pacotes Recebidos com sucesso com a Rota Incompleta – 2 quebras

A Tabela 13 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,7	99,7
Pacotes Recebidos	396	386
Diferença Relativa (%)	2%	

Tabela 13 – Diferença Relativa dos Pacotes recebidos com sucesso com a Rota Incompleta – 2 quebras

Analisando o gráfico e posteriormente a Tabela 13, conclui-se que na estratégia da Rota Incompleta – 2 quebras, o AODV recebeu com sucesso 2% a mais que o DSR.

4ª Estratégia – Pacotes recebidos com sucesso com a Rota Incompleta – 3 quebras

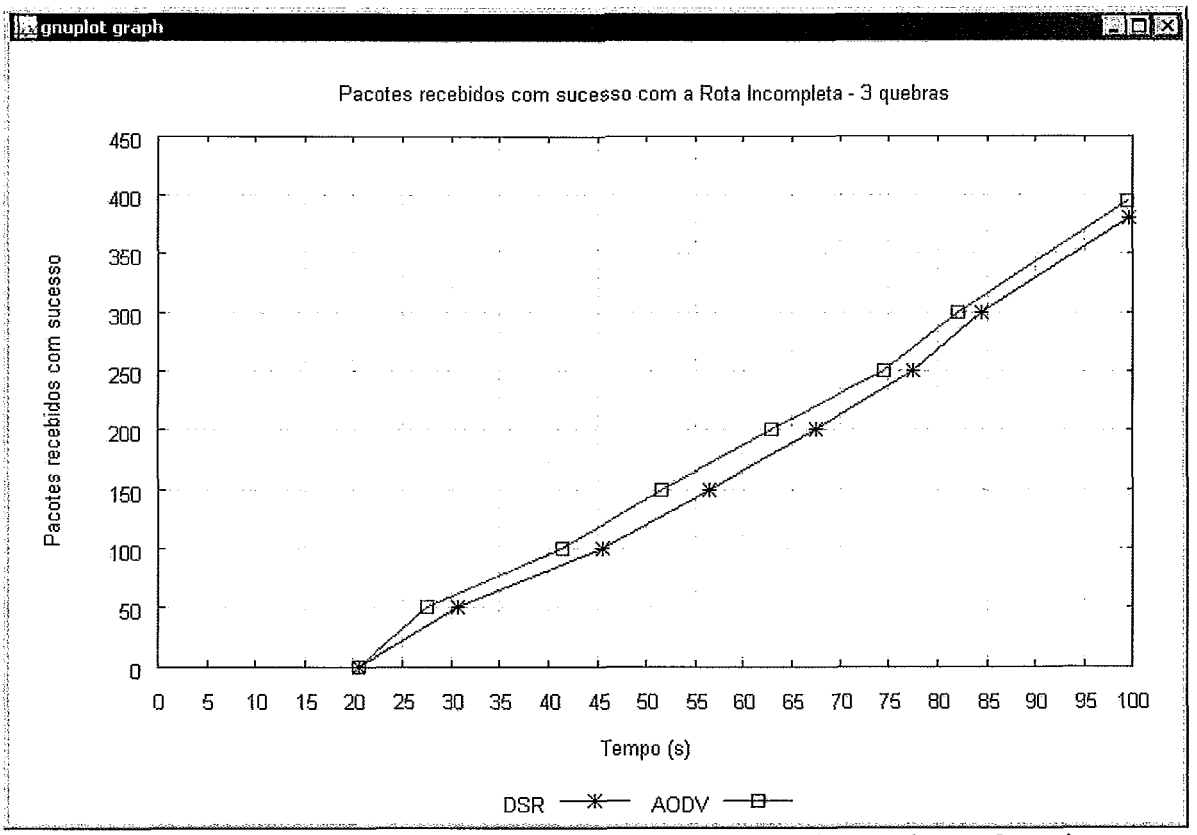


Figura 46- Pacotes Recebidos com sucesso com a Rota Incompleta – 3 quebras

A Tabela 14 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,5	99,7
Pacotes Recebidos	395	380
Diferença Relativa (%)	3%	

Tabela 14 – Diferença Relativa dos Pacotes recebidos com sucesso com a Rota Incompleta – 3 quebras

Analisando o gráfico e posteriormente a Tabela 14, conclui-se que na estratégia da Rota Incompleta – 3 quebras, o AODV recebeu com sucesso 3% a mais que o DSR.

5ª Estratégia – Pacotes recebidos com sucesso com a Rota Incompleta – 4 quebras

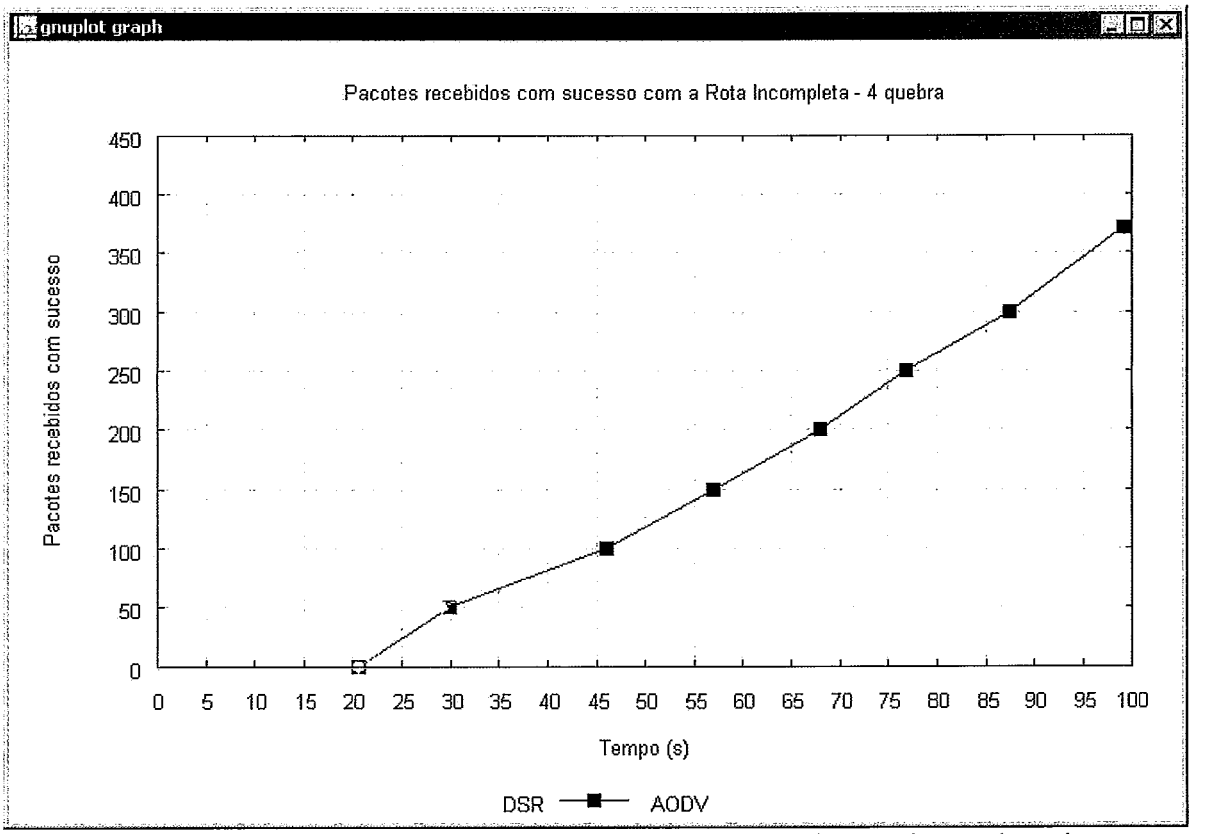


Figura 47- Pacotes Recebidos com sucesso com a Rota Incompleta – 4 quebras

A Tabela 15 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,3	99,1
Pacotes Recebidos	391	372
Diferença Relativa (%)	5%	

Tabela 15 – Diferença Relativa dos Pacotes recebidos com sucesso com a Rota Incompleta - 4 quebras

Analisando o gráfico e posteriormente a Tabela 15, conclui-se que na estratégia da Rota Incompleta - 4 quebras, o AODV recebeu com sucesso 5% a mais que o DSR.

6ª Estratégia – Pacotes recebidos com sucesso com a Rota Incompleta – 5 quebras

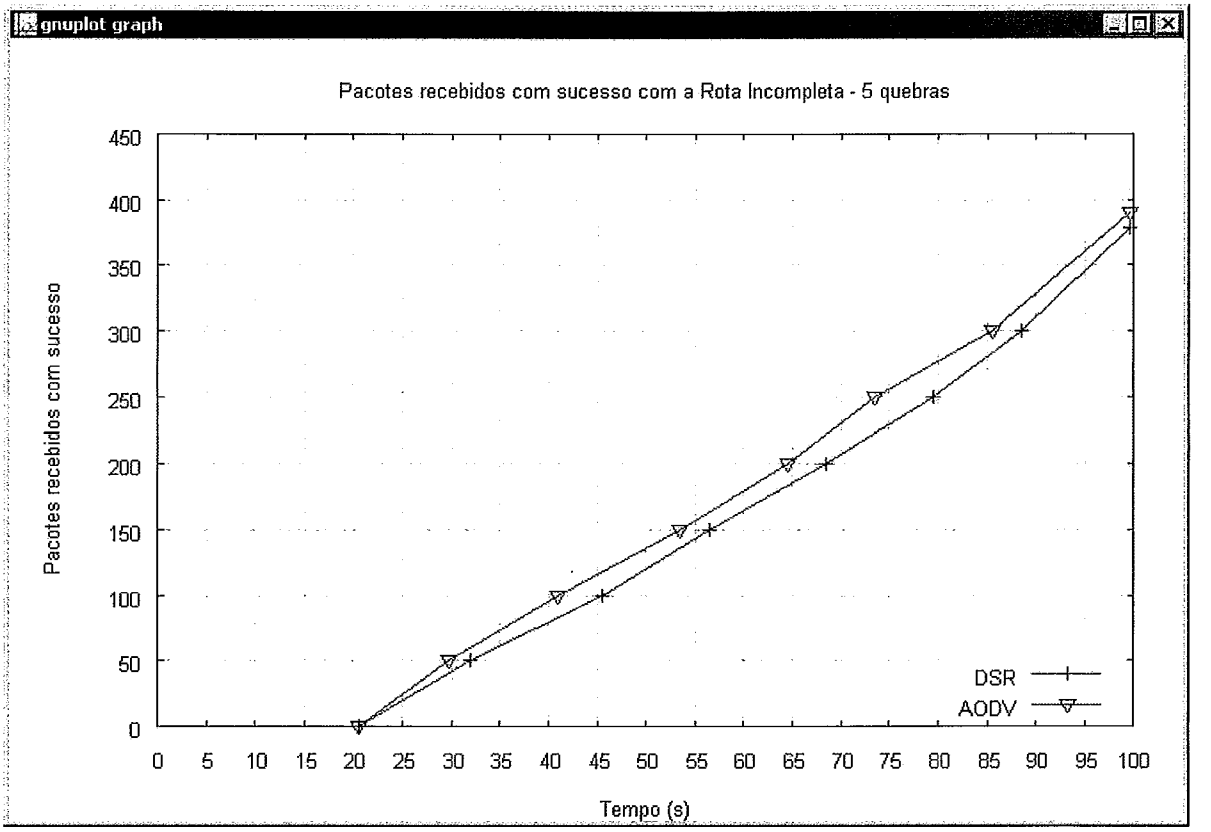


Figura 48 - Pacotes Recebidos com sucesso com a Rota Incompleta – 5 quebras

A Tabela 16 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,7	99,7
Pacotes Recebidos	390	378
Diferença Relativa (%)	3%	

