Bianca Portes de Castro

Redes de Sensores Sem Fio (RSSF)

Orientador:
Eduardo Pagani Júlio

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Juiz de Fora

Monografia submetida ao corpo docente do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como parte integrante dos requisitos necessários para obtenção do grau de bacharel em Ciência da Computação

Prof. Eduardo Pagani Júlio, M. Sc.
Orientador

Prof. Marcelo Lobosco, D. Sc.

Prof. Eduardo Barrére, D. Sc.

Resumo

Este trabalho fundamenta-se na análise e simulação de redes de sensores sem fio com o intuito de apresentar a problemática na projeção de soluções e aplicações voltadas para esta área. Ele abrange o padrão IEEE 802.15.4, prossegue apresentando alguns simuladores disponíveis para este tipo de rede e finaliza com uma breve simulação inspirada em sensoriamento de pontes.

Palavras-chave: redes de sensores sem fio, IEEE 802.15.4, simuladores.

Abstract

This work is based on the analysis and simulation of wireless sensor networks with the aim of to present the problematic in the projection of solutions and applications related to this area. It cover the IEEE 802.15.4 standard, continues presenting some available simulators for this network type and concludes with one brief simulation inspired by sensing bridges.

Keywords: wireless sensor networks, IEEE 802.15.4, simulators.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

As Redes de Sensores Sem Fio (RSSF) têm como origem a integração de circuitos de sensoriamento, comunicação e alimentação elétrica através da tecnologia de circuitos integrados digitais. Tal abordagem permite que sejam construídos circuitos complexos num pequeno espaço físico, originando assim uma arquitetura barata, mas que possui problemas intrínsecos, como grande dissipação de energia.

Estas restrições, porém, não impedem que RSSF ganhem espaço nos dias atuais. O crescimento se dá motivado pela gama de possíveis cenários de atuação, que vão desde a concepção de casas inteligentes, além de propiciar aplicações de saúde e industrial, até chegar em aplicações militares.

Em geral, os principais requisitos das redes de sensores estão relacionados com a questão do consumo e do custo, que por sua vez devem ser baixos. Porém outros requisitos podem surgir dependendo da aplicação alvo, o que leva à consideração por parte dos pesquisadores de implementações alternativas.

Este trabalho segue a seguinte linha: primeiramente, no capítulo 2, são apresentados os problemas inerentes à concepção de uma rede de sensores sem fio. Em seguida, nos capítulos 3 e 4, é exposto um exemplo de protocolo utilizado na comunicação entre sensores, bem como simuladores disponíveis que auxiliem no processo de projeção. No capítulo 5, é apresentada uma aplicação para monitoramento de pontes, juntamente com seus requisitos e resultados experimentais. Por fim, no capítulo 6, a fim de se estender o tempo de vida útil de uma rede, conclui-se sobre a importância da etapa despendida na projeção e a necessidade de gerência de recursos em cada uma das camadas que compõem o modelo de arquitetura de comunicação.

2 *Objetivo de RSSF*

As redes de computadores vêm sofrendo constante evolução desde a sua concepção em função dos avanços em tecnologias de hardware e software. Estes avanços promoveram um grande salto na área de comunicação e computação móvel que por sua vez têm impulsionado pesquisas em comunicação sem fio.

Redes sem fio se referem a redes de computadores nas quais é dispensável o uso de cabos para a transmissão de dados. Este tipo de rede é apropriado para situações onde não se pode ou não se deseja uma instalação com fios e onde é requerido o acesso imediato a informação. Todavia, tais redes são mais suscetíveis a interferências causadas pelo meio externo, já que o meio pelo qual a informação trafega não possui nenhum tipo de proteção física.

A tabela 2.1 apresenta características de várias tecnologias sem fio e suas aplicações.

Tabela 2.1: Tecnologias sem fio e características associadas (PEREIRA; AMORIM; CASTRO, 2003)

Tecnologias sem fio	Área de Aplicação
Celular	Serviços no campo, segurança pública, controle de estoque, transportadoras e atividades de linhas aéreas.
WLAN	Lojas varejistas, serviços de saúde, tele-diagnósticos, estudantes, restaurantes, escritórios, indústria manufatureiras e estoque.
GPS	Pesquisa, agência de aluguel de carros e esportes.
PCS	*GPS*, Multimídia e Telemetria.
Redes *ad hoc* e de sensores	Sensores de ambiente, máquinas de prognósticos, detecção de pontes quebradas, condições das estradas e sensores biológicos.

Sistemas ubíquos foram vislumbrados inicialmente por Mark Weiser (WEISER, 1991), referindo-se ao futuro onde milhares de pequenos computadores estariam imersos no ambiente sem que sua presença fosse notada. A construção de sistemas ubíquos vem sendo reafirmado desde 2005 pelo *The British Computer Society* como um dos grandes desafios para a Ciência da Computação nos próximos anos, dando destaque para a Computação Consciente do Contexto

(GRAND. . . , 2010).

Na computação Consciente do Contexto, as aplicações adaptam seu comportamento de acordo com o ambiente e com modificações que nele ocorrem. Para que essa adaptação seja possível, o sistema deve conseguir perceber essas transformações, papel legado aos sensores. A proposta deste tema em construção é que os computadores se integrem ao cotidiano dos seres humanos, monitorando e provendo informações sobre o ambiente que contextualizem as necessidades que se apresentem, diluindo cada vez mais a atuação direta do homem no sistema (FEHLBERG, 2007).

A partir desta demanda, surgiu a necessidade de se aprofundar em conhecimento sobre pequenos dispositivos com consumo reduzido de energia, capacidade de computação e que adaptem seu comportamento automaticamente de acordo com a situação a qual estão imersos, sem a necessidade de atuação humana direta. Assim, redes de sensores tomaram um impulso e ganharam foco em pesquisas que, aliadas ao desenvolvimento de comunicação sem fio, repercutiram numa nova vertente: Redes de Sensores Sem Fio (RSSF).

Uma RSSF (vide figura 2.1) é formada por diversos dispositivos autônomos, geralmente de mesma capacidade, que tem o intuito de monitorar algum fenômeno de interesse a partir da detecção ou coleta de informações individuais em cada nó. Podemos resumir sua função à monitoração, rastreamento e controle de fatores complexos do ambiente, como temperatura, pressão, degradação ambiental, detecção de presença, entre outros; com o intuito de suprir ao máximo a necessidade de intervenção humana direta que, em muitos casos, é até mesmo inviável.

Essa troca traz benefícios consideráveis ao observador, o interessado nos dados fornecidos pelo fenômeno em questão. O observador obtém informação mais precisa, já que uma grande massa de dados, coletada de forma distribuída pelos sensores na área de monitoramento, pode ser analisada. A partir desses dados, pode-se inferir sobre o estado desta área foco de maneira mais veloz do que aconteceria se sua coleta dependesse de métodos manuais. Em casos nos quais seja necessário responder de acordo com o estado do sistema, como em aplicações de garantia de segurança, a agilidade é outro benefício proporcionado por RSSF.

A princípio, qualquer ambiente é passível de ter uma rede de sensores monitorando algum dado. O levantamento de requisitos para construção e concretização desse projeto deve considerar o fenômeno; o meio no qual a rede será inserida, mensurando-se fatores externos que podem influenciar o desempenho desta rede, como aspectos físicos de atenuação de sinal no canal de propagação, acesso à luz solar ou energia térmica, etc; a escolha dos dispositivos sensores que sejam cabíveis à situação e as variáveis restritivas que representem os recursos disponíveis

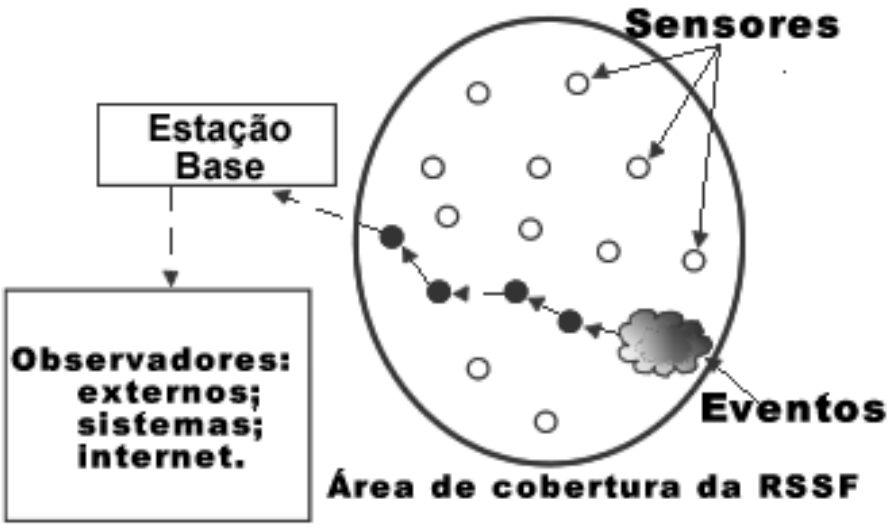

Figura 2.1: Rede de Sensores Sem Fio (AKYILDIZ et al., 2002)

nestes dispositivos.

A atuação diversificada de RSSF atrelada às limitações de recursos físicos implica em gerações de projetos altamente direcionados à aplicação em questão. Como a construção direcionada do hardware tornaria o projeto do sistema, na maioria das vezes, economicamente inviável comparado ao custo da construção do software, a saída é optar pela escolha de plataformas de hardware genéricas existentes no mercado. Isto permite até mesmo que a rede seja reutilizada em outros projetos distintos apenas com a modificação do software relativo à aplicação.

Contudo, mantendo o foco na construção do software, nota-se que variáveis exaustivamente tratadas como prioritárias em um contexto de ambiente podem ser consideradas secundárias em outros ambientes nos quais deseje-se monitorar o mesmo fenômeno. Por exemplo, o monitoramento do cantos dos pássaros em uma floresta densamente fechada possui variáveis restritivas que não seriam tão prioritárias para o monitoramento do mesmo fenômeno numa região semiárida.

O consumo de energia, por exemplo, é amplamente citado na bibliografia como prioritário na construção ou escolha de protocolos de roteamento e comunicação devido ao impacto significativo que este pode provocar na redução de tempo de vida da rede. Porém, em ambientes nos quais seja possível a extração de energia do meio, como energia térmica ou química, o fator consumo ponderaria para menos nas escolhas de caracterização do sistema enquanto outras

variáveis mais diretamente relacionadas com a aplicação em si poderiam ganhar foco.

Reduzir o grau de importância da variável consumo de energia, ainda assim não eliminaria sua importância junto à projeção do sistema. Enquanto redes tradicionais almejam alcançar maior velocidade de comunicação a reduzidos índices de perda de pacotes, protocolos de redes de sensores mantêm seu foco no uso otimizado da energia disponível. Este uso é muitas vezes priorizado a custos de baixo rendimento de processamento de tarefas ou alto tempo de espera na transmissão de um resultado a fim de se prolongar ao máximo o tempo de vida de uma rede (AKYILDIZ et al., 2002).

Logo, o intuito da aplicação em conjunto com os aspectos físicos da rede e fatores externos de influência devem ser mapeados e cuidadosamente relacionados. Este levantamento inicial proporcionará uma boa bagagem para a escolha da melhor ferramenta para simulação, auxiliando no mapeamento de risco e projeção otimizada da rede na escolha do hardware e topologia, por exemplo.

Uma RSSF tende a ser autônoma e requer um alto grau de cooperação para executar suas tarefas. Algoritmos distribuídos tradicionais, como protocolos de comunicação e eleição de líder, devem ser revistos para esse tipo de ambiente antes de serem usados diretamente. Como trocas de mensagens para acordar processamento, que poderiam acabar por inundar a rede devido à participação de centenas de nós neste ínterim.

Ainda assim, o processamento distribuído pode ser uma saída para aumentar o desempenho e uso inteligente desses limitados recursos. Um sensor ocioso pode receber dados de outros sensores e processá-los antes de enviar a uma unidade central de processamento ou ao seu nó vizinho para colaboração na execução de uma determinada tarefa. Além de agilizar o processamento, este paralelismo contribuiria também para diminuir o consumo de energia na transmissão, uma vez que os dados a serem transmitidos pela rede serão menores do que seriam sem um pré-processamento.

Quanto aos padrões de projeto, percebe-se a dificuldade de uso de padrões que objetivem código reusável. Isto porque como a construção do hardware específico para cada aplicação é economicamente inviável para os projetistas, fica a cargo dos desenvolvedores extraírem o melhor das ferramentas existentes em relação ao objetivo, o que os força a escrever versões de softwares específicos para cada aplicação, focando na eficiência e profunda dependência do hardware em questão.

Este fator atua também diretamente na construção das ferramentas de simulação que, decorrente desta falta de padrões, são limitadas quanto aos possíveis testes de modelos de redes

em ambientes diversificados.

Tamanha é a importância das ferramentas de simulação que à frente abordaremos em capítulo separado.

2.1 Métricas qualitativas

A modelagem de redes de sensores é influenciada por vários fatores. Segundo Akyildiz et al. (2002), as principais métricas para avaliar a rede, dada a sua implementação, incluem tolerância a falhas, escalabilidade, custo de produção, restrições de hardware, topologia, ambiente e suas restrições, meio de transmissão e consumo de energia.

Cada um desses fatores exige requisitos específicos na concepção e projeto dos nós e, conseqüentemente na construção de diretrizes para protocolos ou algoritmos para à rede.

2.1.1 Tolerância à falhas

Sensores são facilmente sujeitos a falhas, seja devido as más condições físicas do ambiente, ao término da vida útil da bateria ou a interferências ambientais. Contudo, a rede deve ser capaz de se recuperar e concluir a tarefa proposta, isto para falhas não catastróficas.

Assim, é exigido que redes de sensores possuam topologia dinâmica. Elas devem adotar uma nova topologia que interconecte o máximo de nós para que a informação flua, mesmo sem a presença dos nós faltosos. No entanto, se um grande número de nós da rede foi comprometido, impossibilitando que alguma rota seja traçada, um aviso de que o funcionamento não pode mais ser garantido deve ser emitido para o observador.

Porém, protocolos de descoberta de rota exigem que mensagens adicionais transitem na rede, as quais podem ser muito custosas. Para tanto, os níveis de tolerância a falhas devem ser ponderados segundo à aplicação aos quais eles se propõem e os modelos devem ser definidos com este propósito.

2.1.2 Escalabilidade

Captura de dados do meio sob demanda, nós densamente posicionados, incremento de novos dispositivos, retiradas de outros, entre outras coisas, são tratados constantemente em redes de sensores.

A escalabilidade nesse meio é também um fator crítico, uma vez que a rede pode conter cerca de centenas ou mesmo, dependendo da aplicação, milhões de nós a serem administrados. Novos nós podem ser incrementados à rede. A rede deve ser capaz de lidar com esses fatores e regular o seu processamento de forma eficiente segundo os requisitos que se apresentem, tanto de processamento quanto de hardware.