Tabela 16 – Diferença Relativa dos Pacotes recebidos com sucesso com a Rota Incompleta - 5 quebras

Analisando o gráfico e posteriormente a Tabela 16, conclui-se que na estratégia da Rota Incompleta - 5 quebras, o AODV recebeu com sucesso 3% a mais que o DSR.

7ª Estratégia – Pacotes recebidos com sucesso – Caso a UMC deixe de funcionar

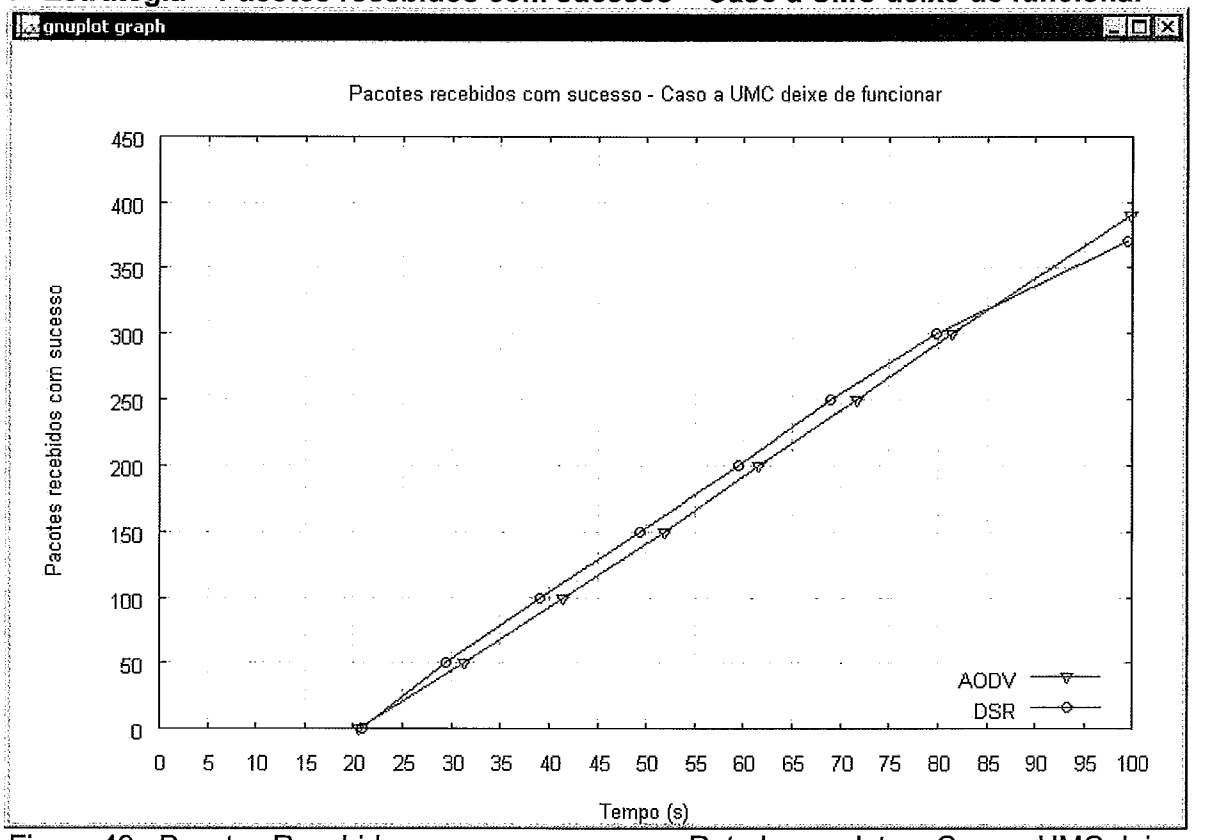


Figura 49 –Pacotes Recebidos com sucesso com a Rota Incompleta – Caso a UMC deixe de funcionar

A Tabela 17 mostra a diferença relativa entre os protocolos de acordo com o gráfico acima.

Protocolos	AODV	DSR
Tempo (s)	99,8	99,5
Pacotes Recebidos	390	370
Diferença Relativa (%)	5%	

Tabela 17 - Diferença Relativa dos Pacotes recebidos com sucesso com a Rota Completa - Caso a UMC deixe de funcionar

Analisando o gráfico e posteriormente a Tabela 17, conclui-se que na estratégia da Rota Completa - Caso a UMC deixe de funcionar, o AODV recebeu com sucesso 5 % a mais que o DSR.

Nas sete estratégias e com os dois protocolos de roteamento, a primeira comunicação é feita em 20.5s, logo, o primeiro pacote é enviado em 20.5s, analisando pelos gráficos acima: o primeio pacote é recebido com sucesso em 20.5s também, ou seja, não houve atraso em relação ao numero de pacotes enviados e o número de pacotes recebidos com sucesso.

Para analisar a quantidade dos pacotes enviados de ambos os protocolos, foi utilizado o critério do somatório dos pacotes perdidos com os pacotes recebidos com sucesso nas sete estratégias como seguem as Tabelas 18 e 19 do protocolo AODV e do DSR respectivamente.

Estratégias	**Rotas**	**Pacotes enviados**	**Pacotes perdidos**	**Pacotes recebidos com sucesso**
1ª Estratégia	Rota Completa	**726**	340	386
2ª Estratégia	Rota Incompleta - 1 quebra	**737**	350	387
3ª Estratégia	Rota Incompleta - 2 quebras	**741**	345	396
4ª Estratégia	Rota Incompleta - 3 quebras	**738**	343	395
5ª Estratégia	Rota Incompleta - 4 quebras	**731**	340	391
6ª Estratégia	Rota Incompleta - 5 quebras	**736**	346	390
7ª Estratégia	Rota Completa Caso a UMC falhe	**720**	330	390

Tabela 18 – Pacotes enviados no protocolo AODV

Com relação aos pacotes enviados no protocolo de roteamento DSR, segue a Tabela 19 abaixo.

Estratégias	Rotas	Pacotes enviados	Pacotes perdidos	Pacotes recebidos com sucesso
1ª Estratégia	Rota Completa	**738**	373	365
2ª Estratégia	Rota Incompleta - 1 quebra	**735**	375	360
3ª Estratégia	Rota Incompleta - 2 quebras	**761**	375	386
4ª Estratégia	Rota Incompleta - 3 quebras	**741**	361	380
5ª Estratégia	Rota Incompleta - 4 quebras	**745**	373	372
6ª Estratégia	Rota Incompleta - 5 quebras	**747**	369	378
7ª Estratégia	Rota Completa Caso a UMC falhe	**735**	365	370

Tabela 19 – Pacotes enviados no protocolo DSR

Pelas tabelas acima, nota-se que em ambos os protocolos de roteamento, foram recebidos com sucesso e perdidos quase 50% dos pacotes enviados.

O que poderia ter sido analisado seria o aumento da topologia de rede (aumento no número de dispositivos) ou diminui-la (retirada do número de dispositivos) ainda mais levando em consideração o alcance do *ZigBee* e as distâncias entre os postes. Porém, isso faz pouca diferença quanto ao número de pacotes enviados, perdidos e recebidos em relação as sete estratégias simuladas e analisadas anteriormente.

Há dois possíveis motivos para esse ocorrido, o primeiro é a não utilização do modo beacon, pois sem ele não há um sincronismo entre os dispositivos, ou seja, eles não estão preparados para receber as transmissões, isso acarreta em uma maior perda de pacotes e o segundo motivo é que foi programado no *script* em OTcl no NS-2 para as sete estratégias (mesmo com a topologia alterada) executarem seus procedimentos (*start* do coordenador, *start* dos roteadores e dos end-devices, busca pelos dispositivos na rede e transmissões de dados) até 100s. Ou seja, o NS-2 permanece "rodando" até completar o tempo de *stop time* (100s), por esse fato, os números de pacotes enviados dos dois protocolos são próximos e elevados. Para justificar esse fato, foi refeita a simulação apenas com a Rota Completa e com ambos os protocolos de roteamento com outros tempos de simulação: 50s, 150s e 300s. Analisa-se que quanto maior o tempo de simulação, maior a quantidade de pacotes enviados, perdidos e recebidos e maior a diferença de desempenho entre os dois protocolos, porém o número de pacotes recebidos com sucesso e perdidos também é quase 50% dos pacotes enviados.

4.5. Análise dos Resultados

Apesar do AODV e do DSR apresentarem o mesmo comportamento reativo, as diferenças nos mecanismos de cada protocolo podem conduzir a significativas diferenças de desempenho. Essas diferenças são analisadas de acordo com a retirada do número de dispositivos em cada estratégia e quantidade de pacotes perdidos ou recebidos em um determinado tempo.

Em todas as estratégias e nas duas situações (pacotes perdidos e recebidos com sucesso) o desempenho do AODV foi melhor do que o desempenho do DSR. Porém, o fato de ambos os protocolos terem perdidos mais ou menos pacotes e recebido mais ou menos pacotes, não implica na comunicação entre a UMC e as UM. Das duas maneiras, as comunicações foram feitas.