2.1.3 Custo de produção

À medida que a tecnologia avança, acompanhada de uma diversidade de produtos já arquitetados e disponíveis no mercado, o custo dos nós tende a reduzir. Como redes de sensores são constituídas por um número muito grande de dispositivos, o custo de um único nó é muito importante na definição do custo total da rede.

Sendo assim, se o custo da rede idealizada com nós produzidos especificamente para dada aplicação for maior do que usar dispositivos já existentes, o projeto torna-se monetariamente crítico, devendo ser revisto.

2.1.4 Restrições de hardware

Nós sensores devem ser pequenos e leves, a fim de facilitar a instalação nos mais diversos locais. Por isso, eles acabam por ter baixa capacidade computacional e pouca memória relativa ao espaço condizente.

Além disso, o baixo consumo de energia deve ser priorizado no intento de aumentar a vida útil da bateria, mesmo que a necessidade do uso de baterias seja eliminada por extração de energia do ambiente, como descrito por (BARBOSA et al., 2005) e graças aos avanços promovidos pela microeletrônica.

2.1.5 Topologia

Realizar manutenção em RSSF é uma tarefa muito difícil, principalmente quando os nós, em sua maioria, estão isolados e inacessíveis. Como há um número elevado de nós densamente posicionados, a rede requer um tratamento cuidadoso da manutenção da topologia.

Akyildiz propôs em (AKYILDIZ et al., 2002) três fases para manutenção da topologia da rede: a fase de instalação, na qual se define a forma como os nós serão depositados (por exemplo, lançamento por avião ou disposição manual); a fase de pós-instalação, onde os nós são gerenciados quanto ao seu nível de energia, funcionamento e posição; e, por último, a fase

de reinstalação que consiste na adição ou substituição de nós para se suprir a demanda devido a nós com mau funcionamento ou mudança de atividade requerida pela rede.

2.1.6 Ambiente e suas restrições

Independente do local ao qual a aplicação se destine, os sensores precisam funcionar. Seja em lugares remotos, fundo do oceano, interior de vulcões ou tornados, campos de batalha, florestas úmidas ou no interior de organismos vivos.

Para cada um desses ambientes, suas restrições devem ser levadas em consideração para a esquematização de soluções para os problemas que se apresentam.

2.1.7 Meio de transmissão

Em redes *multihop*, a comunicação dos nós é feita por meio sem fio, sendo representado basicamento pelo ar ou éter. Este canal está sujeito a vários fatores de atenuação do sinal como reflexão e dispersão em transmissões por ondas de rádio ou espalhamento e quebra de conexão por presença de obstáculos para sinais ópticos, por exemplo.

Assim como nos outros casos, o que irá implicar qual método de comunicação, frequência de banda e velocidade de transmissão de dados será escolhido para o meio é o custo econômico e as necessidades da aplicação.

2.1.8 Consumo de energia

As redes tradicionais *ad hoc* priorizam a QoS, como implementação de $buffers$ ou transmissão de pacotes excessivos por entre a rede para garantir sincronismos e estabelecimento de comunicação. Porém, como já dito, o importante aqui é a eficiência no uso da energia com o intuito de prolongar a vida útil da rede ao máximo. Por isso, pesquisas em protocolos mais eficientes que minimizem o impacto referente ao processamento, troca de mensagens e consumo no uso de recursos se fazem necessárias.

2.1.9 Ambiente e suas restrições

Independente do local ao qual a aplicação se destine, os sensores precisam funcionar. Seja em lugares remotos, fundo do oceano, interior de vulcões ou tornados, campos de batalha, florestas úmidas ou no interior de organismos vivos.

Para cada um desses ambientes, suas restrições devem ser levadas em consideração para a esquematização de soluções para os problemas que se apresentam.

2.2 Aplicações de Redes de Sensores

Aplicações de sensores representam um novo paradigma para operação de rede, tendo objetivos diferentes das redes sem fio tradicionais. A idéia do micro-sensoriamento, unido as conexões sem fio, abre uma gama de possíveis novas áreas de aplicação. Seguem abaixo alguns campos nos quais as RSSF já atuam.

2.2.1 Aplicações militares

A facilidade com que as RSSF podem ser constituídas, dado ao fácil posicionamento dos nós, auto-organização da rede e tolerância à falhas, fazem dessa rede uma ótima opção em aplicações militares, auxiliando em funções como comando, controle, comunicação, computação, inteligência militar, vigilância, aquisição de mira e reconhecimento (C4ISRT – *Command, Control, Communications, Computing, Military Intelligence, Surveillance, Target Acquisition and Reconnaissance*) (AKYILDIZ et al., 2002).

Redes de sensores devem suportar denso posicionamento à baixo custo dos nós. Em redes densas, a destruição de alguns desses nós não afeta a rede como um todo, como ocorre com outras redes tradicionais. Este fato, atrelado à simplicidade de montagem e configuração da rede, é importantíssimo para aplicações em meios hostis.

Pode-se, por exemplo, lançar nós sensores por via aérea em campos de batalha, para monitorar e/ou identificar tropas amigas ou inimigas, controle de equipamentos e munição, entre outros.

2.2.2 Aplicações ambientais

Aplicações ambientais são favorecidas nas mais diversas áreas com o suporte das redes de sensores. Alguns dos possíveis locais de atuação incluem monitoramento de nicho animal; controle de desastres ambientais; auxílio na agricultura, pecuária e meteorologia; monitoração de ambientes de difícil acesso ou inexplorados.

Quanto a este último, o ambiente aquático tem sido grandemente beneficiado e obtido foco na exploração científica devido aos grandes avanços sofridos na tecnologia, o que possibilita

que tais ambientes sejam finalmente desvendados. O meio ideal para essa ampla monitoração pode ser feito com o uso de sistemas distribuídos de rede de sensores sem fio debaixo d'água, o que proporciona uma solução promissora para exploração eficiente.

Segundo Cui et al. (2006), as redes de sensores subaquáticas devem lidar com as seguintes restrições:

- alta pressão, imprevisibilidade de atividades e uma grande dimensão de área inexplorada;
- aquisição de conhecimento localizado e preciso do nó, uma vez que a informação localizada, em ambientes subaquáticos, é muito importante, tendo o seu sentido diretamente ligado ao local e ao tempo dos dados coletados;
- sensores móveis e em grande escala;
- comunicação é feita por canal de acústica que possui sinal com 5 ordens a menos de magnitude de velocidade comparado ao via rádio;
- comunicação com vários complicadores como perda no caminho, ruído, múltiplos-caminhos e efeito de espalhamento (Doppler). Todos esses fatores causam alta taxa de erros em bit e variância no atraso de entrega da mensagem.

Porém, mesmo com todas essas restrições, conhecimentos em oceanografia física podem ser finalmente adquiridos ou confirmados com alta precisão, tais como mapeamento das massas de água. Massas de água são algumas porções do oceano que possuem características físico-químicas e biológicas bem definidas durantes longos períodos de tempo. Como o local de formação determina as características da massa d'água, as mesmas podem ser consideradas como uma "impressão digital", sendo sua denominação relacionada à sua origem (SPERB et al., 1999).

Em seu trajeto, as massas d'água sofrem misturas que alteram suas características originais. A identificação dessas massas e de suas misturas (tipo d'água) ao longo de seu percurso constitui-se num dos problemas enfrentados pela oceanografia física. Contudo, como proposto por Cui Cui et al. (2006), redes de sensores são uma saída promissora para este problema.

3 *Arquitetura de comunicação de RSSF*

Nós sensores são geralmente dispersos na área foco de monitoramento, benefício imprescindível para algumas aplicações que inviabilizariam o posicionamento individual dos nós. Além disso e ao contrário de redes estruturadas, redes *ad-hoc* não possuem a premissa de infra-estrutura fixa ou mesmo uma entidade controladora central para o gerenciamento da rede de comunicação, como redes de telefonia celular. Pressupõem-se então que cada nó deve ser capaz de auto-configuração e de estabelecer rotas de comunicação com todos os outros nós pertencentes à rede. Ou seja, as funções de rede devem ser implementadas de forma distribuída, o que logicamente aumenta a complexidade da rede. Esta auto-suficiência dos nós traz grandes benefícios como rápida instalação e maior recuperabilidade de falhas de conexão, que podem ser provocadas por desativação e até incremento de novos nós na rede, por exemplo. Isto vem motivando novas pesquisas para adaptabilidade de protocolos já existentes de redes *ad hoc* ou criação de novos para o contexto de sensores.

A pilha de protocolos deve combinar o roteamento consciente de energia disponível, viabilizar a comunicação eficiente com o meio sem fio e promover os esforços de cooperação dos nós sensores, por exemplo (AKYILDIZ et al., 2002). Além disso, fatores como capacidade de armazenamento, alcance da freqüência do canal de transmissão, limitação de energia, capacidade de processamento, segurança, entre outros, inserem novas variáveis no processo de comunicação e provocam uma corrida pela busca de novos modelos para o contexto de RSSF que melhorem o desempenho de protocolos legados ou novos. Assim, como elucidado na figura 3.1, os modelos devem ser estruturados de forma a prover em cada uma das camadas e para cada nó, soluções de gerenciamento destes recursos.

Uma breve estruturação do papel delegado à cada uma das cinco camadas, referidas na figura 3.1, segue abaixo (AKYILDIZ et al., 2002):

- segundo o contexto da tarefa a ser sensoriada, diferentes tipos de aplicações de software podem ser construídos e usados na camada de aplicação;
- a camada de transporte ajuda a manter o fluxo de dados caso a aplicação requisite isso;

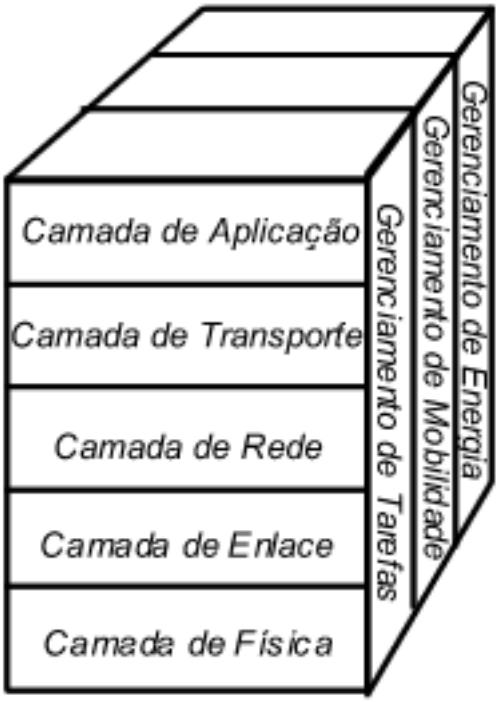

Figura 3.1: Pilha de protocolos de redes de sensores (AKYILDIZ et al., 2002)

- a camada de rede cuida do roteamento dos dados fornecidos pela camada de transporte;
- como o ambiente de comunicação é naturalmente sujeito a ruídos, além da possibilidade de mobilidade dos nós, a subcamada de protocolo MAC deve estar ciente do uso de energia e tentar minimizar colisões de transmissões em broadcast na camada de enlace;
- o endereçamento da camada física necessita de uma simples, porém robusta, técnica de transmissão e recebimento de dados.

A partir destes cuidados, uma gama de protocolos foram desenvolvidos e continuam ainda sendo foco de pesquisas. Porém, ainda são poucos os padrões que atendem RSSF.

O IEEE possui 3 padrões que atendem redes pessoais ou redes de curta distância (WPAN), ilustrado na figura 3.2, que podem ser diferenciados por velocidade de transmissão, consumo de energia e QoS (GUTIÉRREZ; CALAWAY; BARETT, 2007).

- Padrão IEEE 802.15.3 para WPAN com a mais alta taxa de troca de dados. É indicado para aplicações multimídias que requerem alta qualidade de serviço.
- Padrão IEEE 802.15.1 para WPANs com taxa mediana de troca de dados. Surgiu com o intuito de substituição de cabos de conexão entre dispositivos como telefones móveis e PDAs com aceitável qualidade de serviço para aplicações de voz.

- Padrão IEEE 802.15.4 para WPANs com baixa taxa de dados. As redes que empregam este padrão são também conhecidas como LR-WPAN (*Low Rate – Wireless Personal Área Network*). Aqui pretende-se atender dispositivos com restrição de energia e baixo custo. Redes implementadas a partir deste padrão possuem maior flexibilização para velocidade e qualidade de serviço.

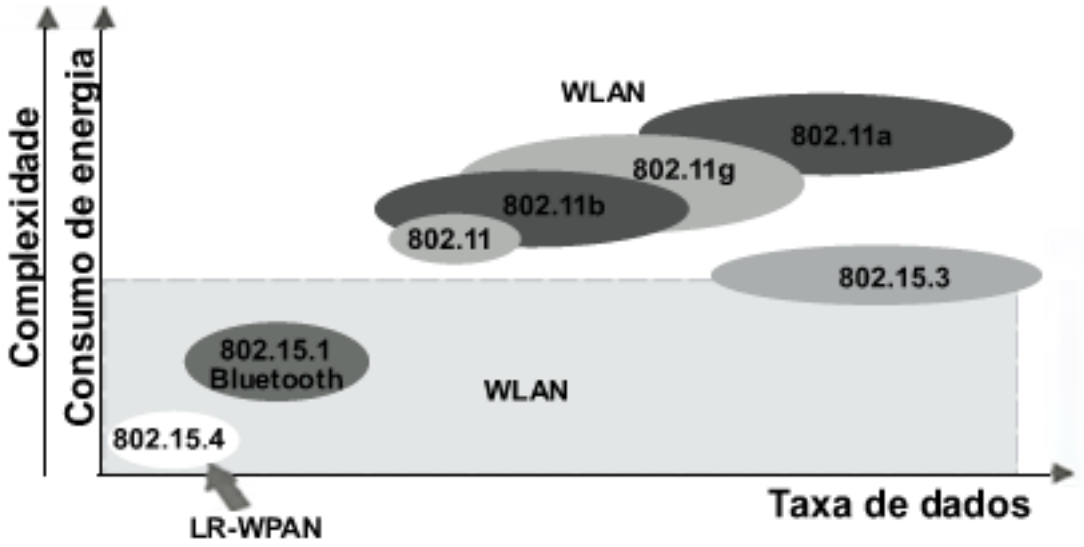

Figura 3.2: Padrões WPAN e WLAN (GUTIÉRREZ; CALAWAY; BARETT, 2007)

Por atender requisitos como custo, baixa taxa de transmissão e uso de energia, o padrão IEEE 802.15.4 é o mais indicado para aplicações de RSSF. Para tanto, será abordado um resumo com base na especificação do padrão disponibilizado online gratuitamente em (IEEE COMPUTER SOCIETY, 2006).