O motivo principal das perdas de pacotes ocorridas no DSR foi devido á falta de rotas para alguns destinos, ocasionados pela retirada de alguns dispositivos na topologia da rede. Em uma rede ad-hoc os nós atuam como roteadores, ou seja, trocam informações entre si para saber se um destino é alcançável ou não. Além disso, o grau de

comprometimento entre os nós é alto e o desempenho da rede depende de cada um dos nós. Em virtude disso, conforme os nós foram sendo retirados da rede, a topologia foi se modificando e assim foi prejudicando a comunicação dos nós que agem como roteadores. Além do mais, uma mudança na topologia ocasiona um aumento na freqüência da troca de mensagens de roteamento (RREQ, RREP, RERR), esse fato acarreta um congestionamento na rede, ocasionando a perda dos pacotes.

O protocolo DSR possui: roteamento pela fonte (somente o nó fonte é que conhece a rota completa até o nó destino), os nós intermediários não tem a iniciativa de se comunicarem entre si, sendo assim esses nós não atualizam seus cachês a todo o momento e além de tudo, o DSR possui múltiplas rotas para um mesmo destino. Por conta destes três motivos, ele pode enviar pacotes por rotas antigas armazenadas no seu *cachê de rotas* (rotas que não são mais válidas). Esse fato ocasiona um atraso no recebimento dos pacotes e assim um menor recebimento do número de pacotes.

O protocolo AODV obteve o melhor desempenho nesta topologia, pois é o que mais se assemelha ao funcionamento do SMRP. Ele utiliza uma Tabela de roteamento (a mesma usada pelo protocolo *ZigBee*), os nós trocam informações entre si para saber se um destino é alcançável ou não. No SMRP as UM também se comunicam entre si até chegarem a UMC.

No próximo capítulo serão apresentadas as conclusões obtidas com a realização deste trabalho.

CAPÍTULO 5

CONCLUSÃO

Para avaliar o desempenho dos protocolos de roteamento AODV e DSR utilizando o protocolo de comunicação *ZigBee* foi projetado e simulado um cenário que retrata a topologia atual de um Sistema de Medição para Redução de Perdas. Para isto, foi feito um levantamento de todas as características deste cenário, em manuais e com pessoas capacitadas de forma que sua apresentação fosse a mais próxima da realidade. O desenvolvimento deste cenário consistiu em fazer a representação dos nós e retirada de alguns dispositivos através de sete diferentes estratégias utilizando a ferramenta NAM.

Como neste cenário os dispositivos são postes de distribuição de energia elétrica e distam de 40m um do outro, as rotas para serem restabelecidas em caso de mais de uma quebra de enlace e caso essa quebra ocorra no nó centralizador, necessitará que algum técnico vá até o local e com um palm top reinicialize e normalize a rede.

Os protocolos que operam *on-demand*, que tenham a característica de Tabela de roteamento, que utilizem temporizadores e troca de informação entre os nós como o AODV, são indicados para este tipo de cenário. Já o DSR utiliza roteamento pela fonte e cachê em rota, seu uso é menos indicado para este tipo de topologia de rede.

De acordo com os resultados obtidos e analisados, concluímos que o AODV foi o que apresentou o melhor desempenho nas diversas simulações realizadas.

Dificuldades Encontradas

As principais dificuldades encontradas para realização do projeto foram:

- Entendimento do protocolo *ZigBee* pois são encontrados bibliografias na Internet que não condizem com o que esta relatado no manual dos padrões: 802.15.4 e *ZigBee Alliance* principalmente em relação a topologia de rede. Tem fabricantes que utilizam até diferentes nomenclaturas para identificar o mesmo tipo de topologia de rede;
- Entendimento da programação em Otcl, pela falta de referências bibliográficas encontradas na Internet sobre o uso do padrão utilizado no projeto (802.15.4);
- Instalação e configuração do *Network Simulator* (NS), que foi feito primeiramente utilizando o *Cygwin* (NS-2 *for Windows*) como o resultado deixou muito a desejar, posteriormente foi passado para o *Linux*;
- Análise e compreenção dos arquivos de *trace* (ferramenta de análise do NS-2) pois não há muitas referências bibliográficas sobre isso;
- Análise e compreenção do arquivo de traço utilizado no NAM (*Network Animator*) para que a topologia seja mostrada no final da simulação;
- A análise gráfica que a priori foi toda feita em *Java*, posteriormente é que foi analisada através do aplicativo *GNUPlot*;

Sugestões para Trabalhos Futuros

- Implementação deste trabalho em campo para serem feitas medições reais.
- Ampliação do local (2 quadras ou mais) causando assim um aumento da topologia de rede. Novas medições serão feitas para analisar o comportamento e o desempenho dos protocolos de roteamento para redes *ad-hoc* AODV e DSR.

Referências Bibliográficas

[1] GAMA, S.Z. et al, *Uma Nova Abordagem Tecnológica de Combate Às Perdas Comerciais,* XV Seminário Nacional de Distribuição de Energia Elétrica, SENDI 2002.

[2] Wikipédia, *Power Line Communication.* Disponível em < http://pt.wikipedia.org/wiki/PLC> Acesso: 21 jan. 2008.

[3] IEEE Std 802.15.4-2003, *IEEE Standard for Information technology, Telecommunications and information exchange between systems, Local and metropolitan area networks, Specific requirements. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*

[4] YITRAN COMMUNICATIONS Ltd. Disponível em< http://www.yitran.com/download1.htm > Acesso em 28 fev.2008

[5] *Bluethooth III,* SCELISUL UPDATE Seminário de Atualização Tecnológica ITEC, Out. 2004. Disponível em <www.itecsp.com.br/instituto/palestra/Apresentação%20Bluetooth%20Scelisul.ppt > Acesso em 11 fev. 2007

[6] MALAFAYA, Hugo; TOMÁS, Luís; SOUSA, João Paulo, *Sensorização sem fios sobre ZigBee e IEEE 802.15.4,* Faculdade de Engenharia da Universidade do Porto. Disponível em < http://www.deetc.isel.ipl.pt/jetc05/JETC05/Artigos/Electronica/Poster%20E/136.pdf > Acesso em: 22 mar. 2007

[7] Freescale Semiconductor. Disponível em< www.freescale.com/zigbee > Acesso em 28 fev.2008

[8] Zigbee Specification v1, *Zigbee Alliance*, 2005. Disponível em < http://www.zigbee.org > Acesso em 01 fev. 2007

[9] *AODV*. Disponível em< http://moment.cs.ucsb.edu/AODV/aodv.html> Acesso em 24 jul. 2007

[10] PERKINS, C. E. et.al, *Ad Hoc On-Demand Distance Vector (AODV) Routing. Request for Comments 3561*, Jul. 2003.

[11] BRAGA, Reinaldo B, *Dynamic Source Routing (DSR)*. GTA/UFRJ, Rio de Janeiro, RJ, Brasil. Disponível em <www.gta.ufrj.br/ensino/CPE825/2006/resumos/resumo-DSR-Reinaldo.pdf > Acesso em 26 jul.2007

[12] PEREIRA, Ivana Cardial de Miranda; PEDROZA, Aloysio de Castro, *Redes Móveis Ad-hoc aplicadas a Cenários Militares*. GTA/UFRJ, Rio de Janeiro, RJ, Brasil. Disponível em <www.gta.ufrj.br/ftp/gta/TechReports/PP04c.pdf.gz > Acesso em 03 dez. 2007

[13] MORAES, Igor Monteiro, *AODV*. GTA/UFRJ, Rio de Janeiro, RJ, Brasil. Disponível em < www.gta.ufrj.br/~rezende/cursos/eel879/trabalhos/aodv/introducao.html > Acesso em 05 jan.2008

[14] ZHENG, J., E LEE, M. J, *A Comprehensive Performance Study of IEEE 802.15.4. Sensor Network Operations*, IEEE Press, Wiley Interscience, Chapter 4, pp. 218-237,2006. Disponível em <http://www-ee.ccny.cuny.edu/zheng/pub/ > Acesso em 10 ago. 2007

[15] ZHENG, J., E LEE, M. J, *Will IEEE 802.15.4 Make Ubiquitous Networking a Reality?* A Discution on a Potencial Low Power Bit Rate. *Standard IEEE Communications, Magazine,* jun. 2004, pp. 140–146. Disponível em <http://www-ee.ccny.cuny.edu/zheng/pub/ > Acesso em 10 ago. 2007

[16] SANTOS, Sergio Torres, *Redes de Sensores sem fio em monitoramento e controle.* Dissertação de M.Sc. , COPPE/UFRJ, Rio de Janeiro, RJ, Brasil. Disponível em < www.gta.ufrj.br/ftp/gta/TechReports/Sergio07/Sergio07.pdf> Acesso em 18 dez.2007

[17] *IEEE 802.15 WPAN™ Task Group 4 (TG4).* Disponível em <http://www.ieee802.org/15/pub/TG4.html> Acesso em 28 abril. 2007

[18] BRANDÃO, Carlos Eduardo et al, *WLAN -Wireless Local Area Network (Redes Locais sem fio).* Universidade Federal da Bahia.
Disponível em< http://twiki.im.ufba.br/pub/MAT060/WLAN/UFBA_Trabalho_WLAN.pdf> Acesso em 23 mar. 2007

[19] GUIMARÃES, Dayani Adionel, *Sistema de Comunicação Móvel de Terceira Geração, WirelessBR.*
Disponível em
<http://www.wirelessbrasil.org/wirelessbr/colaboradores/dayani/3g_08.html> Acesso em 24 mar.2007

[20] NAVARRO, Marcos Cunha, *Protocolos de Roteamento para Redes Ad-Hoc*, 2005.
Disponível em
<http:www.twiki.dcc.ufba.br/pub/MAT057/TrabalhosSemestre20051/Marcos_Navarro_Apresentacao.pdf > Acesso em 20 dez.2006

[21] Microsoft TechNet. *Visão Geral sobre Redes sem fio,* 2005. Disponível em <http://technet2.microsoft.com/WindowsServer/pt-BR/Library/f2552467-f693-4c14-b421-49cb2491bb361046.mspx?mfr=true > Acesso em 23 fev. 2007

[22] MESSIAS, Antonio Rogério, *Controle remoto e aquisição de dados via XBee/ZigBee (IEEE 802.15.4).*Disponível em <http://www.rogercom.com/ZigBee/ZigBee.htm> Acesso em 07 abril. 2007

[23] AREZIO, Claudia, *ZigBee*, PUC/PR. Disponível em
<www.ppgia.pucpr.br/~jamhour/Download/pub/Mestrado%202006/zigbee_Claudia.ppt>
Acesso em 06 dez.2006

[24] TEIXEIRA, André, *Módulos de Comunicação Wireless para Sensores.* Faculdade de Engenharia da Universidade do Porto, 2007.
Disponível em <http://www.rogercom.com/ZigBee/ZigBee.htm> Acesso em 28 mar. 2007

[25] *Wireless IP.* Disponível em <http://www.wirelessip.com.br/wirelessip/faqs> Acesso em 25 abril. 2007

[26] Wikipédia. *Zigbee.* Disponível em < http://pt.wikipedia.org/wiki/ZigBee> Acesso em 25 abril. 2007

[27] PRAKASH, Vaddina; MARANDIN, Dimitri, *Adaptive Backoff Exponent Algorithm for Zigbee (IEEE 802.15.4).* Department of Electrical Engineering and Information Technology, Technische Universität Dresden, Dresden, Germany. Disponível em< http://www.vaddina.com/pages/Zigbee/snav_1_0.php> Acesso em 04 maio. 2007

[28] SILVA, Ana Cristina Benso, *Roteamento.* Disponível em <www.inf.pucrs.br/~benso/redes601/2004_2/Roteamento.ppt > Acesso em 08 ago. 2007

[29] ASCAMA, Hector Dave Orrillo et. al, *Avaliação da Segurança para Gerenciamento na troca de informações em uma rede sem fio.* Laboratório de Sistemas Integráveis: Universidade de São Paulo, São Paulo.
Disponível em < http://www.pad.lsi.usp.br/humanlab/trabalhos/SELASI_2005.pdf >
Acesso em 10 jul. 2007

[30] PERKINS, C. E. et. al, *Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks*, IEEE *Personal Communications.*

[31] COUTINHO, Mauro Margalho, *Network Simulator - Guia Básico para Iniciantes*, 2007.
Disponível em<www.cin.ufpe.br/~sfd/universo/sim/nsr1.pdf> Acesso em 08 ago.2007

[32] *High Speed Network Technologies.*
Disponível em <http://w3.antd.nist.gov/Hsntg/prd_atm-sim.html> Acesso em 10 maio. 2007

[33] Relatório Técnico. *TAQUARA - Tecnologia, Aplicações e Qualidade de Serviço em Redes Avançadas no Projeto GIGA.* GTA/UFRJ, Rio de Janeiro, RJ, Brasil, 2007. Disponível em < www.gta.ufrj.br/taquara/publicacao/relatorios/1o-relatorio-giga.pdf > Acesso em 12 dez. 2007

[34] FRANCA, Alexandre, *Tcl/Tk -Programação Linux*, 2 ed., 2005. Disponível em <http://www.brasport.com.br/index.php?Escolha=8&Livro=L00155> Acesso em 12 maio.2007

[35] BASTOS, Eduardo Leivas, *Simulação do modelo RCSP em redes ATM utilizando escalonamento EDD e realimentação.* Disponível em <http://www.inf.ufrgs.br/pos/SemanaAcademica/Semana2000/EduardoBastos/> Acesso em 11 maio.2007

[36] *NS-2, Contributed Code.* Disponível em < http://nsnam.isi.edu/nsnam/index.php/Contributed_Code#Differentiated_Services> Acesso em 13 maio.2007

[37] CARVALHO, Guilherme Soares, *Implementação e Avaliação de Serviços Diferenciados (DIFFSERV) para provimento de qualidade de serviço (QoS).* Universidade Estadual de Montes Claros, Minas Gerais, MG, 2004. Disponível em < www.ccet.unimontes.br/arquivos/monografias/14.pdf> Acesso em 24 out. 2007

[38] *Low Rate Wireless Personal Area Networks (LR-WPANs): NS2 Simulation Platform.* Disponível em <http://ees2cy.engr.ccny.cuny.edu/zheng/pub/> Acesso em 14 set. 2007.

[39] JARGAS, Aurélio Marinho, *Cygwin - O Linux no Windows*, 2003. Disponível em < http://aurelio.net/cygwin/rdl/ > Acesso em 07 ago.2007

[40] *NS-2 Trace Formats.* Disponível em <http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats#AODV_Trace_Formats> Acesso em 09 ago.2007

[41] *Crimson Editor.* Disponível em< http://www.crimsoneditor.com > Acesso em 01 jul. 2007.

[42] *Nam: Network Animator.* Disponível em <http://www.isi.edu/nsnam/nam/> Acesso em 10 out. 2007

[43] MARIZ, Dênio; KAMIENSKI, Carlos, *Gerenciamento e Avaliação de Desempenho de Redes: NS Network Simulator.* Disponível em < http://www.arquivos.coinfo.cefetpb.edu.br/~denio/disciplinas/slides/gad/04b-SIMULACAO%20-%20NS.pdf > Acesso em 25 ago.2007

[44] COUTINHO, Mauro Margalho et.al, *Avaliação de Desempenho de redes ad-hoc em um cenário típico da região Amazônica.* SBrT 2003, XX Simpósio Brasileiro de Telecomunicações, Universidade Federal do Pará. Disponível em < www.lprad.ufpa.br/~margalho/wdeec/SBRTAH03.pdf > Acesso 09 set.2007

[45] *Xgraph.* Disponível em *<http://jean-luc.aei-potsdam.mpg.de/Codes/xgraph/> Acesso em 10 nov.2007*

[46] *Trace graph: Network Simulator NS-2 trace files analyzer.* Disponível em <http://www.tracegraph.com> Acesso 10 nov. 2007

[47] *GNUPlot.* Disponível em <http://www.cpgei.cefetpr.br/~ekalin/gnuplot.html> Acesso em 12 nov. 2007

[48] *Introdução ao uso do aplicativo GNUPlot.* Disponível em <http://www2.prudente.unesp.br/dcartog/galo/gnuplot/gnu_tutorial.htm> Acesso em 12 nov. 2007

[49] *GNUPlot.* Disponível em <http://www.gnuplot.info> Acesso em 13 nov 2007

[50] *Network Simulator for Windows.* Disponível em<http://ce.sharif.edu/~m_amiri/project/networksimulator1/index.htm#NSBinaryForWindows32bitDwLnk > Acesso em 18 set. 2007

[51] *Cygwin.* Disponível em < http://www.cygwin.com/ > Acesso em 18 set.2007

A – Código Fonte das Aplicações Utilizadas

A.1 Scripts utilizando o aplicativo GNUPlot

```
# Exemplo de Visualização dos gráficos
# Pacotes Perdidos
# Cristiane Amaral / COPPE
# Aplicativo: gnuplot

# 1º Situação: Rota Completa
unset mouse
reset
set key right bottom
set grid
set yrange [0:450]
set xrange [0:100]
set xtics 5
set title "Pacotes perdidos com a Rota Completa"
set xlabel "Tempo (s)"
set ylabel "Pacotes Perdidos"
set key outside below
plot "Pacotes Perdidos2.dat" using ($3):($4) t"DSR" with linespoints 3 3
rep "Pacotes Perdidos2.dat" using ($1):($2) t"AODV" with linespoints 4 4
pause -1 "Fecha?"

#2º Situação: Rota Incompleta - 1 quebra
unset mouse
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes perdidos com a Rota Incompleta - 1 quebra"
set xlabel "Tempo (s)"
set ylabel "Pacotes perdidos"
set key outside below
plot "Pacotes Perdidos2.dat" using ($7):($8) t"DSR" with linespoints 5 5
rep "Pacotes Perdidos2.dat" using ($5):($6) t"AODV" with linespoints 15 15
pause -1 "Fecha?"

# 3º Situação: Rota Incompleta - 2 quebras
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes perdidos com a Rota Incompleta - 2 quebras"
set xlabel "Tempo (s)"
set ylabel "Pacotes perdidos"
plot "Pacotes Perdidos2.dat" using ($11):($12) t"DSR" with linespoints 16 16
```

```
rep "Pacotes Perdidos2.dat" using ($9):($10) t"AODV" with linespoints 10 10
pause -1 "Fecha?"

# 4º Situação: Rota Incompleta - 3 quebras
unset mouse
reset
set key right bottom
set grid
set yrange [0:450]
set xrange [0:100]
set xtics 5
set title "Pacotes perdidos com a Rota Incompleta - 3 quebras"
set xlabel "Tempo (s)"
set ylabel "Pacotes Perdidos"
set key outside below
plot "Pacotes Perdidos1.dat" using ($15):($16) t"DSR" with linespoints 3 3
rep "Pacotes Perdidos1.dat" using ($13):($14) t"AODV" with linespoints 4 4
pause -1 "Fecha?"

#5º Situação: Rota Incompleta - 4 quebras
unset mouse
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes perdidos com a Rota Incompleta -4 quebra"
set xlabel "Tempo (s)"
set ylabel "Pacotes perdidos"
set key outside below
plot "Pacotes Perdidos1.dat" using ($19):($20) t"DSR" with linespoints 5 5
rep "Pacotes Perdidos1.dat" using ($17):($18) t"AODV" with linespoints 15 15
pause -1 "Fecha?"

# 6º Situação: Rota Incompleta - 5 quebras
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes perdidos com a Rota Incompleta - 5 quebras"
set xlabel "Tempo (s)"
set ylabel "Pacotes perdidos"
plot "Pacotes Perdidos1.dat" using ($21):($22) t"DSR" with linespoints 16 16
rep "Pacotes Perdidos1.dat" using ($23):($24) t"AODV" with linespoints 10 10
pause -1 "Fecha?"


# 7º Situação: Rota Incompleta - Com Palm Top
```

```
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes perdidos - Caso a UMC deixe de funcionar"
set xlabel "Tempo (s)"
set ylabel "Pacotes perdidos"
plot "Pacotes Perdidos3.dat" using ($25):($26) t"AODV" with linespoints 10 10
rep "Pacotes Perdidos3.dat" using ($27):($28) t"DSR" with linespoints 8 8
pause -1 "Fecha?"
```