3.1 O padrão IEEE 802.15.4

O padrão IEEE 802.15.4 ou 802.15 WPAN (*Wireless PAN, Wireless Personal Area Network*) trata de especificações tanto para protocolo quanto comunicação dentro de redes pessoais, nas quais se trabalha sob baixas taxas de transmissão de dados. Ele é voltado para a camada física (PHY) e para a subcamada de controle de acesso ao meio (MAC, *Media Acess Control*) com o intuito de proporcionar baixa transmissão de dados entre dispositivos simples que consumam pouca energia e operem a 10 metros ou menos de distância.

Este padrão foi criado para atender as camadas de níveis mais baixos de comunicação. Ele oferece um padrão de comunicação e interconexão à camada física (PHY) e à subcamada MAC.

É responsabilidade da camada física (PHY) a ativação e desativação dos transmissores de rádio, detecção de energia do canal, indicação da qualidade do link para pacotes recebidos, seleção de canal de transmissão, avaliação de canal livre e transmissão e recepção de pacotes pelo meio físico.

Já a subcamada MAC pode ser caracterizada pela gerência de *beacons*, acesso ao canal, gerenciamento de garantia de parcela de tempo para transmissão (GTS), validação de $frame$, entrega $frame$ de reconhecimento (*ack*), associação e dissociação.

3.1.1 Topologia da Rede

Para atender aos requisitos da aplicação que se pretende construir, pode-se usar três tipos de dispositivos elucidados na figura 3.3 e expressos a seguir.

- Coordenador. É o principal dispositivo da rede e o de maior capacidade de processamento. É ele quem inicia uma nova PAN e, para cada PAN, existe exatamente um coordenador. A ele é delegada a função de selecionar o canal de comunicação e atribuir um endereço de identificação da PAN. Este endereço será o usado por todos os dispositivos que desejem se conectar à rede. O coordenador, por possuir maior capacidade de transmissão, pode ser usado como interface de comunicação para solicitação de execução ou extração de informação da estação ou conexão com outras PANs ao seu alcance.

- Roteador. Não há limite de roteadores na rede. Ele é um dispositivo opcional que deve ser permissionado pelo coordenador antes de poder atuar. Ele pode transmitir e receber transmissões de dados de rádio freqüência, podendo roteá-los pela rede, além de poder permitir que outros roteadores ou dispositivos finais façam parte da PAN.

- Dispositivos finais. Estes dispositivos são similares aos roteadores, porém não possuem o poder de permissionar entrada de dispositivo algum à PAN e nem encaminhar dados através da rede. A eles é dada a ordem de execução de alguma tarefa e retorno ao seu nó pai (coordenador ou roteador).

Uma rede LR-WPAN baseada no padrão IEEE 802.15.4 só será constituída se existir um dispositivo coordenador responsável por integrar todos os outros dispositivos à rede. Para tanto, é freqüente a construção de rede com dispositivos heterogêneos. Isto é, os dispositivos possuem restrições diferentes de hardware, como menor grau de restrição de energia e poder computacional superior, visando atender às necessidades de mais recursos demandados pelo coordenador comparado aos outros nós.

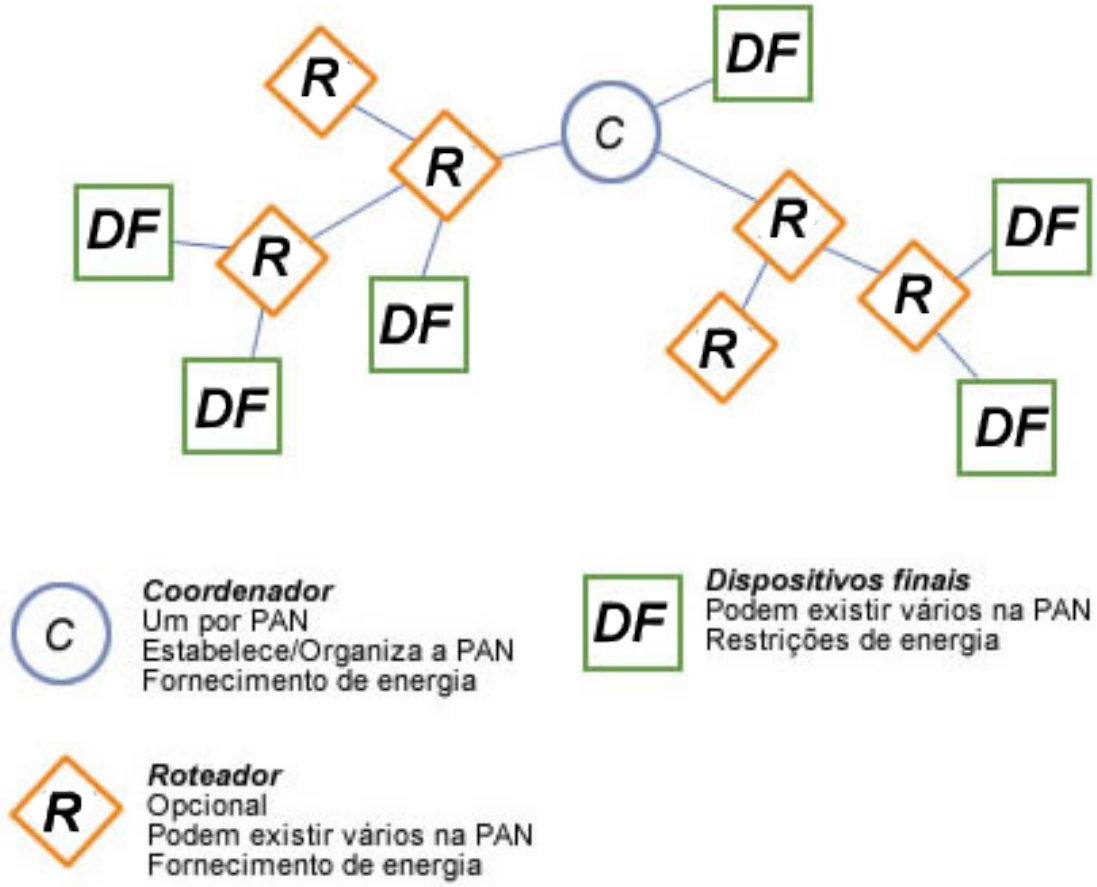

Figura 3.3: Tipos de nós suportados na rede WPAN (XBEE. . . , 2008)

Vale lembrar que a comunicação direta entre dois dispositivos só ocorrerá se ambos estiverem dentro do mesmo raio de influência de frequência de rádio e obedecerem as restrições de comunicação (dispositivos finais não se comunicam entre si). Duas são as possíveis topologias que se pode construir: estrela e ponto-a-ponto.

Na topologia estrela a comunicação é centralizada e provida apenas entre os dispositivos e o coordenador. Comunicações entre dispositivos não é permitida, sejam eles finais ou roteadores. O coordenador será usado para iniciar, terminar ou dirigir a comunicação nesta rede. Logo, se um dispositivo precisa enviar dados para outros dispositivos, ele necessita enviá-los ao coordenador da rede e este os encaminhará para o destinatário.

A topologia ponto-a-ponto permite a montagem de uma rede mais descentralizada na qual os dispositivos podem se comunicar entre si desde que estejam no mesmo raio de alcance. Esta rede pode ser estendida a partir da inclusão de roteadores, porém ainda é necessário um coordenador PAN para iniciar a montagem do aglomerado de dispositivos. A partir desta topologia, redes mais complexas podem ser montadas, como redes *mesh*.

Modelo de transferência de dados

A fim de simplificar a representação abaixo e segundo os dispositivos descritos anteriormente (coordenador, roteador e dispositivos finais), o uso isolado do termo coordenador, doravante se referirá tanto ao coordenador da PAN como aos roteadores. O mesmo será para o termo dispositivo que se referirá a qualquer nó da rede quando ele não for explicitado como sendo um dispositivo final.

É fácil deduzir os tipos de transações possíveis nesta arquitetura: emitir dados de um dispositivo para o coordenador, receber dados da rede advindos do coordenador e transferir dados entre dois dispositivos.

I- Transferência de dados para o coordenador.

Rede usando estrutura de *superframe* (figura 3.4(a)):

1. coordenador envia *beacon* de transmissão;
2. dispositivo que deseja enviar os dados escuta o canal à procura do *beacon*;
3. após receber o beacon, dispositivo sincroniza com a estrutura *superframe* e transmite o dado no momento apropriado;
4. coordenador pode opcionalmente confirmar o recebimento do *frame* enviando uma mensagem de reconhecimento.

Rede sem usar estrutura de *superframe* (figura 3.4(b)):

1. dispositivo escuta o canal e, estando o meio ocioso, transmite o dado para o coordenador;
2. coordenador pode opcionalmente confirmar o recebimento do *frame* enviando uma mensagem de reconhecimento.

II- Transferência a partir do coordenador.

Rede usando estrutura de *superframe* (figura 3.5(a)):

1. coordenador indica no *beacon* que existe uma mensagem pendente para transmissão;
2. dispositivo escuta periodicamente o canal e recebe a mensagem de pendência;
3. dispositivo envia um comando MAC solicitando os dados;
4. coordenador retorna um *frame* de reconhecimento sobre a requisição dos dados;
5. coordenador envia o *frame* de dados que estava pendente;

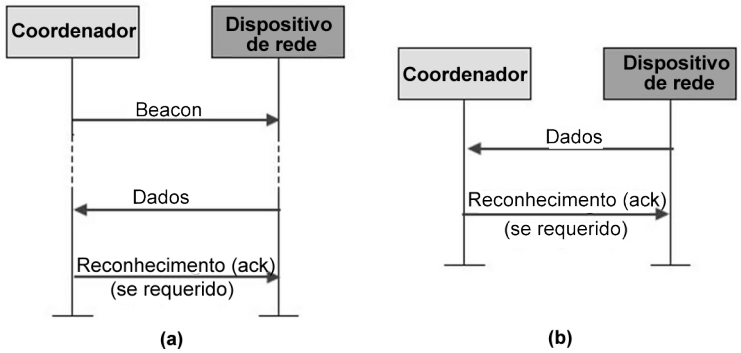

Figura 3.4: Comunicação para um coordenador usando (a) ou não (b) a estrutura de *superframe* (IEEE COMPUTER SOCIETY, 2006)

6. após receber o dado, dispositivo envia a confirmação de recebimento com sucesso do dado;
7. coordenador retira a mensagem da lista de mensagens pendentes assim que a transação completa.

Rede sem usar estrutura de *superframe* (figura 3.5(b)):

1. coordenador armazena o dado endereçado ao dispositivo e aguarda até que este dispositivo envie uma requisição de dados;
2. dispositivo envia um comando MAC de requisição de dados;
3. para toda requisição de dados advindas de dispositivos, o coordenador envia um *frame* de confirmação;
4. se os dados que estiverem pendentes forem para este dispositivo, o coordenador envia o *frame* de dados. Se não, o coordenador envia um *frame* com comprimento zero para indicar que não existem dados pendentes;
5. dispositivo retorna um *frame* de recepção com sucesso do dado transmitido.

III- Transferência de dados ponto a ponto. Todos os dispositivo de uma mesma PAN podem se comunicar, desde que estejam sob o mesmo raio de influência do sinal de rádio de transmissão.

Vale ressaltar que alguns agravantes podem pesar no processo de transmissão de dados no canal. O primeiro é o porte da rede, já que existem muitos nós disputando o meio para transmitirem seus dados. O segundo consiste na natureza do canal: o ar, sendo comum simplesmente nominá-lo por canal rádio. Aqui, vários fatores podem influenciar na degradação do sinal que transita no canal, como difração, espalhamento e ruídos de outras naturezas.

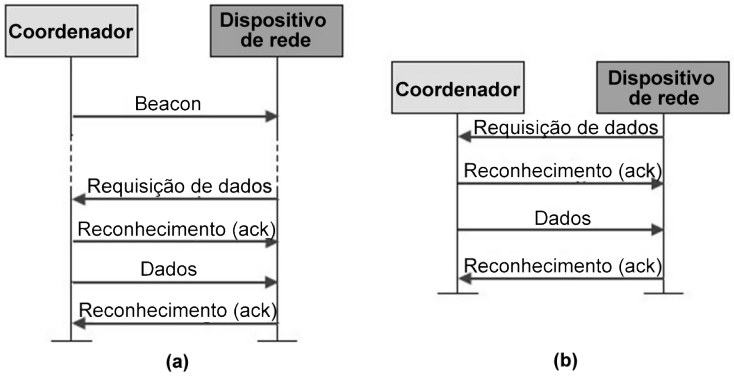

Figura 3.5: Comunicação a partir do coordenador baseado (a) ou não (b) na estrutura de *superframe* (IEEE COMPUTER SOCIETY, 2006)

Assim, para se controlar transmissões desordenadas e evitar excesso de colisões, cada dispositivo emprega o mecanismo de acesso múltiplo ao canal por escuta da portadora utilizando mecanismos de tentativa de prevenção de colisão (CSMA-CA). Dependendo da prioridade estratégica de alguns dispositivos para determinada aplicação, pode-se usar mecanismos de transmissão agendada (GTS), onde eles terão acesso exclusivo ao canal para transmissão e não executarão o CSMA-CA. *Frames* que também serão transmitidos sem o processo CSMA-CA são os *beacons* de sinalização de transmissão e *frames* de reconhecimento para informação recebida (*ack*).

O tempo de aguardo para retransmissão após uma colisão (*backoff*) em mecanismos CSMA-CA ocorre tanto em redes com *beacons* ativos ou não. Porém, este período em redes com beacons ativos é condicionado aos limites dos *slots* do *superframe*.

Estrutura do *superframe*

O *superframe* é uma estrutura dividida em 16 *slots* de tamanhos iguais e delimitada por *beacons* de transmissão. O coordenador é quem define o seu formato e transmite o *beacon* no primeiro intervalo de tempo (*slot*) de cada *superframe*.

Ele pode ser dividido em duas porções: ativa e inativa. A porção ativa compreende o período no qual a comunicação ocorre. Já na inativa o coordenador não deve interagir com a PAN, permitindo que os dispositivos entrem no modo de baixo consumo.

Beacons são usados para sincronizar os dispositivos da rede, identificar a PAN e descrever a estrutura de *superframes*. Para cancelar o uso da estrutura de *superframe*, basta desabilitar a transmissão de beacons.

Tipos de *frames*

O padrão IEEE 802.15.4 define quatro tipos de pacotes ou estruturas de *frames*: *beacon*, dados, reconhecimento (*ack*) e comandos para a camada MAC. Cada uma dessas estruturas foi desenhada a fim de possuir o mínimo de complexidade possível e, ao mesmo tempo, robustez para ser usado em canais ruidosos. A seguir uma breve descrição sobre cada um.