Script para a quantidade de pacotes recebidos com sucesso utilizando as sete estratégias:

```
# Exemplo de Visualização dos gráficos
# Pacotes Recebidos com sucesso
# Cristiane Amaral / COPPE
# Aplicativo: gnuplot

# 1º Situação: Rota Completa
unset mouse
reset
set key right bottom
set grid
set yrange [0:450]
set xrange [0:100]
set xtics 5
set title "Pacotes recebidos com sucesso com a Rota Completa"
set xlabel "Tempo (s)"
set ylabel "Pacotes recebidos com sucesso"
set key outside below
plot "Pacotes recebidos1.dat" using ($3):($4) t"DSR" with linespoints 3 3
rep "Pacotes recebidos1.dat" using ($1):($2) t"AODV" with linespoints 4 4
pause -1 "Fecha?"

#2º Situação: Rota recebidos com sucesso - 1 quebra
unset mouse
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes recebidos com sucesso com a Rota Incompleta - 1 quebra"
set xlabel "Tempo (s)"
set ylabel "Pacotes recebidos com sucesso"
set key outside below
plot "Pacotes recebidos1.dat" using ($7):($8) t"DSR" with linespoints 5 5
```

```
 rep "Pacotes recebidos1.dat" using ($5):($6) t"AODV" with linespoints 15 15
pause -1 "Fecha?"

# 3º Situação: Rota Incompleta - 2 quebras
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes recebidos com sucesso com a Rota Incompleta - 2 quebras"
set xlabel "Tempo (s)"
set ylabel "Pacotes recebidos com sucesso"
plot "Pacotes recebidos1.dat" using ($11):($12) t"DSR" with linespoints 16 16
rep "Pacotes recebidos1.dat" using ($9):($10) t"AODV" with linespoints 10 10
pause -1 "Fecha?"

# 4º Situação: Rota Incompleta - 3 quebras
unset mouse
reset
set key right bottom
set grid
set yrange [0:450]
set xrange [0:100]
set xtics 5
set title "Pacotes recebidos com sucesso com a Rota Incompleta - 3 quebras"
set xlabel "Tempo (s)"
set ylabel "Pacotes recebidos com sucesso"
set key outside below
plot "Pacotes recebidos2.dat" using ($3):($4) t"DSR" with linespoints 3 3
rep "Pacotes recebidos2.dat" using ($1):($2) t"AODV" with linespoints 4 4
pause -1 "Fecha?"

#5º Situação: Rota Incompleta - 4 quebras
unset mouse
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes recebidos com sucesso com a Rota Incompleta - 4 quebra"
set xlabel "Tempo (s)"
set ylabel "Pacotes recebidos com sucesso"
set key outside below
plot "Pacotes recebidos2.dat" using ($7):($8) t"DSR" with linespoints 5 5
rep "Pacotes recebidos2.dat" using ($5):($6) t"AODV" with linespoints 15 15
pause -1 "Fecha?"

# 6º Situação: Rota Incompleta - 5 quebras
reset
```

```
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes recebidos com sucesso com a Rota Incompleta - 5 quebras"
set xlabel "Tempo (s)"
set ylabel "Pacotes recebidos com sucesso"
plot "Pacotes recebidos2.dat" using ($11):($12) t"DSR" with linespoints 16 16
rep "Pacotes recebidos2.dat" using ($9):($10) t"AODV" with linespoints 10 10
pause -1 "Fecha?"

# 7º Situação: Rota Incompleta - Com Palm Top
reset
set key right bottom
set grid
set xrange [0:100]
set yrange [0:450]
set xtics 5
set title "Pacotes recebidos com sucesso - Caso a UMC deixe de funcionar"
set xlabel "Tempo (s)"
set ylabel "Pacotes recebidos com sucesso"
plot "Pacotes recebidos3.dat" using ($1):($2) t"AODV" with linespoints 10 10
rep "Pacotes recebidos3.dat" using ($3):($4) t"DSR" with linespoints 21 21
pause -1 "Fecha?"
```

A.2 Script em OTcl

```
# ======================================================================
# Define options
#
# ======================================================================
set val(chan)           Channel/WirelessChannel
# Tipo de Canal: Wireless
set val(prop)           Propagation/TwoRayGround
# Modelo de propagação de radio: Two Ray Ground
set val(netif)          Phy/WirelessPhy/802_15_4
# Tipo de Interface de rede
set val(mac)            Mac/802_15_4
# Netif e mac já dizem que estamos trabalhando com o padrao 802_15_4, ou seja, o padrão do ZigBee
set val(ifq)            Queue/DropTail/PriQueue
# Interface do tipo "queue" / Drop Tail
# Fila Drop Tail corresponde ao método FIFO
set val(ll)             LL
# LL é chamado de LINK LAYER, é um tipo camada de enlace utilizado para o padrão wireless
set val(ant)            Antenna/OmniAntenna
# Modelo da antenna: OmniAntenna
set val(ifqlen)         50
# Número máximo de pacotes na interface de fila
set val(nn)             49
# Número de nós movies (Rota Completa)
set val(rp)             AODV
# Protocolo de Roteamento utilizado neste programa: AODV
# Nessa linha é inserido o protocolo de roteamento DSR
set val(x)              80
set val(y)              80
# val (x) val (y) é a dimensao da rede, no caso, ela é 80x80
set val(cp)             "/root/ns-allinone-2.31/ns-2.31/wpan"
# Local do Linux onde o padrão 802.15.4 é encontrado
set val(dst)            3
set val(dstit)          1
set val(tr)             wpan_teste5.tr
# O ns vai gerar um Trace File (Tr) com esse nome: wpan_teste5.tr
set val(nam)            wpan_teste5.nam
#O ns vai gerar um arquivo de nam com esse nome: wpan_teste5.nam
set val(nodeDown)       yes
set val(errRate)        0
# Porcentagem de erro
set val(traffic)        mix
# Tráfico Misto (mix), ou seja, o tráfico é feito de três maneiras diferentes: cbr/poisson/ftp
set val(trInterval)     0.2             ;# in seconds

# Parameters used for traffic generation
set val(nn)             49
# Número de nós móveis
set val(starttime)      20.5
# Inicio da comunicação: 20.5s
set stopTime            100
```

```
# Final da simulação: 100s

        # read command line arguments
        proc getCmdArgu {argc argv} {
            global val
            for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue
                set name [string range $arg 1 end]
                set val($name) [lindex $argv [expr $i+1]]
            }
        }
        getCmdArgu $argc $argv
        set appTime1        20.5 ;# in seconds
# Apptime1= Primeira comunicação (UMC Nó 0 com a UM6- Cons28) em 20.5s
        set appTime2        20.8 ;# in seconds
# Apptime2= Segunda comunicação (UMC Nó 0 com a UM15- Cons43) em 20.8s
        set stopTime        100  ;# in seconds
# StopTime = Final da simulação em 100.0s, ou seja, ele vai ficar enviado, recebendo e
perdendo pacotes

        # Initialize Global Variables
        set ns_             [new Simulator]
# Novo simulador
        set tracefd     [open ./wpan_teste5.tr w]
# Abrir o arquivo wpan_teste5.tr automaticamente
        $ns_ trace-all $tracefd
        if { "$val(nam)" == "wpan_teste5.nam" } {
            set namtrace     [open ./$val(nam) w]
            $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
        }
# Abrir o arquivo wpan_teste5.nam automaticamente mantendo a dimensão da rede 80x80

        $ns_ puts-nam-traceall {# nam4wpan #} ;
# Informa ao NAM e ao trace file que a rede é WPAN
        Mac/802_15_4 wpanCmd verbose on
        Mac/802_15_4 wpanNam namStatus on
        #Mac/802_15_4 wpanNam ColFlashClr gold
# default = gold
        #Mac/802_15_4 wpanNam NodeFailClr grey
# default = grey

# Usando o modelo de propagação de radio 'TwoRayGround'
        # For model 'TwoRayGround'
        set dist(5m)  7.69113e-06
        set dist(9m)  2.37381e-06
        set dist(10m) 1.92278e-06
        set dist(11m) 1.58908e-06
        set dist(12m) 1.33527e-06
        set dist(13m) 1.13774e-06
        set dist(14m) 9.81011e-07
        set dist(15m) 8.54570e-07
        set dist(16m) 7.51087e-07
        set dist(20m) 4.80696e-07
        set dist(25m) 3.07645e-07
```

```
        set dist(30m) 2.13643e-07
        set dist(35m) 1.56962e-07
        set dist(40m) 1.20174e-07
        Phy/WirelessPhy set CSThresh_ $dist(9m)
        Phy/WirelessPhy set RXThresh_ $dist(9m)

        # set up topography object
        set topo      [new Topography]
        $topo load_flatgrid $val(x) $val(y)
# Ele esta pegando a topologia 80x80 que foi programada acima

        # Create God
# Cria o God(General Operations Director)

# O God é um objeto usado para armazenar informações globais sobre o estado do
ambiente, rede e nós. Um observador onipresente, mas que não é conhecido por nenhum
dos participantes da simulação
        set god_ [create-god $val(nn)]
        set chan_1_ [new $val(chan)]
# Cria o canal
# Esse parâmetro é necessário para o funcionamento correto do programa


# ConFiguração dos nós
        # configure node
        $ns_ node-config -adhocRouting $val(rp) \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -topoInstance $topo \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace ON \
                         -movementTrace ON \
                    #-energyModel "EnergyModel" \
                    #-initialEnergy 1 \
                    #-rxPower 0.3 \
                    #-txPower 0.3 \
                         -channel $chan_1_
# Repete todos os parâmetros do cabeçalho, comenta a energia, ou seja, a energia nao esta
sendo medida e insere ON em todos os parâmetros que queremos analisar

        # insert error model
        if {$val(errRate) != 0} {
                $ns_ node-config -errProc UniformErr
        }

        proc UniformErr {} {
             set err [new ErrorModel]
             $err unit pkt
             $err set rate_ [expr $val(errRate) / 100.0]
             #$err drop-target [new Agent/Null]
```

```
        return $err
    }
# Define a posição inicial do nó no NAM
    for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0          ;# disable random motion
    }

# Esses números são vistos no arquivo de traço do NAM. Isso é feito na própria tela do NAM, antes de programar.
# O z é sempre zero pois o cenário é bidimensional e não tridimensional.
    # --------- Criacao do Cenario ----------
    puts " Cenario ....."
    puts " "
    # nodes: 49, pause: 2.00, max speed: 2.00  max x = 80.00, max y: 80.00
# Número total de nós =49 / Velocidade máxima= 2s/ Dimensão x e y = 80.00
# Posição que todos os nós ocupam em X e em Y no arquivo de traço na tela da NAM.
    $node_(0) set X_ 597.5
    $node_(0) set Y_ 535.5
    $node_(0) set Z_ 0.000000000000
    $node_(1) set X_ 650.8
    $node_(1) set Y_ 532.2
    $node_(1) set Z_ 0.000000000000
    $node_(2) set X_ 627.1
    $node_(2) set Y_ 538.0
    $node_(2) set Z_ 0.000000000000
    $node_(3) set X_ 610.9
    $node_(3) set Y_ 549.3
    $node_(3) set Z_ 0.000000000000
    $node_(4) set X_ 611.1
    $node_(4) set Y_ 561.7
    $node_(4) set Z_ 0.000000000000
    $node_(5) set X_ 611.4
    $node_(5) set Y_ 574.1
    $node_(5) set Z_ 0.000000000000
    $node_(6) set X_ 611.5
    $node_(6) set Y_ 586.0
    $node_(6) set Z_ 0.000000000000
    $node_(7) set X_ 611.8
    $node_(7) set Y_ 597.4
    $node_(7) set Z_ 0.000000000000
    $node_(8) set X_ 611.6
    $node_(8) set Y_ 610.1
    $node_(8) set Z_ 0.000000000000
    $node_(9) set X_ 622.1
    $node_(9) set Y_ 625.6
    $node_(9) set Z_ 0.000000000000
    $node_(10) set X_ 603.3
    $node_(10) set Y_ 625.2
    $node_(10) set Z_ 0.000000000000
    $node_(11) set X_ 579.2
    $node_(11) set Y_ 625.0
    $node_(11) set Z_ 0.000000000000
    $node_(12) set X_ 549.0
    $node_(12) set Y_ 624.7
    $node_(12) set Z_ 0.000000000000
```

```
$node_(13) set X_ 565.7
$node_(13) set Y_ 612.9
$node_(13) set Z_ 0.000000000000
$node_(14) set X_ 565.4
$node_(14) set Y_ 600.8
$node_(14) set Z_ 0.000000000000
$node_(15) set X_ 564.8
$node_(15) set Y_ 588.8
$node_(15) set Z_ 0.000000000000
$node_(16) set X_ 565.4
$node_(16) set Y_ 576.5
$node_(16) set Z_ 0.000000000000
$node_(17) set X_ 564.5
$node_(17) set Y_ 564.4
$node_(17) set Z_ 0.000000000000
$node_(18) set X_ 564.6
$node_(18) set Y_ 550.0
$node_(18) set Z_ 0.000000000000
$node_(19) set X_ 543.9
$node_(19) set Y_ 537.7
$node_(19) set Z_ 0.000000000000
$node_(20) set X_ 563.4
$node_(20) set Y_ 528.0
$node_(20) set Z_ 0.000000000000
$node_(21) set X_ 572.4
$node_(21) set Y_ 518.5
$node_(21) set Z_ 0.000000000000
$node_(22) set X_ 581.2
$node_(22) set Y_ 507.2
$node_(22) set Z_ 0.000000000000
$node_(23) set X_ 583.3
$node_(23) set Y_ 536.1
$node_(23) set Z_ 0.000000000000
$node_(24) set X_ 610.5
$node_(24) set Y_ 521.5
$node_(24) set Z_ 0.000000000000
$node_(25) set X_ 633.2
$node_(25) set Y_ 625.4
$node_(25) set Z_ 0.000000000000
$node_(26) set X_ 622.5
$node_(26) set Y_ 610.1
$node_(26) set Z_ 0.000000000000
$node_(27) set X_ 622.5
$node_(27) set Y_ 598.0
$node_(27) set Z_ 0.000000000000
$node_(28) set X_ 622.0
$node_(28) set Y_ 586.2
$node_(28) set Z_ 0.000000000000
$node_(29) set X_ 621.1
$node_(29) set Y_ 574.2
$node_(29) set Z_ 0.000000000000
$node_(30) set X_ 621.6
$node_(30) set Y_ 561.9
$node_(30) set Z_ 0.000000000000
$node_(31) set X_ 621.1
$node_(31) set Y_ 548.9
```

```
$node_(31) set Z_ 0.000000000000
$node_(32) set X_ 637.6
$node_(32) set Y_ 537.7
$node_(32) set Z_ 0.000000000000
$node_(33) set X_ 661.4
$node_(33) set Y_ 537.8
$node_(33) set Z_ 0.000000000000
$node_(34) set X_ 620.9
$node_(34) set Y_ 521.1
$node_(34) set Z_ 0.000000000000
$node_(35) set X_ 572.9
$node_(35) set Y_ 536.1
$node_(35) set Z_ 0.000000000000
$node_(36) set X_ 533.5
$node_(36) set Y_ 538.4
$node_(36) set Z_ 0.000000000000
$node_(37) set X_ 554.1
$node_(37) set Y_ 525.4
$node_(37) set Z_ 0.000000000000
$node_(38) set X_ 562.7
$node_(38) set Y_ 514.1
$node_(38) set Z_ 0.000000000000
$node_(39) set X_ 571.7
$node_(39) set Y_ 503.4
$node_(39) set Z_ 0.000000000000
$node_(40) set X_ 553.7
$node_(40) set Y_ 550.0
$node_(40) set Z_ 0.000000000000
$node_(41) set X_ 553.9
$node_(41) set Y_ 564.9
$node_(41) set Z_ 0.000000000000
$node_(42) set X_ 554.8
$node_(42) set Y_ 576.9
$node_(42) set Z_ 0.000000000000
$node_(43) set X_ 554.1
$node_(43) set Y_ 589.4
$node_(43) set Z_ 0.000000000000
$node_(44) set X_ 554.8
$node_(44) set Y_ 600.6
$node_(44) set Z_ 0.000000000000
$node_(45) set X_ 555.2
$node_(45) set Y_ 613.4
$node_(45) set Z_ 0.000000000000
$node_(46) set X_ 538.1
$node_(46) set Y_ 624.9
$node_(46) set Z_ 0.000000000000
$node_(47) set X_ 568.3
$node_(47) set Y_ 624.9
$node_(47) set Z_ 0.000000000000
$node_(48) set X_ 593.1
$node_(48) set Y_ 625.9
$node_(48) set Z_ 0.000000000000

# Definição de quem é coordenador e quem são os dipositivos.
# O coordenador (UMC) é sempre o nó zero pois é ele quem comanda a rede
# Os nós 29, 48, 43 e 35 correspondem ao nome das ruas do estudo de caso
```

```
# sscs startDevice 1 - são os FFDs da topologia de rede (Unidades de Medição)
# sscs startDevice 0 - são os RFDs da topologia de rede (Consumidores)
        # --------- Criacao da topologia da rede (inicio) ----------
        puts " Criando Topologia da Rede ....."
        puts " "
        $ns_ at 0.0     "$node_(0) NodeLabel PAN Coord"
        $ns_ at 0.0     "$node_(0) sscs startPANCoord 1"
        $ns_ at 0.5     "$node_(0) NodeLabel Rota com AODV"
        $ns_ at 0.5     "$node_(29) NodeLabel Rua Artur Napoleao Lebre"
        $ns_ at 0.5     "$node_(48) NodeLabel Avenida dos Imigrantes"
        $ns_ at 0.5     "$node_(43) NodeLabel Rua Major Fernandes GS Brejense"
        $ns_ at 0.5     "$node_(35) NodeLabel Rua Frederico Frota"
        $ns_ at 0.5     "$node_(1) NodeLabel RotaR2"
        $ns_ at 0.5     "$node_(1) sscs startDevice 1"
        $ns_ at 0.5     "$node_(2) NodeLabel RotaR2"
        $ns_ at 1.5     "$node_(2) sscs startDevice 1"    ;
# As UM e o coordenador estão programados como FFD=1
        $ns_ at 0.5     "$node_(3) NodeLabel RotaR3"
        $ns_ at 2.5     "$node_(3) sscs startDevice 1"
        $ns_ at 0.5     "$node_(4) NodeLabel RotaR3"
        $ns_ at 3.5     "$node_(4) sscs startDevice 1"
        $ns_ at 0.5     "$node_(5) NodeLabel RotaR3"
        $ns_ at 4.5     "$node_(5) sscs startDevice 1"
        $ns_ at 0.5     "$node_(6) NodeLabel RotaR3"
        $ns_ at 5.5     "$node_(6) sscs startDevice 1"
        $ns_ at 0.5     "$node_(7) NodeLabel RotaR3"
        $ns_ at 3.5     "$node_(7) sscs startDevice 1"
        $ns_ at 0.5     "$node_(8) NodeLabel RotaR3"
        $ns_ at 4.0     "$node_(8) sscs startDevice 1"
        $ns_ at 0.5     "$node_(9) NodeLabel RotaR3"
        $ns_ at 4.5     "$node_(9) sscs startDevice 1"
        $ns_ at 0.5     "$node_(10) NodeLabel RotaR3"
        $ns_ at 5.0     "$node_(10) sscs startDevice 1"
        $ns_ at 0.5     "$node_(11) NodeLabel RotaR4"
        $ns_ at 5.5     "$node_(11) sscs startDevice 1"
        $ns_ at 0.5     "$node_(12) NodeLabel RotaR4"
        $ns_ at 6.0     "$node_(12) sscs startDevice 1"
        $ns_ at 0.5     "$node_(13) NodeLabel RotaR4"
        $ns_ at 6.5     "$node_(13) sscs startDevice 1"
        $ns_ at 0.5     "$node_(14) NodeLabel RotaR4"
        $ns_ at 7.0     "$node_(14) sscs startDevice 1"
        $ns_ at 0.5     "$node_(15) NodeLabel RotaR4"