I- *Beacon frames*. Somente coordenadores (roteadores ou coordenador da PAN) podem transmitir *beacon frame*, independente da topologia da rede. Ele é provido como um serviço originário da subcamada de protocolo MAC e interface para a camada de protocolo física, tendo uma série de usos como marcar as fronteiras de *superframes*, sincronizar e associar dispositivos à rede.

II- *Frame* de dados. A carga útil deste *frame* vem das camadas superiores e contém o dado real a ser transmitido. É na subcamada MAC que ele recebe o cabeçalho e rodapé , caracterizando-o como um serviço desta subcamada, para logo em seguida ser remetido para a camada física.

Aqui não há discriminação quanto ao tipo de dispositivo ou topologia usada. Todos os dispositivos podem usar este recurso.

III- *Frame* de reconhecimento (*ack*). Ele é fornecido como um serviço originário da subcamada MAC com interface para a camada física, sendo usado na comunicação como um recurso para garantia de entrega de dados pelas camadas superiores na rede. Assim como o *frame* de dados, *frame* de reconhecimento não possuem restrição de uso quanto ao dispositivo ou topologia da rede.

IV- *Frame* de comando MAC. Como os outros *frames*, o *frame* de comando é fornecido como um serviço originário da subcamada MAC com interface para a camada física. Este *frame* tem por intuito comunicar a intenção na rede e será transmitido somente em *slots* ociosos na parcela ativa do *superframe*, denominada período de acesso de conteção (CAP), ou em tempos determinados para comunicação sem uso da estrutura de *superframe*.

O único *frame* de comando que os coordenadores não são capazes de transmitir ou receber é o de requisição para reserva de períodos de transmissão (GTS) no canal. Períodos de transmissão reservados ocorrem na outra parte ativa do *superframe* denominada período livre de contenção (CFP).

4 Simuladores Disponíveis para estudo

A área de redes *ad-hoc*, apesar dos avanços, ainda requer muita pesquisa. Alguns pontos como desenvolvimento de protocolos de roteamento, eficiência de comunicação e gerência de energia, são grandes desafios enfrentados pelos pesquisadores e projetistas de redes móveis, por exemplo.

Além disso, ao contrário de redes trandicionais que possuem algoritmos e protocolos legados, em RSSF o que existe são inúmeros protocolos desenvolvidos e adaptados para aplicações específica, com características específicas. Mesmo que este cenário esteja passando por um processo contínuo de padronização com o intuito de facilitar a integração de dispositivos fabricados por empresas distintas, é pouco provável que determinado protocolo seja sempre uma solução ótima para inúmeras circunstâncias.

Estes fatores fazem com que seja necessário efetuar simulações para análise de performance em RSSF e estimar se os parâmetros escolhidos são realmente os mais indicados para a maximização do tempo de vida desta rede. Desta forma, os simuladores surgem como ferramentas de subterfúgio para validação de eficiência e desempenho das soluções propostas para estas redes, através da replicação do ambiente desejado, mesmo que com grau de complexidade bem menor.

4.1 Simuladores

É fato que o nível de detalhamento fornecido por ferramentas de simulação é inferior ao contexto real no quesito de variáveis de influência, até mesmo para se simplificar a simulação. Isto permite que as variáveis problema sejam discriminadas e exaustivamente testadas até que resultados satisfatórios sejam obtidos e, logo em seguida, submetidas a testes reais em ambientes complexos para medições físicas; reduzindo-se, assim, substancialmente os custos de projeto.

Alguns dos simuladores encontrados e disponíveis para pesquisas, dentre estes, alguns já

em desuso por diversos fatores, como a falta de investimento em pesquisas de aperfeiçoamento; serão resumidamente abordados neste capítulo.

4.1.1 Atemu

Atemu é um simulador a nível de hardware para pequenos dispositivos. Ele emula as operações de muitos componentes em um nó sensor, como o processador, *timers* e rádio interface (THE MARYLAND HYBRID NETWORKS CENTER - HYNET, 2004).

Atuando, principalmente, como um emulador para sistemas baseados nos processadores AVR. Porém, é compatível com outros dispositivos da plataforma MICA2, podendo ser utilizado diretamente por desenvolvedores de software voltados para o sistema operacional TinyOS (THE MARYLAND HYBRID NETWORKS CENTER - HYNET, 2004).

Este simulador é, particularmente, usado em estudos de monitoramento de consumo de energia ou que envolvam nós heterogêneos em alguma rede, em termos de software e hardware. Sendo uma vantagem para desenvolvedores ligados ao TinyOS, já que o simulador nativo desta ferramenta, o TOSSIM, não dá suporte à redes heterogêneas.

O projeto encontra-se parado na versão 0.4, lançada em 31/03/2004.

4.1.2 GloMoSim

O GloMoSim (*Global Móbile Information Systems Simulation Library*) é um projeto da UCLA (*University of Califórnia, Los Angeles*).

No GloMoSim é possível construir ambientes de simulação escaláveis para sistemas de redes com e sem fio, sendo possibilitado pelo uso da capacidade de simulação discreta de eventos paralelos providos pelo Parsec (GLOMOSIM, 2000).

O Parsec (*Parallel Simulation Environment for Complex Systems*) é um linguagem de simulação baseada em C, desenvolvida pelo Laboratório de Computação Paralela da UCLA, para execução seqüencial e paralela de modelos de simulação de eventos discretos (PARSEC. . . ,).

Segundo GloMoSim (2000), a abordagem usual de construção de sistemas neste simulador é similar à arquitetura OSI de sete camadas, com o intuito de facilitar a integração de modelos construídos em diferentes camadas e até mesmo por diferentes pessoas. Já existem protocolos e outros tantos também podem ser construídos, bastando que o desenvolvedor se familiarize com a linguagem Parsec ou mesmo construa o protocolo em linguagem puramente C (com algumas funções Parsec de gestão de tempo).

A última versão disponibilizada para download foi a 2.0, liberada em dezembro de 2000. A partir daí, este produto seguiu para linha comercial com o nome QualNet. Mesmo estando atualmente em linha comercial, licenças gratuitas são avaliadas e disponibilizadas para pesquisa em universidades.

4.1.3 SHOX

Shox é um simulador escalável para redes *ad hoc* sem fio e escrito em Java. Os passos para instalação e configuração são fáceis, fornecendo resultados de simulação em um curto intervalo de tempo. Ao contrário de outros simuladores de propósitos gerais, como NS-2 e OMNet++, que requerem um enorme esforço para o aprendizado e treinamento antes dos primeiros resultados serem obtidos, ShoX possui uma sequência de passos muito simples de usar e fazer cenários de simulação usuais dentro de uma projeção gráfica de rede com centenas de nós com uma mobilidade específica, gerando certos padrões de tráfico muito fáceis de implementar.

Além dessas vantagens listadas, destacam-se a facilidade de projeção gráfica de arquitetura, extremamente flexível e limpa, na qual o desenvolvimento de novos modelos e protocolos é tão fácil como uma subclasse e extensível a superclasses existentes, o que dá uma orientação imediata quanto a interface que é requerida pelo kernel da simulação; um forte suporte a GUI para visualização dos nós, movimentos, roteamento de mensagens, etc; e uma coleção de dados estatísticos e geração automática de representações gráficas da simulação (SHOX, 2009).

Shox é um simulador de código aberto, sendo sua última versão disponibilizada para download a 1.0, liberada no dia 29 de janeiro de 2008. Porém, esta ferramenta até a presente data não possui suporte aos padrões mais indicados para redes de sensores.

4.1.4 OMNet++

OMNet++ é um simulador popular no meio acadêmico, isto devido ao seu modelo de código aberto e abundante documentação online.

Ele é um simulador de redes com eventos discretos, cuja programação é orientada a objetos, permitindo a criação de modelos hierárquicos a partir de módulos mais simples criados em C++ usando classes definidas nas suas bibliotecas (OMNET++..., 2010).

Áreas de aplicações específicas são servidas por vários *frameworks* e modelos de simulação, a maioria deles de código aberto (OMNET++..., 2010). Esses modelos são desenvolvidos completamente independente do OMNet++, e seguem seus próprios ciclos de lançamento.

Até a presente data, o simulador encontra-se na versão 4.1, liberada no dia 14 de junho de 2010.

4.1.5 TOSSIM

O TOSSIM (*TinyOS Simulator*) é um simulador discreto de eventos para RSSF que compila código TinyOS diretamente. Ao executar, eventos da fila de eventos (organizado por chegada) são retirados e executados. Dependendo do nível da simulação, eventos podem representar interrupções de hardware ou eventos de alto nível do sistema (TOSSIM. . . , 2009).

Este simulador é uma biblioteca para TinyOS que permite que usuários depurem, testem e analisem algoritmos em ambientes controlados, ao invés de compilar a aplicação do TinyOS diretamente para o dispositivo sensor. Em contrapartida, ele não permite que ambientes heterogêneos sejam montados, ou seja, cada nó sensor executa o mesmo programa TinyOS e possui as mesmas características físicas.

TOSSIM possui credibilidade no meio acadêmico, sendo amplamente citado em trabalhos como ferramenta de apoio na definição de novos projetos devido a sua biblioteca extensa e maleável. Porém, demanda considerável tempo para aprendizagem e adaptação ao ambiente de simulação até conseguir gerar alguns resultados.

A última liberação do TinyOS com correções para o TOSSIM foi na versão 2.1.1 e data do dia 20 de janeiro de 2010.

O TinyOS

O TinyOS é um sistema operacional de código aberto, projetado para dar suporte a aplicações de propósito geral em redes de sensores. Mais especificamente, ele se destina a suportar operações de intensa concorrência requeridas por sensores ligados em rede com requisitos mínimos de hardware. Foi desenvolvido na Universidade da Califórnia, em Berkeley (TINYOS, 2009).

Este sistema operacional possui uma arquitetura baseada em componentes. Os componentes desta biblioteca que melhor atenderem às especificidades da aplicação são selecionados pelo projetista e combinados. O que implica em um código fonte com o mínimo necessário à aplicação, desenvolvido com abstração de alto nível, reutilizável e suficientemente pequeno para compatibilizar com às restrições de memória impostas pela plataforma na qual irá rodar.

A biblioteca de componentes do TinyOS inclui protocolos de rede, serviços distribuídos,

drivers de sensores, ferramentas de aquisição de dados, podendo ser usadas como são ou redefinidas em ajuste fino aos aspectos peculiares à cada aplicação (TINYOS, 2009).

Porém, programação TinyOS pode ser um desafio, por requerer o uso de uma nova linguagem: nesC (*network embedded system C*).

O nesC

O nesC é uma linguagem orientada a componentes com um modelo de execução baseado em eventos. Em alguns aspectos, componentes nesC são similares a objetos, no sentido que ambos encapsulam estado e interagem entre si através de interfaces bem definidas.

Porém, componentes nesC usam um espaço puramente local, no qual além das declaração das funções que ele executa (*command*), as funções que ele chama também devem ser declaradas (*event*). Isto é, um componente pode referenciar somente variáveis pertencentes ao seu espaço local e isto inclui a declaração das funções que ele irá chamar.

No uso de componentes, existem três abstrações computacionais (TINYOS, 2009):

- *commands*: requer do componente que o mesmo execute algum serviço;
- *events*: sinaliza ao componente que a execução de algum serviço chegou ao fim. Não devem requerer grande quantidade de processamento, já que a estratégia de execução é não bloqueante (permanece executando até que toda a tarefa tenha sido cumprida). Não há preempção de execução entre *commands* e *events*;
- *tasks*: permitem que *components* executem processamento de propósito geral em '*background*' em um aplicativo. Uma tarefa é uma função que um componente sinaliza ao TinyOS que a mesma deve rodar mais tarde, em vez de agora. *Tasks* representam concorrência interna no componente e o TinyOS gerencia as mesmas *tasks* com um escalonador de política FIFO (*First In First Out*) não preemptiva. Funções de *events* e *commands* preemptam *tasks*.

Existem dois tipos de componentes: configuração e módulo. Configurações definem como os componentes são ligados e os módulos como eles são implementados.

Tomaremos como exemplo a aplicação Blink, inclusa na biblioteca padrão do TinyOS e explicitada na figura 4.1. Trata-se uma aplicação básica que alterna a ligação e desligamento de três *leds* periodicamente de um nó. A aplicação é composta por dois componentes: um módulo "BlinkM.nc"e uma configuração "Blink.nc".

Configuração - Blink.nc

```
configuration BlinkAppc{
}
implementation{
  components MainC, BlinkC, LedsC;
  components new TimerMilliC() as Timer0;
  components new TimerMillic() as Timer1;
  components new TimerMilliC() as Timer2;

  BlinkC -> MainC.Boot;

  BlinkC.Timer0 -> Timer0;
  BlinkC.Timer1 -> Timer1;
  BlinkC.Timer0 -> Timer2;
  BlinkC.Leds -> LedsC;
```

Módulo - BlinkM.nc

```
#include "Timer.h"

module BlinkC @safe(){
   uses interface Timer<TMilli> as Timer0;
   uses interface Timer<TMilli> as Timer1;
   uses interface Timer<TMilli> as Timer2;
   uses inteface Leds;
   uses interface Boot;
}
implementation{
   uint8_t counter = 0;

   event void Boot.booted(){
      call Timer0.startPeriodic( 250 );
      call Timer1.startPeriodic( 500 );
      call Timer2.startPeriodic( 1000 );
   }
   event void Timer0.fired(){
      call Leds.led1Toggle();
   }
   event void Timer1.fired(){
      call Leds.led0Toggle();
   }
   event void Timer2.fired(){
      call Leds.led2Toggle();
   }
}
```

Figura 4.1: Exemplo de aplicação utilizando a linguagem nesC

O módulo é responsável por englobar a implementação da aplicação, enquanto a configuração descreve quais componentes são necessários e como eles devem ser interconectados.

Na configuração, foram declarados o uso de 4 distintos componentes: *LedsC*, *TimerMillic*, *MainC* e *BlinkC*. Além de serem redefinidas algumas interfaces, como no trecho *BlinkC.Leds - LedsC* que explicita que a implementação provida por *BlinkC* para a interface *Leds* é redefinida pela implementação provida pelo componente *LedsC*.

No módulo deve-se declarar quais funções são providas (*provides*) e quais são usadas (*uses*). No exemplo em questão, não há declarações de funções providas, logo ele não pode ter nenhuma função sua chamada por outro componente. Porém, ele faz chamadas de funções a 3 componentes diferentes, todas preconizadas pela palavra chave *call*.

Em linguagens tradicionais, referenciar uma função requer uma referenciação a um único nome no espaço global do programa, de modo que o código usa um nível de referenciação indireta (ponteiro). O nesC tem uma abordagem diferente, o código é dividido em componentes, unidades discretas de funcionalidade que são ligados estaticamente. O compilador sabe exatamente que funções são chamadas e o onde (TINYOS, 2009).

Por fim, o código fonte dessa aplicação, somado aos demais componentes da biblioteca do TinyOS podem ser compilados pelo compilador nesC, resultando em programas na linguagem C. Para executá-la, basta compilá-la para o hardware em questão ou usar um simulador para tanto, vide figura 4.2.

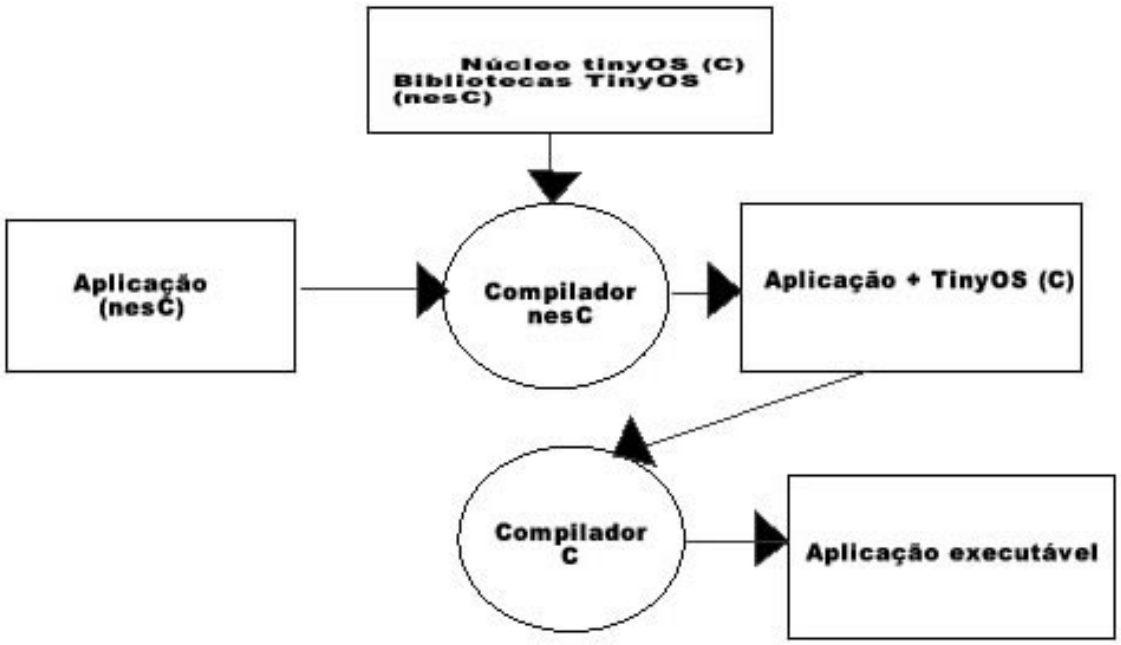

Figura 4.2: Modelo de montagem de programa

4.1.6 NS-2

NS (*Network Simulator*) é um simulador de eventos discretos voltado para pesquisas em redes no geral. É um simulador muito popular no meio acadêmico, sendo citado em diversos trabalhos como ferramenta de apoio e tendo o seu uso motivado, principalmente, por ser de código aberto e gratuito.

A programação de seu conteúdo é feito em duas linguagens: C++ para estruturas básicas como protocolos, agentes, etc.; e OTCL (*Object-oriented Tool Command Language*) para uso

como *frontend*.

NS oferece substancial suporte para simulação de TCP, roteamento e protocolos multicast para redes cabeadas ou não. Além de poder ser usado como emulador capaz de interagir com uma rede real (DARPA AND USC/ISI AND XEROX PARC AND LBNL AND UNIVERSIDADE DE BERKELEY, 1995). Esta ferramenta incorporou ao seu pacote de instalação suporte à simulações WPAN para o padrão 802.15.4 a partir da versão 2.33.

Na versão 2.34 já é possível o uso do módulo opcional de visualização gráfica NAM para simulações WPAN.

O último lançamento data de 17 de junho de 2009 e encontra-se na versão 2.34.

4.2 Comparativo entre simuladores

A tabela 4.1 faz um comparativo entre os simuladores aqui abordados segundo alguns critérios levantados em Leithardt. (2008) e outros tomados como relevantes neste trabalho.

- disponibilidade do código fonte do simulador;
- disponibilidade de qualquer tipo de documentação que permita uma avaliação mais minuciosa;
- comprometimento dos desenvolvedores, ou seja, se há um roadmap para lançamento de novos releases, se há desenvolvedores engajados na correção de bugs, etc;
- plataforma para a qual o simulador foi desenvolvido;
- linguagem de programação utilizada.
- um modelo que permita averiguar o consumo de energia;
- um modelo que permita definir o meio de comunicação utilizado pelos nós sensores;
- um modelo que permita a movimentação dos nós sensores, ou seja, a mudança constante da topologia;
- data da última atualização disponível na corrente data de escrita deste trabalho.

[1] NE: informação precisa não foi encontrada.

Tabela 4.1: Característica de Simuladores RSSF

	ns-2	**Atemu**	**SHOX**	**GloMoSim**	**OMNet++**	**Tossim**
Código fonte	Sim	NE [1]	Sim	Sim	Sim	Sim
Documentação	Sim	NE [1]	Sim	Sim	Sim	Sim
Roadmap	Sim	Não	Sim	Não	Sim	Sim
Portabilidade	Existem versões para diversos SOs, dentre os quais, FreeBSD, Linux, SumOS, Solaris e para a família Windows.	Linux/ Windows	Linux, MAC e Windows	Linux/ Windows	Linux/ Windows	Linux/ Windows
Linguagem utilizada	C++ para estrutura básica (protocolos, agente, etc) e OTCL pra uso com frontend	Assembler/ nesC	Java	C	C++	Python ou C++
Modelo de consumo de energia	Sim	Sim	NE [1]	Sim	Sim	Sim
Modelo de comunicação	Sim	Sim	Sim	Sim	Sim	Sim
Modelo de topologia dinâmica	Sim	NE [1]	NE [1]	NE [1]	Sim	Sim
Ano da última atualização	2009	2004	2008	2000	2010	2010

Com base nas informações contidas na tabela comparativa dos simuladores aqui abordados, observa-se que os mais indicados para uso em simulações de redes de sensores são o ns-2, OMNet++ e Tossim. Dentre os indicados, o ns-2 foi o escolhido para efetuar as simulações deste trabalho com base na simplicidade e robustez do ambiente por ele providas.

5 *Simulação e resultados*

Construções civis estão sujeitas a alterações no seu entorno ou mesmo em suas próprias estruturas devido a fatores químicos e físicos que acabam por degradar o projeto arquitetônico. Em estruturas nas quais lida-se com deformações e movimentos críticos, medidas preventivas de segurança com base em relatórios periódicos da saúde da construção devem ser tomadas.

Pontes de legado, por exemplo, podem ter sua estrutura danificada com o tempo e até ruírem. Porém, uma construção não rui sem antes enviar sinais claros de degradação. Estes sinais podem ser monitorados e analisados constantemente de forma automática e em uma freqüência muito mais segura do que poderia ser feito por uma pessoa.

A análise freqüente destes dados pode permitir monitorar movimentos imprevistos que estejam fora da oscilação natural da estrutura e que não foram previstos em projeto. Isto permite que o responsável pela obra atue com medidas corretivas ou, em casos nos quais o colapso é iminente, interdite o acesso da estrutura por transeuntes. Este último pode ser feito até mesmo remota e automaticamente.

Casos como o desastre da ponte de Tacoma Narrows entraram pra história da engenharia. Esta ponte foi construída em 1940 nos Estados Unidos e ficou famosa por sua grande oscilação e conseqüente colapso na presença de ventos considerados fracos, porém, constantes. Este fato gerou um efeito físico conhecido como ressonância que amplificou a freqüência natural da ponte, impondo-a a movimentos intensos devido ao gradual acúmulo de energia. A diferença para as arquiteturas de estruturas posteriores a este evento é que agora este tipo de influência do meio já é sabido e sistemas para atuarem na dissipação de energia são previstos em projeto (TACOMA, 2009).

Um outro desastre, dessa vez com vítimas fatais, foi o ocorrido no Rio Grande do Sul referente à ponte do Rio Jacuí, entre Agudo e Restinga Seca no km 191 da ERS-287. Ela ruiu no dia 5 de janeiro de 2010 e foi noticiado em diversos jornais da época. A mesma possuía 314 metros de comprimento e estimados 8 metros de largura. No caso desta ponte, porém, o abalo das estruturas se deu devido às fortes chuvas e força das águas que acabaram por arrastar

significativamente o fundo do rio. Isto levou à queda dos pilares e aproximadamente 100 metros de estrutura ruíram em conjunto.

Assim como estas pontes, tantas outras ainda em atividade são projetos muitas vezes antigos e não estão preparadas para a força das chuvas nos dias atuais e nem muito menos para outras tantas influências externas do meio não sabidas no momento de sua projeção.

Porém, aplicações simples de monitoramento, como as providas por redes de sensores, podem evitar que casos como estes se repitam. Fortes vibrações que destoem da já conhecida vibração natural da estrutura podem ser captadas por meio de sensores, permitindo-se reagir de forma corretiva ou mesmo preventiva, auxiliando nas vistorias periódicas ou interdição desta estrutura.

Motivado por estes fatores e os já existentes recursos técnicos para provimento da solução, uma simulação simples para elucidar o uso de redes de sensores em uma aplicação concreta será abordada: redes de sensores sem fio para monitoramento de estruturas de pontes. A simulação consiste de 3 cenários, sob a topologia ponto-a-ponto, utilizando o protocolo IEEE 802.15.4 para provimento de comunicação.

Como visto no capítulo 2, o padrão IEEE 802.15.4 foi criado para atender redes nas quais há baixa transmissão de dados a curtas distâncias, baixo consumo de energia e baixo custo. Além de ser um padrão suportado por dispositivos de baixo custo graças aos esforços de uma associação de empresas para a padronização de uma tecnologia de comunicação sem fio: a *ZigBee Alliance* com a criação do padrão ZigBee, que usa o padrão 802.15.4 para as camadas inferiores e direciona seus esforços para a padronização das camadas superiores (ZIGBEE ALLIANCE, 2009).

A escolha deste padrão para ilustrar este cenário simplificado de simulação de rede de sensores, deve-se a estes fatores e à sua representatividade junto aos simuladores como o ns-2 e ao sistema operacional TinyOS, logo, por simuladores como o Tossim.

O ns-2 foi o simulador escolhido para este trabalho. Nele, o módulo responsável por dar suporte ao padrão IEEE 802.15.4 é o WPAN. O código C++ que abrange este módulo contém aproximadamente 12800 linhas e foi desenvolvido por Jianliang Zheng e Myung Lee como pode ser verificado na biblioteca do próprio programa. Esta extensão do 802.15.4 para o ns-2 está disponível para as versões ns-2.26 e 2.27 e já vem inclusa no pacote ns-allinone-2.33 (DARPA AND USC/ISI AND XEROX PARC AND LBNL AND UNIVERSIDADE DE BERKELEY, 1995).

O script para simulação, em anexo, tomou com referência primária o exemplo criado por

Jianliang Zheng (RAMACHANDRAN, 2006). Os parâmetros, porém, foram escolhidos e variados de acordo com o levantamento de requisitos para esta aplicação.

Para a simulação, usou-se uma máquina executando o sistema operacional GNU/Linux Ubuntu 8.10 com as seguintes características:

- memória de 1 GB;
- HD de 80 GB;
- processador AMD Athlon XP 2400+ 1.96 GHz;
- simulador de rede NS-2, versão 2.34;
- Tcl versão 8.4.18;
- GNU *Compiler Collection* (GCC) versão 4.3.2.

5.1 Configurações de simulação

A principal dificuldade na simulação de uma rede é a configuração dos parâmetros. Esta tarefa é *hard code* e totalmente relacionada ao cenário escolhido.

Como não foi utilizado nenhum modelo de interferência nos testes, o desempenho da rede fica a cargo apenas dos parâmetros escolhidos. Assim, a variação dos resultados está relacionada à topologia e à competição entre os próprios nós pelo acesso ao meio.

Abaixo seguem breves descrições sobre os parâmetros usados. Para mais informações, consulte (SIMULATION. . . , 2009).

5.1.1 Parâmetros de energia

Os parâmetros de energia foram escolhidos com base na especificação de hardware do processador e plataforma de rádio (MPR400CB) do popular nó MICA2 da Crossbow. Este foi desenhado especificamente para redes de sensores profundamente embarcados (MICA2, 2009).

Como o intuito deste trabalho é apresentar as restrições em redes de sensores, este dispositivo foi escolhido devido aos seus recursos limitados, sendo assim mais fácil explicitar a problemática do consumo de energia. Ressaltando que o MICA2 pode executar o XMesh, uma pilha de protocolos da Crossbow construída em cima padrão 802.15.4 no sistema operacional TinyOS (CROSSBOW TECHNOLOGY, INC., 2005). Logo, ele pode ser usado de acordo

com os propósitos acima e segundo os recursos disponíveis para esta simulação: ns-2 e padrão 802.15.4.

- Uso de modelo de energia (*EnergyModel*). Os parâmetros abaixo só serão requeridos quando o modelo for explicitado na configuração do nó.

  O nó coordenador será o único na rede a não possuir restrições quanto ao consumo de energia, podendo ser alimentado por energia contínua. Logo os parâmetros de consumo de energia só serão relevantes para os demais dispositivos nessa simulação.

  1. Energia inicial (*initialEnergy*). O mote MICA2 usa 2 pilhas para fornecimento de energia para a CPU/rádio. Não é a intenção esgotar os recursos de energia na simulação, para tanto, o valor de energia estipulado foi de $1000W/s$, sem perda de representatividade alguma dos resultados.
     $initialEnergy = 1000Joules = 1000W/s$.
  2. Energia gasta na transmissão (*txPower*). É a energia consumida pelo tranceptor para transmitir um pacote de dados.
     $txPower = 27mA * 3V = 81mW$
  3. Energia consumida na recepção (*rxPower*). É a energia consumida para receber um pacote de dados.
     $rxPower = 10mA * 3V = 30mW$
  4. *sleepPower* ou *idlePower*. A especificação do *mote* indica que o mesmo consome menos do que $15x10^{-6}A$ para o processador ocioso e para o rádio de transmissão $1x10^{-6}A$ quando ocioso.
     Radio multi-canal ocioso:

$$\begin{aligned} &< 1x10^{-6}A \\ < 1x10^{-6}A * 3V &= 3x10^{-6}W \\ &< 3x10^{-6}W \end{aligned} \tag{5.1}$$

     Processador dormindo:

$$< 15x10^{-6}A < 15x10^{-6}A * 3V = 45x10^{-6}W < 45x10^{-6}W$$

     Dentre os dois consumos reduzidos de energia, tomou-se os valores para *idlePower* e *sleepPower* como sendo, respectivamente, sem atuação no canal de transmissão (equação 5.1) e sem processamento (equação 5.2).