        $ns_ at 7.5     "$node_(15) sscs startDevice 1"
        $ns_ at 0.5     "$node_(16) NodeLabel RotaR4"
        $ns_ at 8.0     "$node_(16) sscs startDevice 1"
        $ns_ at 0.5     "$node_(17) NodeLabel RotaR4"
        $ns_ at 8.5     "$node_(17) sscs startDevice 1"
        $ns_ at 0.5     "$node_(18) NodeLabel RotaR4"
        $ns_ at 9.0     "$node_(18) sscs startDevice 1"
        $ns_ at 0.5     "$node_(19) NodeLabel RotaR5"
        $ns_ at 9.5     "$node_(19) sscs startDevice 1"
        $ns_ at 0.5     "$node_(20) NodeLabel RotaR6"
        $ns_ at 10.0    "$node_(20) sscs startDevice 1"
        $ns_ at 0.5     "$node_(21) NodeLabel RotaR6"
        $ns_ at 10.5    "$node_(21) sscs startDevice 1"
        $ns_ at 0.5     "$node_(22) NodeLabel RotaR6"
```

```
        $ns_ at 11.0    "$node_(22) sscs startDevice 1"
        $ns_ at 0.5     "$node_(23) NodeLabel RotaR6"
        $ns_ at 0.5     "$node_(23) NodeLabel RotaR5"
        $ns_ at 0.5     "$node_(23) NodeLabel RotaR4"
        $ns_ at 11.5    "$node_(23) sscs startDevice 1"
        $ns_ at 0.5     "$node_(24) NodeLabel RotaR1"
        $ns_ at 12.0    "$node_(24) sscs startDevice 1"
        $ns_ at 0.5     "$node_(25) NodeLabel ConsumRotaR3"
        $ns_ at 12.5    "$node_(25) sscs startDevice 0"    ;# 0 -->
# RFD=0 → Consumidores só se comunicam com o medidor (UM) deles
        $ns_ at 0.5     "$node_(26) NodeLabel ConsumRotaR3"
        $ns_ at 13.0    "$node_(26) sscs startDevice 0"
        $ns_ at 0.5     "$node_(27) NodeLabel ConsumRotaR3"
        $ns_ at 13.5    "$node_(27) sscs startDevice 0"
        $ns_ at 0.5     "$node_(28) NodeLabel ConsumRotaR3"
        $ns_ at 14.0    "$node_(28) sscs startDevice 0"
        $ns_ at 0.5     "$node_(29) NodeLabel ConsumRotaR3"
        $ns_ at 14.5    "$node_(29) sscs startDevice 0"
        $ns_ at 0.5     "$node_(30) NodeLabel ConsumRotaR3"
        $ns_ at 15.0    "$node_(30) sscs startDevice 0"
        $ns_ at 0.5     "$node_(31) NodeLabel ConsumRotaR3"
        $ns_ at 15.5    "$node_(31) sscs startDevice 0"
        $ns_ at 0.5     "$node_(32) NodeLabel ConsumRotaR2"
        $ns_ at 16.0    "$node_(32) sscs startDevice 0"
        $ns_ at 0.5     "$node_(33) NodeLabel ConsumRotaR2"
        $ns_ at 16.5    "$node_(33) sscs startDevice 0"
        $ns_ at 0.5     "$node_(34) NodeLabel ConsumRotaR1"
        $ns_ at 17.0    "$node_(34) sscs startDevice 0"
        $ns_ at 0.5     "$node_(35) NodeLabel ConsumRotaR5"
        $ns_ at 0.5     "$node_(35) NodeLabel ConsumRotaR6"
        $ns_ at 0.5     "$node_(35) NodeLabel ConsumRotaR4"
        $ns_ at 17.5    "$node_(35) sscs startDevice 0"
        $ns_ at 0.5     "$node_(36) NodeLabel ConsumRotaR5"
        $ns_ at 18.0    "$node_(36) sscs startDevice 0"
        $ns_ at 0.5     "$node_(37) NodeLabel ConsumRotaR6"
        $ns_ at 18.5    "$node_(37) sscs startDevice 0"
        $ns_ at 0.5     "$node_(38) NodeLabel ConsumRotaR6"
        $ns_ at 19.0    "$node_(38) sscs startDevice 0"
        $ns_ at 0.5     "$node_(39) NodeLabel ConsumRotaR6"
        $ns_ at 19.5    "$node_(39) sscs startDevice 0"
        $ns_ at 0.5     "$node_(40) NodeLabel ConsumRotaR4"
        $ns_ at 20.0    "$node_(40) sscs startDevice 0"
        $ns_ at 0.5     "$node_(41) NodeLabel ConsumRotaR4"
        $ns_ at 20.5    "$node_(41) sscs startDevice 0"
        $ns_ at 0.5     "$node_(42) NodeLabel ConsumRotaR4"
        $ns_ at 21.0    "$node_(42) sscs startDevice 0"
        $ns_ at 0.5     "$node_(43) NodeLabel ConsumRotaR4"
        $ns_ at 21.5    "$node_(43) sscs startDevice 0"
        $ns_ at 0.5     "$node_(44) NodeLabel ConsumRotaR4"
        $ns_ at 22.0    "$node_(44) sscs startDevice 0"
        $ns_ at 0.5     "$node_(45) NodeLabel ConsumRotaR4"
        $ns_ at 22.5    "$node_(45) sscs startDevice 0"
        $ns_ at 0.5     "$node_(46) NodeLabel ConsumRotaR4"
        $ns_ at 23.0    "$node_(46) sscs startDevice 0"
        $ns_ at 0.5     "$node_(47) NodeLabel ConsumRotaR4"
        $ns_ at 23.5    "$node_(47) sscs startDevice 0"
```

```
        $ns_ at 0.5      "$node_(48) NodeLabel ConsumRotaR3"
        $ns_ at 24.0     "$node_(48) sscs startDevice 0"

# $ns_ at 24.0 "$node_(48) sscs startDevice 0" → significa dizer que o consumidor 48 aparecerá na tela do NAM em 24.0s

# apptime 1 é como é chamada a primeira comunicação
# Ao comunicar, ira aparecer escrito na tela no ns "Transmitting data"
        Mac/802_15_4 wpanNam PlaybackRate 12ms
        $ns_ at $appTime1 "Mac/802_15_4 wpanNam PlaybackRate 1.0ms"
        $ns_ at [expr $appTime1 + 0.5] "Mac/802_15_4 wpanNam PlaybackRate 2.0ms"
        $ns_ at $appTime1 "puts \"\nTransmitting data ...\n\""


# Para utilizar o protocolo do ZigBee é necessário utilizar 3 tipos de trafego: CBR, Poisson e FTP onde CBR e FTP correspondem ao tráfico dos protocolos UDP e TCP, respectivamente.

# 1º Tipo de trafego utilizado: CBR (Utilizados para representar o tráfico de aplicações de voz e vídeo). CBR utiliza o protocolo UDP
        # Setup traffic flow between nodes
        puts " CbrTraffic Packet size=80....."
        puts " "
        proc cbrtraffic { src dst interval starttime } {
          global ns_ node_
          set udp_($src) [new Agent/UDP]
          eval $ns_ attach-agent \$node_($src) \$udp_($src)
# attach-agent → Conexão dos agentes aos respectivos nós da rede
          set null_($dst) [new Agent/Null]
# new agent → cria o agente Null
          eval $ns_ attach-agent \$node_($dst) \$null_($dst)
# Null receberá os pacotes CBR
          set cbr_($src) [new Application/Traffic/CBR]
# new Application → cria a aplicação CBR
          eval \$cbr_($src) set packetSize_ 80
# Tamanho do pacote: 80 bytes (para uso do NAM)
          eval \$cbr_($src) set interval_ $interval
# Não há intervalo entre cada pacote transmitido (para uso do NAM)
          eval \$cbr_($src) set random_ 0
          #eval \$cbr_($src) set maxpkts_ 10000
          eval \$cbr_($src) attach-agent \$udp_($src)
          eval $ns_ connect \$udp_($src) \$null_($dst)
          $ns_ at $starttime "$cbr_($src) start"
        }

# 2º Tipo de Trafego Utilizado: Poisson
# Comunica a fonte (utilzando o protocolo UDP) com o destino (Agente Null)
# Os parametros utilizados sao necessrios para o funcionamento do programa
        puts " PoissonTraffic packet size=70 ....."
        puts " "
        proc poissontraffic { src dst interval starttime } {
          global ns_ node_
          set udp($src) [new Agent/UDP]
          eval $ns_ attach-agent \$node_($src) \$udp($src)
          set null($dst) [new Agent/Null]
          eval $ns_ attach-agent \$node_($dst) \$null($dst)
          set expl($src) [new Application/Traffic/Exponential]
```

```
# Tamanho do pacote: 70 bytes
        eval \$expl($src) set packetSize_ 70
        eval \$expl($src) set burst_time_ 0
        eval \$expl($src) set idle_time_ [expr $interval*1000.0-70.0*8/250]ms
        eval \$expl($src) set rate_ 250k
# O que resulta de uma taxa de aproximadamente 250KBps
        eval \$expl($src) attach-agent \$udp($src)
        eval $ns_ connect \$udp($src) \$null($dst)
        $ns_ at $starttime "$expl($src) start"
      }
# Utilizando o padrão 802.15.4. (ZigBee)
      # ZBR -> ZigBee Routing (padrão 802_15_4)
      if {$val(rp) == "ZBR"} {
            Mac/802_15_4 wpanCmd callBack 2
      }
# Utilização dos 2 tipos de traficos juntos: Mix (CBR + Poisson), CBR e Poisson
      if { ("$val(traffic)" == "mix") || ("$val(traffic)" == "cbr") || ("$val(traffic)" == "poisson") } {
        if { "$val(traffic)" == "mix" } {
            set trafficName "cbr + poisson"
            set traffic1 cbr
            set traffic2 poisson
        } else {
            set trafficName $val(traffic)
            set traffic1 $val(traffic)
            set traffic2 $val(traffic)
        }
 # Inicio de Comunicação
        puts "\nTraffic: $trafficName"
        #Mac/802_15_4 wpanCmd ack4data on
        puts [format "Acknowledgement for data: %s" [Mac/802_15_4 wpanCmd ack4data]]
# Faz a primeira comunicação em apptime 1 (20.5s) do coordenador com o consumidor 28
        ${traffic1}traffic 0 28 $val(trInterval) $appTime1
# Faz a segunda comunicação em apptime 2 (20.8s) do coordenador com o consumidor 43
        ${traffic2}traffic 0 43 $val(trInterval) $appTime2
        if { "$val(nodeDown)" == "yes" } {
            $ns_ at [expr $appTime2 + 3.0] "$node_(0) node-down"
            set tmpTime [format "%.1f" [expr $appTime2 + 3.0]]
            $ns_ at [expr $appTime2 + 3.0] "$ns_ trace-annotate \"(at $tmpTime) node down:
0\""
# node-down
            $ns_ at [expr $appTime2 + 10.0] "$node_(0) node-up"
            set tmpTime [format "%.1f" [expr $appTime2 + 10.0]]
            $ns_ at [expr $appTime2 + 10.0] "$ns_ trace-annotate \"(at $tmpTime) node up:
0\""
        }
# Insere as cores ao AODV, ARP E MAC
# Nessa linha também seria trocado o protocolo AODV pelo DSR
        Mac/802_15_4 wpanNam FlowClr -p AODV -c tomato
        Mac/802_15_4 wpanNam FlowClr -p ARP -c green
        Mac/802_15_4 wpanNam FlowClr -p MAC -c navy
# Ao comunicar o coordenador com o consumidor 28, o NAM vai inserir um circulo azul
nestes dispositivos
        $ns_ at $appTime1 "$node_(0) add-mark m1 blue circle"
        $ns_ at $appTime1 "$node_(28) add-mark m2 blue circle"
        $ns_ at $appTime1 "$ns_ trace-annotate \"(at $appTime1)
```

```
# Ao comunicar o coordenador com o consumidor 43, o NAM vai inserir um circulo verde nesses nós
        $ns_ at $appTime2 "$node_(0) add-mark m3 green4 circle"
        $ns_ at $appTime2 "$node_(43) add-mark m4 green4 circle"
        $ns_ at $appTime2 "$ns_ trace-annotate \"(at $appTime2)
        if { "$val(traffic)" == "cbr" } {
                set pktType cbr
                set pktType2 cbr
        } elseif { "$val(traffic)" == "poisson" } {
                set pktType exp
                set pktType2 exp
        } else {
                set pktType cbr
                set pktType2 exp
        }
# -s = source / -d= destino -- Estipula as cores
        Mac/802_15_4 wpanNam FlowClr -p $pktType -s 0 -d 28 -c blue
        Mac/802_15_4 wpanNam FlowClr -p $pktType2 -s 0 -d 43 -c green4
    }
# 3° Tipo de Trafego utilizado: FTP Traffic utilizado para representar aplicações com transferências de arquivo, utiliza o TCP.
    puts " FtpTraffic packet size=60 ....."
    puts " "
    proc ftptraffic { src dst starttime } {
      global ns_ node_
      set tcp($src) [new Agent/TCP]
      eval \$tcp($src) set packetSize_ 60
      set sink($dst) [new Agent/TCPSink]
      eval $ns_ attach-agent \$node_($src) \$tcp($src)
      eval $ns_ attach-agent \$node_($dst) \$sink($dst)
      eval $ns_ connect \$tcp($src) \$sink($dst)
      set ftp($src) [new Application/FTP]
      eval \$ftp($src) attach-agent \$tcp($src)
      $ns_ at $starttime "$ftp($src) start"
    }
# Repete a mesma comunicação acima, só que agora para o FTP
# Repete todos os procedimentos já ditos anteriormente
    if { "$val(traffic)" == "ftp" } {
      puts "\nTraffic: ftp"
      #Mac/802_15_4 wpanCmd ack4data off
      puts [format "Acknowledgement for data: %s" [Mac/802_15_4 wpanCmd ack4data]]
      ftptraffic 0 6 $appTime1
      ftptraffic 0 15 $appTime2
      Mac/802_15_4 wpanNam FlowClr -p AODV -c tomato
      Mac/802_15_4 wpanNam FlowClr -p ARP -c green
      Mac/802_15_4 wpanNam FlowClr -p MAC -c navy
      $ns_ at $appTime1 "$node_(0) add-mark m1 blue circle"
      $ns_ at $appTime1 "$node_(28) add-mark m2 blue circle"
      $ns_ at $appTime1 "$ns_ trace-annotate \"(at $appTime1) ftp traffic from node 0 to node 28\""
      Mac/802_15_4 wpanNam FlowClr -p tcp -s 0 -d 28 -c blue
      Mac/802_15_4 wpanNam FlowClr -p ack -s 28 -d 0 -c blue
      $ns_ at $appTime2 "$node_(0) add-mark m3 green4 circle"
      $ns_ at $appTime2 "$node_(43) add-mark m4 green4 circle"
      $ns_ at $appTime2 "$ns_ trace-annotate \"(at $appTime2) ftp traffic from node 0 to node 43\""
```

```
        Mac/802_15_4 wpanNam FlowClr -p tcp -s 0 -d 43 -c green4
        Mac/802_15_4 wpanNam FlowClr -p ack -s 43 -d 0 -c green4
    }

# Definição dos nós no NAM
    # defines the node size in nam
    for {set i 0} {$i < $val(nn)} {incr i} {
        $ns_ initial_node_pos $node_($i) 2
    }
# Definição dos nós no final da simulação
    # Tell nodes when the simulation ends
    for {set i 0} {$i < $val(nn) } {incr i} {
        $ns_ at $stopTime "$node_($i) reset";
    }

# Stop Time = Pára a simulação em 100s
    $ns_ at $stopTime "stop"
    $ns_ at $stopTime "puts \"\nNS EXITING...\n\""
    $ns_ at $stopTime "$ns_ halt"

    proc stop {} {
        global ns_ tracefd appTime val env
        $ns_ flush-trace
        close $tracefd
        set hasDISPLAY 0
        foreach index [array names env] {
            #puts "$index: $env($index)"
            if { ("$index" == "DISPLAY") && ("$env($index)" != "") } {
                set hasDISPLAY 1
            }
        }
        if { ("$val(nam)" == "wpan_teste5.nam") && ("$hasDISPLAY" == "1") } {
                puts "Running NAM..."
            exec nam wpan_teste5.nam &
        }
    }
    puts "Iniciando a Simulação..."
    puts "Por: Cristiane Amaral de Magalhaes"
    puts "Número de Nós:$val(nn)"
    puts "Modelo:$val(prop)"
    puts "Protocolo:$val(rp)"
    puts "Starting Simulation..."
    $ns_ run
```