$idlePower = 3x10^{-6}W$

$sleepPower = 45x10^{-6}W$

- Potência de transmissão ($pt\_$). A potência de rádio freqüência do MICA2 pode variar de -20 a $5dBm$. Já o padrão estipula que um transmissor deve ser capaz de transmitir pelo menos $-3dBm$. Para esta simulação, escolheu-se o valor $0dBm$.

  $pt\_ = 0dBm = 0.001W$.

- Potência mínima requerida do pacote para uma recepção com sucesso (*Receiver Threshold*, $RXThresh\_$). O valor do *receivethreshold* é o nível mínimo de energia de um pacote para sua transmissão ser detectada com sucesso. Se a potência do sinal recebido é menor que esse limite, o pacote não poderá ser detectado e será descartado pela camada física.

  A sensibilidade do transceptor do MICA2 é de no mínimo $-98dBm$ para os pacotes recebidos. Porém, o padrão IEEE 802.15.4 especifica que o limite inferior de sensoriamento de rádio freqüência é de $-92dBm$ para a frequência do canal de $868/915MHz$ e $-85dBm$ para a de $2.4GHz$.

  A frequência de operação para o dispositivo escolhido foi de $915MHz$. Contudo, neste cenário, não há necessidade de recebimento de pacotes à distâncias tão grandes, ou no caso, com energias tão baixas representadas por este limite inferior. Para tanto, escolheu-se um corte superior com um valor que permitisse maior número de saltos dos dados de um dispositivo direcionados ao coordenador da PAN: $-21dBm$.

  $RXThresh\_ = -21dBm \approx 7.94328x10^{-6}W$

- Potência mínima para detecção do pacote na portadora (*Carrier Sensing Threshold*, $CSThresh\_$). Descreve o alcance de sensoriamento do nó.

  Quando o pacote recebido tem nível de energia abaixo do limite de energia para recepção ($RXThresh\_$), mas acima do limiar de detecção da portadora ($CSThresh\_$), quer dizer que o receptor está dentro do raio de sensoriamento da portadora. Porém, a decodificação deste pacote recebido não é garantida. Para se evitar estes casos, configurou-se os valores do $CSThresh\_$ como sendo pelo menos o mesmo valor do $RXThresh\_$, sendo comum esta medida em trabalhos dessa natureza de simulação.

  $CSThresh\_ = RXThresh\_ = -21dBm \approx 7.94328x10^{-6}W$.

- Limite de captura (*Capture Threshold*, $CPThresh\_$). Durante a transmissão de pacotes, é possível que 2 pacotes possam chegar simultaneamente (isto é, colidirem) na recepção do

nó. É um fenômeno comum. O limite de captura ($CPThresh\_$) determina se um pacote é recebido com sucesso a partir da comparação de energia de ambos os pacotes recebidos.

Assim, se a relação entre os pacotes for maior que o $CPThresh\_$, o pacote com maior nível de energia será escolhido. Senão, ambos os pacotes serão descartados. O valor de $10dB$ é amplamente aceito na comunidade para simulações.

$CPThresh\_ = 10dB$

- Freqüência do canal ($Freq\_$). A taxa nominal de velocidade transmissão do meio depende da banda de freqüência, vide figura 5.1.

Freqüência da banda (MHz)	N° de canais	Taxa de transmissão (kb/s)
868-868.6	1	20
902-928	10	40
2400-2483.5	16	250

Figura 5.1: IEEE 802.15.4: bandas de frequência e taxa de transmissão

O mote MICA2 suporta frequência do canal nas faixas de $868/916MHz$. Para esta simulação, escolheu-se o valor de $916MHz$. Isto reduzirá a competição dos dispositivos pelo canal de transmissão e, em um ambiente real, as interferências, uma vez que esta frequência possui 16 canais enquanto a primeia apenas 1.

$Freq\_ = 916MHz$

- Fator de perda ($L\_$). Nesta simulação não se considerou fatores de atenuação de sinal, como ruídos no geral. Para tanto, usou-se $L\_$ igual a 1.0.

5.1.2 Parâmetros gerais para uso do protocolo

- O comando *$ns_ puts-nam-traceall {# nam4wpan #}* informa ao *nam* que este é um arquivo trace para WPAN, sendo imprescindível este cabeçalho especial.

- Tráfego. Não se gerou uma fonte de tráfego específica para este trabalho. Usou-se uma fonte de tráfego de CBR (*Constant Bit Rate*), com intervalo de 3s para pacotes de 10 bytes de tamanho a fim de representar uma monitoração de dados com periodicidade fixa e constante.

  É importante frisar que o número de nós transmitindo os pacotes nas simulações não varia, sendo composto por 16 nós. Estes sempre estarão dispostos na borda da área de monitoramento.

- Tipo de canal (*chan*). O meio físico escolhido foi o sem fio (*Channel*/*WirelessChannel*).

- Modelo de propagação de rádio (*prop*). O ns2 permite alguns modelos de propagação de rádio: *free space*, *two-ray ground*, *Ricean*, *Rayleigh* e *shadowing models*. Estes modelos são usados para prever a potência do sinal recebido de cada pacote.

  O modelo escolhido para esta simulação foi o *two-ray ground*. Este modelo confere uma previsão mais precisa de propagação por incluir tanto o caminho direto entre dois nós como o caminho de reflexão do solo.

- Interface de rede (*netif*). A interface de hardware escolhida para acesso ao canal é implementada pela classe *Phy*/*WirelessPhy*/802_15_4.

- Interface MAC (*mac*). O protocolo escolhido é o IEEE 802.15.4, implementado pela classe *Mac*/802_15_4.

- Tipo de fila (*ifq*). Fila implementada pela classe Queue/DropTail/PriQuee. Aqui, pacotes de roteamento possuem uma prioridade maior. Porém, a técnica de gerenciamento básica da fila é FIFO (*First In First Out*). Basicamente, o pacote que entra na fila será atendido primeiro. Quando a fila atinge o seu limite, o último pacote a entrar na fila será descartado (*buffer* limitado).

- Camada de enlace (*link layer, ll*). É a camada responsável por simular os protocolos de enlace dos dados. Outra importante função da camada de enlace é definir o endereço MAC de destino no cabeçalho MAC do pacote.

- Antena (*ant*). Antenas podem ser configuradas como omnidirecionais ou direcionais no ns2. Para este cenário foram usadas antenas omnidirecionais.

- Protocolo de roteamento (*rp*). O padrão IEEE 802.15.4 prevê soluções apenas para a camada física e subcamada MAC. Por isso, neste cenário usou-se um protocolo comum de roteamento para redes *ad-hoc*.

- Tamanho do *buffer* (*ifqlen*). O valor escolhido para o tamanho do *buffer* foi de no máximo 100 pacotes na fila. Foi um valor aleatório, porém tomou-se como base valores observados em outras simulações de redes de sensores.

- Área de simulação (X x Y). Para simplificar a depuração do resultados, um cenário de apenas 20m x 4m foi usado.

- Número de nós fixos (nn). Por se tratar de um cenário de monitoração de uma ponte, os nós serão fixos e não móveis. A quantidade usada aqui foi de 17 e 25 nós, contando com o coordenador da rede.

5.2 Resultados obtidos

A partir da saída gerada nas simulações do ns2, estipulou-se dois critérios para possibilitar a avaliação qualitativa dos cenários: baixo consumo de energia e aceitável taxa de entrega. Isto possibilitará avaliar a rede quanto ao seu tempo de vida útil e eficiência na entrega dos dados. Porém, aqui é tomado como prioritário o tempo de vida da rede.

O consumo de energia (CE) (equação 5.2) foi definido como a média de consumo entre todos os nós ao longo da simulação.

$$CE_{no} = \frac{Gasto\_de\_energia}{tempo\_de\_simulacao}$$

$$CE_{total} = \sum \frac{CE_{no}\_para\_todos\_os\_nos}{total\_de\_nos} \tag{5.2}$$

A taxa de entrega (TE) (equação 5.3) é resultado da razão do número total de pacotes CBR entregues com sucesso pelo total enviado até o momento da medição.

$$TE = \sum \frac{Total\_pacotes\_recebidos_{CBR}}{Total\_pacotes\_enviados_{CBR}} \tag{5.3}$$

Para as simulações, estipulou-se 8 cenários dentro de uma topologia ponto-a-ponto. Na concepção destes cenários, permutou-se a quantidade de nós (16 e 24, excetuando-se o coordenador da PAN), o método de comunicação (usando a estrutura de *superframe* ou não) e a escolha das atribuições de funções destes dispositivos (roteadores ou dispositivos finais). Observe a tabela 5.1 e a disposição dos nós na figura 5.2.

São 4 as topologias propostos para os nós roteadores na área de monitoramento: 16 nós roteadores em um total de 17 nós (figura 5.2(a)); 24 nós roteadores em um total de 25 nós (figura 5.2(b)); 8 roteadores dispostos em ziguezague em um total de 17 nós (figura 5.2(c)); 8 roteadores dispostos na fileira central em um total de 25 nós(figura 5.2(d)). Para melhor entender sobre como os nós estariam fixados em uma ponte, há uma representação elucidativa na figura 5.3 sobre estes cenários.

Tabela 5.1: Cenários definidos para as simulações

Usando estrutura de *superframe*	Cenário 1	16 nós; todos os nós trabalhando como roteadores.
	Cenário 2	16 nós; roteadores posicionados em ziguezague.
	Cenário 3	24 nós; todos os nós trabalhando como roteadores.
	Cenário 4	24 nós; apenas a fila central composta por roteadores.
Sem o uso de *superframe*	Cenário 5	16 nós; todos os nós trabalhando como roteadores.
	Cenário 6	16 nós; roteadores posicionados em ziguezague.
	Cenário 7	24 nós; todos os nós trabalhando como roteadores.
	Cenário 8	24 nós; apenas a fila central composta por roteadores.

Em comunicações com uso de *superframe*, os coordenadores são os responsáveis pela gerência da estrutura delimitada pelos *beacons* e por transmitirem os *beacons*. Os *beacons* permitem que os dispositivos se sincronizem e moralizem a comunicação.

Opcionalmente, a estrutura do *superframe* pode ser dividida em uma porção ativa e uma inativa, onde os nós ficam acordados ou dormindo, respectivamente (figura 5.4). Este recurso permite que a vida útil dos nós seja estendida, porém estes períodos de atividade e inatividade devem ser estipulados. Isto porque os intervalos dependem da aplicação, não existindo um valor correto para ser definido.

Sendo assim, as simulações com uso de *superframe* levantaram a necessidade de determinar quais seriam os intervalos nos quais os nós estariam ativos ou não. Os parâmetros que definem esta estrutura são o *beacon order* (BO) e o *superframe order* (SO).

O BO é usado para definir com que frequência os *beacons* serão transmitidos na rede e pode ir de 0 a 14, segundo o padrão IEEE Computer Society (2006). Ou seja, esse valor vai determinar a duração total de um *superframe*, já que o *superframe* tem seu início e fim delimitado por *beacons*.

O SO está relacionado à porção ativa do *superframe*. A porção ativa do *superframe* pode ter duração até no máximo igual ao tamanho total do *superframe*, ou seja, sem porção inativa. A relação entre o BO e SO é:

$0 \leq SO \leq BO \leq 14$

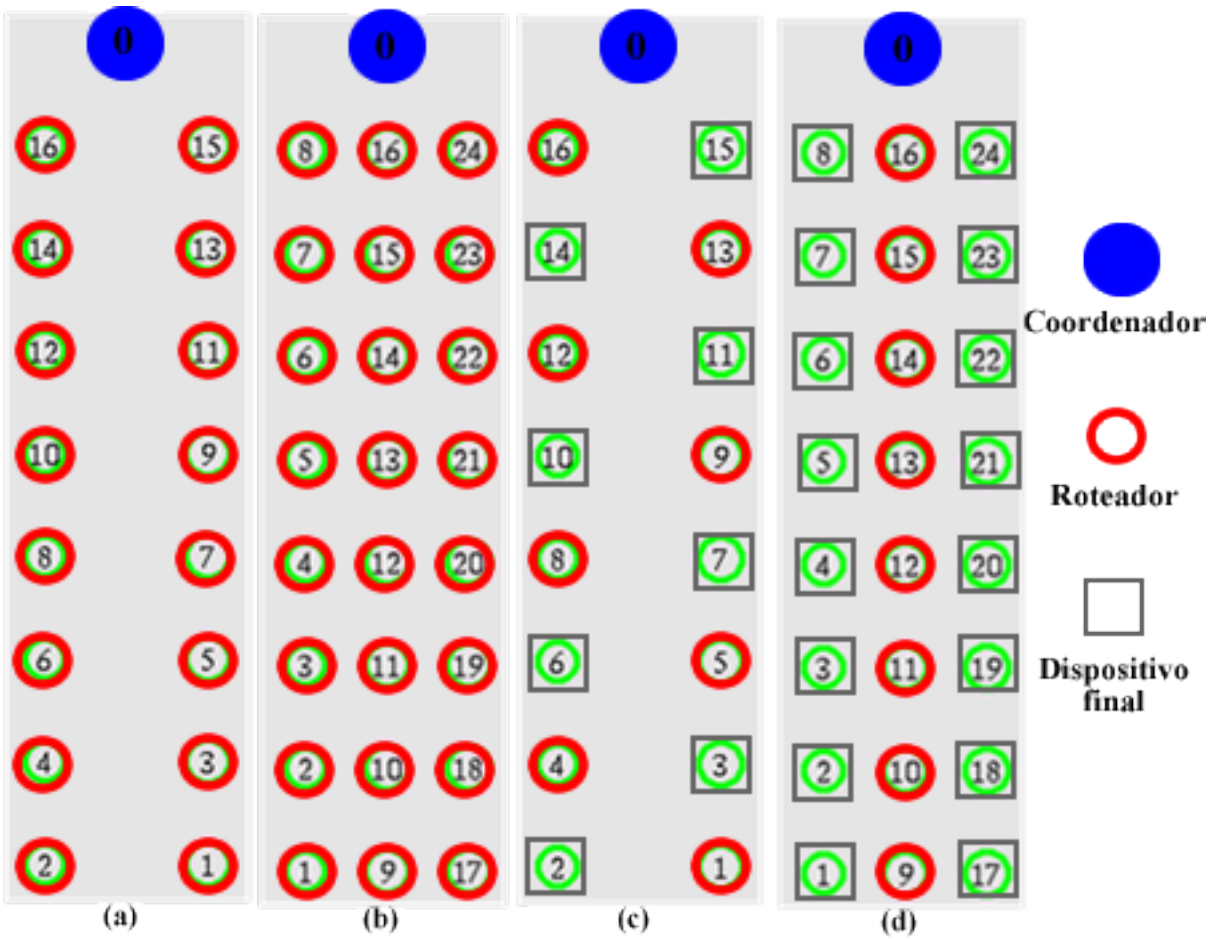

Figura 5.2: Posicionamento dos nós roteadores usado nas simulações

Para escolher as dimensões da estrutura *superframe* a serem aplicadas a cada um dos 4 cenários com beacons ativos, 8 simulações de 7200 segundos (ou unidade de tempo do ns2) foram realizadas.

Na tabela 5.2 estão os resultados obtidos. Com base neles e nos critérios anteriormente definidos para escolha de solução (consumo e entrega), a estrutura $BO = 6$ e $SO = 2$ se mostrou mais aplicável por ser a que consumiu menos recursos de energia dos dispositivos.

De posse desse valor, iniciaram-se as simulações para cada um dos 4 cenários com uso de *superframe*. Para o consumo, efetuaremos um corte inferior de uso de energia, selecionando os 2 cenários mais econômicos. Já para a taxa de entrega, os cenários com maior proporção de entrega de dados serão os selecionados. Os resultados obtidos podem ser observados nos gráficos 5.5 e 5.6.

Observa-se que os cenários com melhor economia no consumo de energia foram o de 16 nós, sendo todos roteadores de dados na rede, fechando ao fim da simulação em $3Watts$ de consumo médio de energia para cada dispositivo; e o de 24 nós, com roteamento parcial (roteamento

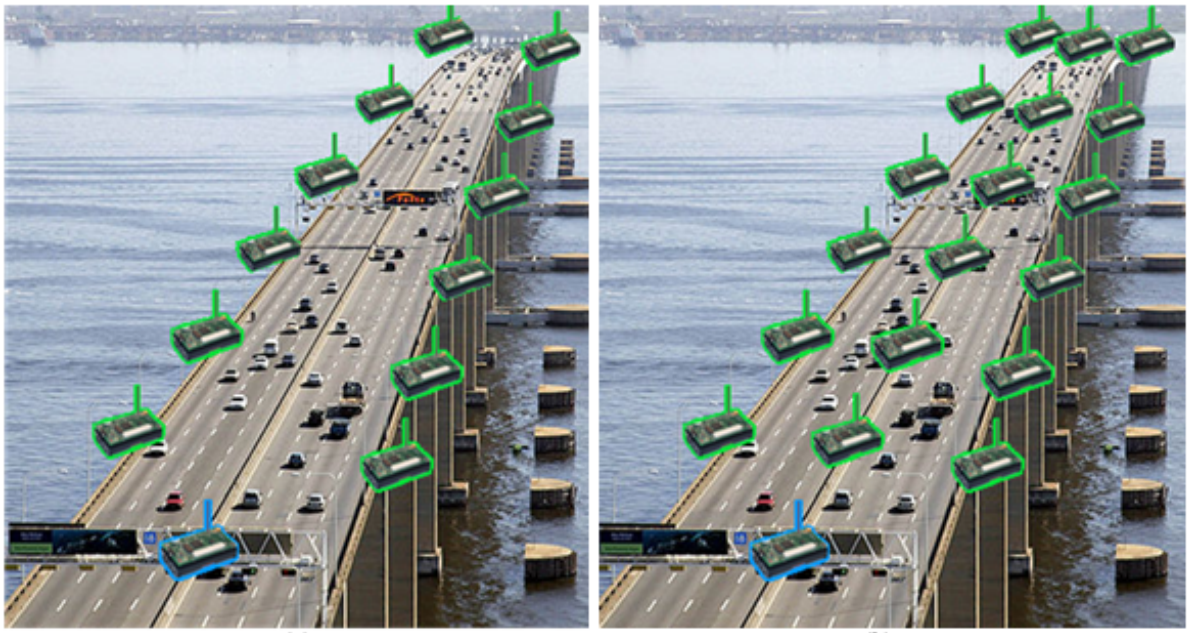

Figura 5.3: Esboço simbólico da disposição dos 17 nós (a) e 25 nós (b) sob uma ponte

Tabela 5.2: Levantamento de parâmetros para a estrutura de superframe

Parâmetros	Taxa de entrega (%)	Taxa de consumo (Watts)
BO=10 e SO=2	88.405459	0.899146
BO=10 e SO=5	94.229138	1.181390
BO=10 e SO=10	67.352025	6.393707
BO=6 e SO=2	96.458407	0.475769
BO=6 e SO=4	88.644129	6.542174
BO=6 e SO=6	64.593114	9.025572
BO=3 e SO=2	83.334537	10.020070
BO=3 e FO=3	61.547560	15.144937

central), fechando em aproximadamente $3.37 Watts$. Estes cenários se referem, respectivamente, aos cenários 1 e 4 acordados na tabela 5.1.

Em relação à taxa de entrega, o cenário 1 se mostrou muito eficiente, mantendo-se acima de 95% de proporção de entrega de pacotes de dados, sendo seguido pelo cenário 3. Este último manteve-se quase que invariavelmente na faixa de 88% de sucesso na entrega dos pacotes.

A escolha do melhor cenário usando a estrutura de *superframe* na comunicação foi o cenário 1, segundo os resultados obtidos. Mesmo ele tendo um número total de pacotes recebidos no destino menor que os outros cenários (tabela 5.3), ele foi o que atingiu maior proporção de entrega e também menor consumo médio de energia por nó. Isto quer dizer que dentre os cenários com uso de *superframe*, este tem maior garantia de entrega de dados e maior tempo de

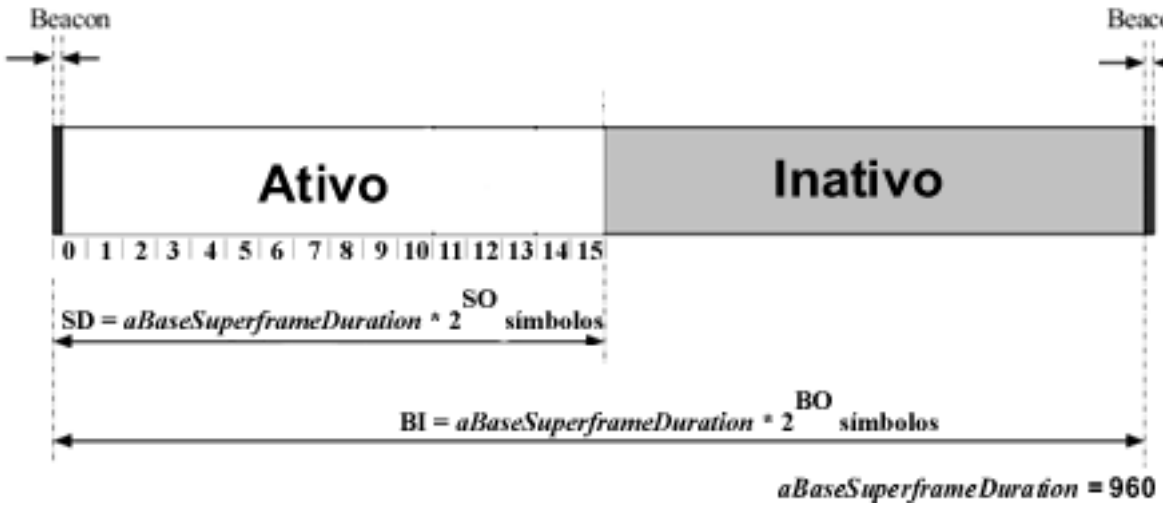

Figura 5.4: Estrutura do superframe e suas possíveis divisões de período

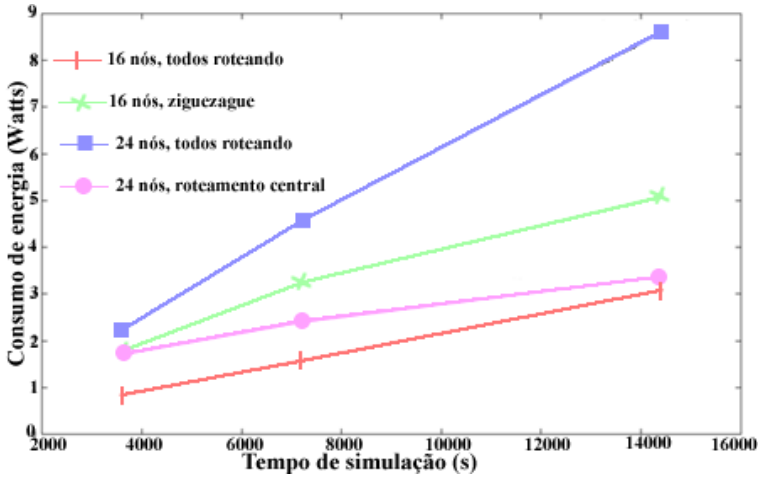

Figura 5.5: Consumo de energia x Tempo de simulação em redes com beacons ativos

vida útil dos nós, atendendo perfeitamente a proposta de monitoramento de uma estrutura que não exige grande massa de dados em um curto espaço de tempo, mas sim a longo prazo.

Os outros 4 cenários referem-se a redes sem o uso de *superframe* na comunicação. Para estas simulações, os valores de BO e SO não são válidos. Os resultados podem ser observados nas figuras 5.7 e 5.8. A comunicação nestes cenários se dá por disputa do canal, pura e simplesmente.

Dos 4 cenários simulados sem uso de *superframe*, mostraram-se mais eficientes ao fim da simulação os cenários 7 e 8 em economia de energia. Eles obtiveram, respectivamente, 1.37*Watts* e 1.57*Watts* por nó . Já para o sucesso de entrega de dados, os cenários 5 e 7 foram os melhores. O cenário 5 teve um pequeno ganho progressivo na qualidade de entrega dos dados ao longo da simulação, permanecendo entre a faixa de 75% e 79% de dados entregues com sucesso, enquanto que o cenário 7 manteve-se quase que constantemente a uma taxa de

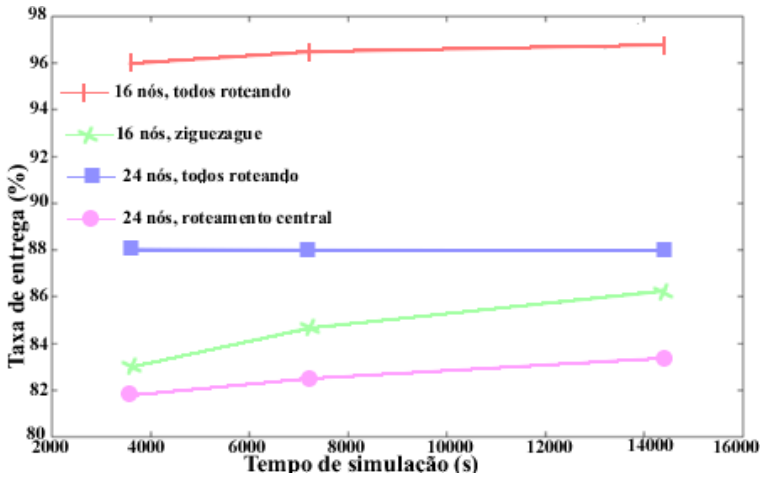

Figura 5.6: Proporção de entrega de pacotes x Tempo de simulação em redes com beacons ativos

Tabela 5.3: Número de pacotes enviados (CBR) x tempo de simulação em redes com beacons ativos

Tempo de simulação	Cenário 1	Cenário 2	Cenário 3	Cenário 4
3600	13124	49138	53590	40166
7200	25497	93081	114482	44239
14400	50219	147777	213604	46745

77% de entrega.

Tabela 5.4: Número de pacotes enviados (CBR) x tempo de simulação em redes sem beacons ativos

Tempo de simulação	Cenário 5	Cenário 6	Cenário 7	Cenário 8
3600	12374	20386	14964	15897
7200	21365	37741	16406	23267
14400	39285	59307	16406	27214

Com base nos resultados analisados, escolheu-se como melhor cenário para comunicação sem uso de *superframe* o cenário 8. Este cenário também possui baixa quantidade de dados transmitidos para uma faixa de tempo (tabela 5.4), porém, além de possuir menor consumo de energia médio por nó, ele possui aceitáveis taxas de entrega de dados em relação aos outros.

De posse dos 2 cenários selecionados como melhores para a aplicação em pontes com e sem uso de *superframe*, resta emparelhá-los e observar seus resultados de forma comparativa. Aqui, além da habitual comparação usada (consumo e taxa de entrega), inseriu-se um novo parâmetro de avaliação da rede, a taxa de transferência ou *throughput*.

O *throughput* (equação 5.4) determina a quantidade de dados que são transmitidos de uma fonte para um nó de destino por unidade de tempo (bits por segundo). Isto garante que o valor

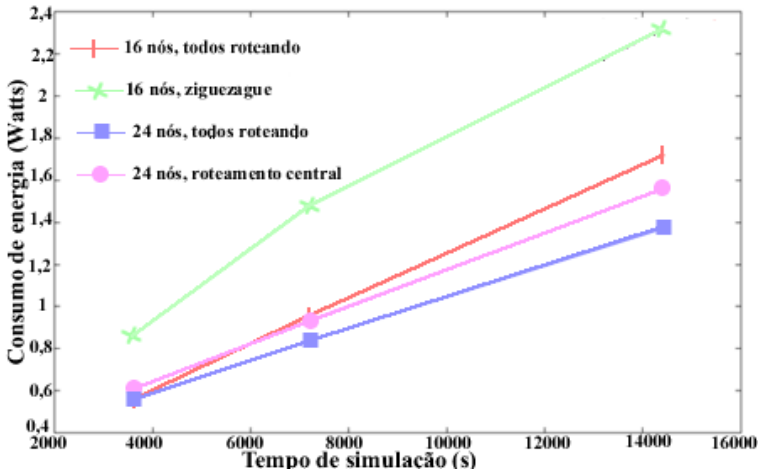

Figura 5.7: Consumo de energia x Tempo de simulação em redes sem beacons ativos

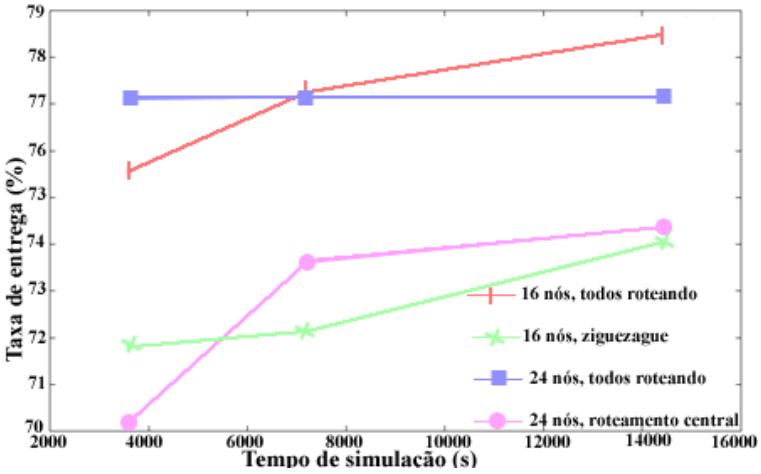

Figura 5.8: Proporção de entrega de pacotes x Tempo de simulação em redes sem beacons ativos

calculado represente a conexão entre cada nó fonte e o coordenador da rede. O calculo foi feito com base na média do total de bits recebidos com sucesso no destino em relação ao tempo de simulação para cada nó.

$$Vazao_{no} = \frac{Total\_de\_bits\_recebidos\_com\_sucesso\_no\_destino}{tempo\_de\_simulacao}$$

$$Vazao_{rede} = \sum \frac{Vazao_{no}\_para\_todos\_os\_nos}{total\_de\_nos} \tag{5.4}$$

Os dados obtidos estão nas figuras 5.9, 5.10 e 5.11.