B Instalação do NS-2:

Há versões de NS para diversos tipos de sistemas operacionais como: *FreeBSD, Linux, SunOS, Solaris e Windows.* Neste trabalho, será mostrada apenas duas instalações: uma em *Microsoft Windows* (XP) e outra no *Linux* (Distribuição *Slackware*).

Passo a passo para instalação no *Windows*:

1) São necessários alguns pacotes para a instalação em *Windows* adquiridos em [50]. São eles:
    - *NS Binary for Windows* (32-bits) → ns-2.1b9a-win32.exe
    - *Active TCL*
    - *Cygwin* → setup.exe
    - NAM → nam-1.0a11a-win32.exe
2) Para instalar basta criar uma nova pasta no diretório C como por exemplo: C:/cygwin
3) Fazer o download do *NS Binary for Windows*, salvar na pasta acima e renomeá-lo para: ns.exe
4) Fazer o download do NAM, salvar na pasta acima e renomeá-lo para: nam.exe
5) Fazer o download do *Cygwin* e instala-lo dentro do diretório acima
6) Fazer o download do *ActiveTCL* e instala-lo no diretório acima
7) Criar o diretório home dentro de C:/cygwin e salvar todos os scripts em tcl dentro do *home*
8) Copiar o nam.exe e o ns.exe para dentro dos diretórios: C:/cygwin/bin e C:/Tcl/bin
9) Procure um exemplo qualquer pela Internet em tcl, como por exemplo, exemplo2.tcl. Para rodar, digite: $ ns exemplo2.tcl

Esta instalação é chamada de instalação em partes (*in pieces*), pois é necessário baixar programa por programa. É bastante útil quando se deseja ou se necessita economizar espaço em disco.

O Cygwin é um emulador e atualmente está na versão 1.5.24 e pode ser encontrado em [51]. Nas telas de instalação do Cygwin, há alguns pacotes necessários

para que este funcione corretamente, são eles: xorg-x11-base, xorg-x11-bin, xorg-x11-bin-dlls, xorg-x11-bin-lndir, xorg-x11-devel, xorg-x11-etc, xorg-x11-f100, xorg-x11-fcyr, xorg-x11-fenc, xorg-x11-fnts, xorg-x11,fscl, xorg-x11-libs-data, xorg-x11-man-pages, xorg-x11-man-pages-html, xorg-x11-nest, X-start-menu-icons, X-startup-scripts, xorg-x11-vfb, xorg-x11-xwin, xorg-x11-xwin-gl. Além disso, instalação do Cygwin tem que ser feita de modo *full* para que todos s pacotes sejam instalados e para o NS-2 tenha um bom funcionamento.

Passo a passo para a instalação em *Linux* (Distribuição *Slackware*):

Utilizar para esse caso o pacote "*allinone*", ou seja, todos os pacotes dentro de uma única instalação. É um pacote único que requer aproximadamente 250 MB de espaço em disco. O *allinone* é mais recomendado para instalação em *Linux* e vem com os seguintes pacotes:

Tcl release 8.3.2 (componente requerido)
Tk release 8.3.2 (componente requerido)
Otcl release 1.0a8 (componente requerido)
TclCL release 1.0b12 (componente requerido)
Ns release 2.1b9a (componente requerido)
Nam release 1.0a11a (componente requerido)
Xgraph version 12 (componente requerido)
CWeb version 3.4g (componente requerido)
SGB version 1.0 (componente requerido)
Gt-itm gt-itm e sgb2ns 1.1 (componente opcional)
Zlib version 1.1.3 (componente opcional, mas requeridos se o pacote Nam for utilizado).

1) Salvar o pacote ns-allinone-2.31.tar.gz para dentro do meu diretório C;
2) Descompactar esse arquivo com o comando: tar xvfz ns-allinone-2.31.tar.gz (ele mostra na tela uma série de ítens que estão sendo instalados)
3) Para começar a usar o NS-2 basta entrar no diretório dele com o comando: cd ns-allinone-2.31 e depois digitar: ./install. (ele mostra na tela os pacotes sendo instalados)
4) Para terminar, tem que criar um PATH com o caminho para toda vez que chamarmos a função NS, ele entrar no programa automaticamente.
5) Para iniciar o modo gráfico, basta dar o comando: startx ou startxwin.bat

6) Para rodar algum *script* em tcl, basta entrar no diretório em que estão os *scritps* e digitar: ns exemplo.tcl
7) Para rodar o NAM, basta digitar nam exemplo.tcl ou somente nam que ele já entra na tela do NAM.
8) Ao entrar na tela do NAM tem a opção: File → New que serve para criar /desenhar o estudo de caso que o usuário desejar.
9) Dentro do diretório do C:/ns-allinone-2.31/ns-2.31 temos exemplos de implementações com os protocolos: AODV, DSR, DSDV, TORA além da pasta WPAN com exemplos de implementações e instalação do protocolo 802.15.4