Enquanto o cenário 1 mantém uma taxa de transmissão quase que invariável de 50 bits por segundo, o cenário 7 inicia as transmissões a uma taxa alta de aproximadamente 32 bits por segundo e finaliza a uma taxa média de 2,20 bits por segundo. Esta diferença na velocidade

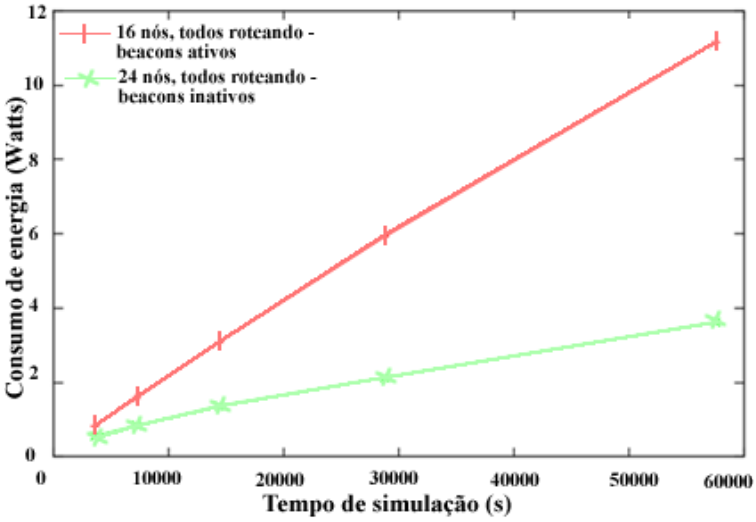

Figura 5.9: Consumo de energia x Tempo de simulação

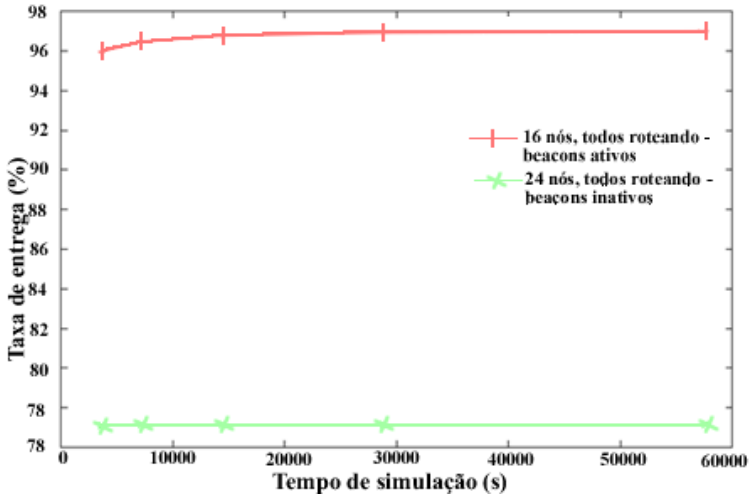

Figura 5.10: Proporção de entrega de pacotes x Tempo de simulação

está relacionada primeiramente à densidade dos nós na área de monitoramento e, em segundo, à consequente disputa do canal para transmissão. Sendo assim, a vazão da rede no segundo caso fica prejudicada pela espera dos nós pela liberação do canal para transmissão. Porém, justamente por essa menor taxa de transmissão, o cenário 7 ganha em economia de energia comparado ao cenário 1, já que os nós gastarão menos energia nos processos de recepção e transmissão de dados.

Ao longo deste trabalho foi frisada a importância do consumo de energia em detrimento da qualidade de serviço e a proposta desta simulação é escolher o cenário que permita um maior tempo de vida útil à rede de sensores em pontes. Como monitoramento em pontes não é uma aplicação de risco que necessite analisar uma grande massa de dados em um curto intervalo de tempo, o principal critério a ser considerado aqui será o consumo de energia. Sendo assim,

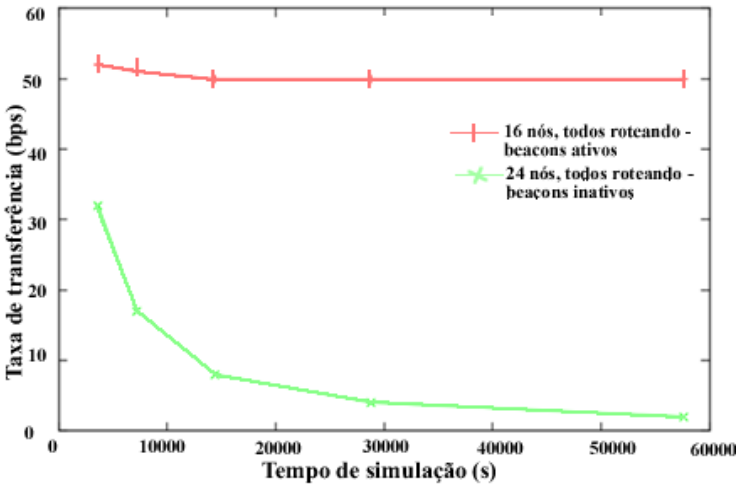

Figura 5.11: Vazão na rede x Tempo de simulação

toma-se o cenário 7 como o mais indicado.

Vale lembrar que dinamismo é um atributo imprescindível em RSSF. O uso de uma das duas últimas topologias não exclui a possibilidade do uso da outra em alguns períodos de tempos de maior risco para as estruturas da ponte, como em períodos de fortes chuvas.

Isto é, pode-se preparar a rede para trabalhar por longos períodos de baixa taxa de transferência de dados e consumo nos dias normais (cenário 7) e aumentar a vazão na coleta de dados sensoriados para os dias nos quais as pontes estejam submetidas a fortes influências externas (cenário 1).

Toda essa versatilidade possibilita várias soluções, sendo as sugeridas por este trabalho apenas algumas delas.

6 Conclusão

Neste trabalho foram vistos os conceitos primordiais para o entendimento de redes de sensores sem fio. A apresentação da problemática no uso consciente dos recursos de cada dispositivo, em conjunto com as dificuldades de concepção de um projeto, foram pontos abordados com o intuito de salientar a necessidade da quebra de padrões de qualidade de serviço referentes às redes estruturadas e *ad-hoc*.

Redes de sensores são abrangentes no nível de aplicação, restritas no nível de hardware e requerem alto grau de dinamismo e auto-gerenciamento de seus escassos recursos. Assim, tendo em vista a maximização do uso destes recursos no quesito tempo, outros fatores acabam por serem prejudicados, como velocidade de transmissão dos dados, velocidade de trocas de mensagens, qualidade do sinal e garantia de entrega. Ou seja, os parâmetros comuns de qualidade devem ser ponderados de acordo com as reais necessidades da aplicação alvo.

Logo, devido a este alto nível de detalhamento dos requisitos e características da aplicação, a etapa de projeção é a mais trabalhosa e a que demanda por mais tempo. Ferramentas de simulação se apresentam como imprescindíveis nesta etapa, mesmo que o nível de complexidade com o qual se lida nestes ambientes simulados seja muito inferior ao de uma aplicação real.

Observou-se também um leque vasto de ferramentas de simulação em atuação na comunidade científica e, ainda assim, existem necessidades ainda não supridas que motivam pesquisas e projetos em áreas que vão se apresentando à medida que redes de sensores expandem seus horizontes de atuação. São exemplos de pesquisas motivadas por determinada aplicação a melhoria do canal de comunicação subaquático ou intracorpóreo, extração de energia do meio monitorado e durabilidade dos dispositivos em ambientes inóspitos.

A fim de explicitar estas dificuldades, todo o processo inicial de projeção de uma rede com um propósito específico foi descrito neste trabalho. O intuito aqui foi projetar uma rede de sensores para monitoramento de pontes. O processo transitou desde questões sobre o que é RSSF, como surgiu, de que forma ela se comporta ou lida com restrições no processo de

comunicação. Levantou-se também ferramentas de simulação para auxiliar na projeção e, após escolhida a ferramenta, mapeou-se com um considerável nível de detalhamento uma rede real.

A partir de todos esses passos, observou-se como processos que não foram modelados especificamente para tratar essas restrições em redes de sensores repercutem negativamente na avaliação final de desempenho e vida útil do projeto. Assim, conclui-se sobre a necessidade de se tratar não só as camadas inferiores, mas também as camadas superiores da pilha de protocolos.

Como sugestão para trabalhos futuros, é proposto o mapeamento deste ou outro cenário de aplicação com enfoque nas camadas superiores, como a de transporte e/ou roteamento. Sugere-se ainda que o enfoque seja dado ao padrão aqui usado, IEEE 802.15.4. Isto, principalmente, por ele ser suportado por alguns dos principais simuladores atualmente usados na comunidade e por sua expressividade em redes de sensores industriais, tema também não abordado neste trabalho.

Recomenda-se o uso de específico do simulador ns2 para este propósito, por ser um simulador de fácil aprendizado e por possibilitar o acesso ao código base, proporcionando maior maleabilidade e aquisição de considerável conhecimento da arquitetura. Além disso, após este conhecimento consolidado, pode-se contribuir com a comunidade com o incremento de mais bibliotecas ou adaptação de protocolos antigos que objetivem projeções mais otimizadas.

Referências Bibliográficas

AKYILDIZ, I. F. et al. Wireless sensor networks: a survey. *IEEE Computer, vol. 38, no. 4, pág. 393-422*, 2002.

BARBOSA, T. M. G. de A. et al. Arquitetura de software para redes de sensores sem fios: a proeminência do middleware. *SEMISH - Anais do Seminário Integrado de Software e Hardware*, p. 1616–1630, 2005.

CROSSBOW TECHNOLOGY, INC. *Crossbow Technology Releases Industry's First End-to-End, Low-Power, Wireless Sensor Network Solution for Security, Monitoring, and Tracking Applications: Highly Reconfigurable XMesh protocol stack and MOTE-VIEWTM software enables TrueMesh Sensor Networks with 802.15.4 Support*. 2005. Disponível em: <http://www.xbow.com/pdf/XMesh_MOTEVIEW_PR.pdf>. Acessado em: 01/07/2010.

CUI, J.-H. et al. The challenges of building scalable mobile underwater wireless sensor networks for aquatic applications. *Network, IEEE May-June 2006 volume: 20*, p. 12–18, 2006.

DARPA AND USC/ISI AND XEROX PARC AND LBNL AND UNIVERSIDADE DE BERKELEY. *The Network Simulator - ns-2*. [S.l.], 1995. Disponível em: <http://www.isi.edu/nsnam/ns/>. Acessado em: 21/08/2009.

FEHLBERG, F. W. Multis: um servidor de contexto voltado à computação pervasiva. 2007. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/10745/000600791.pdf?sequence=1>. Acessado em: 09/07/2009.

GLOMOSIM. *GloMoSim - Global Mobile Information Systems Simulation Library*. [S.l.], 2000. Disponível em: <http://pcl.cs.ucla.edu/projects/glomosim/>. Acessado em: 07/09/2009.

GRAND Challenges in Computing. 2010. Disponível em: <http://www.bcs.org/server.php?show=nav.5821>. Acessado em: 30/07/2009.

GUTIÉRREZ, J. A.; CALAWAY, E. H.; BARETT, R. L. *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4*. [S.l.]: Standards Information Network, IEEE Press, 2007. 200 p.

IEEE COMPUTER SOCIETY. *IEEE Standard 802.15.4-2006, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. [S.l.], Setembro 2006.

LEITHARDT., V. R. Q. 2008. Disponível em: <https://saloon.inf.ufrgs.br/twiki-data/Disciplinas/CMP157/TF08Valderi/Artigo_Produzido.pdf>. Acessado em: 26/03/2009.

MICA2. *MICA2, Wireless Measurement System*. 2009. Disponível em: <http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf>. Acessado em: 10/02/2010.

OMNET++ Community. 2010. Disponível em: <http://www.omnetpp.org/>. Acessado em: 07/07/2010.

PARSEC - Parallel Simulation Environment for Complex Systems. [S.l.]. Disponível em: <http://pcl.cs.ucla.edu/projects/parsec/>. Acessado em: 07/09/2009.

PEREIRA, M. R.; AMORIM, C. L. de; CASTRO, M. C. S. de. Tutorial sobre redes de sensores. *Caderno do IME, vol.14*, 2003.

RAMACHANDRAN, I. *IEEE 802.15.4 MAC implementation in NS-2*. 2006. Disponível em: <http://www.ee.washington.edu/research/funlab/802_15_4/>. Acessado em: 25/04/2010.

SHOX. *Shox, a scalable ad hoc network simulator*. [S.l.], 2009. Disponível em: <http://shox.sourceforge.net/>. Acessado em: 03/09/2009.

SIMULATION of IEEE 802.15.4/ZigBee with Network Simulator-2 (ns-2). 2009. Disponível em: <http://www.ifn.et.tu-dresden.de/ marandin/ZigBee/ZigBeeSimulation.html>. Acessado em: 21/08/2009.

SPERB, R. M. et al. Classificação de massas d'Água: um enfoque difuso. *ENEGEP*, 1999.

TACOMA. *History of the Tacoma Narrows Bridge*. University of Washington Libraries Special Collections, 2009. Disponível em: <http://www.lib.washington.edu/specialcoll/exhibits/tnb/page5.html>. Acessado em: 29/03/2010.

THE MARYLAND HYBRID NETWORKS CENTER - HYNET. *Atemu - Sensor Network Emulator / Simulator / Debugger*. [S.l.], 2004. Disponível em: <http://www.hynet.umd.edu/research/atemu/>. Acessado em: 23/08/2009.

TINYOS. [S.l.], 2009. Disponível em: <http://www.tinyos.net>. Acessado em 13/11/2009.

TOSSIM - TinyOS Documentation Wiki. [S.l.], 2009. Disponível em: <http://docs.tinyos.net/index.php/TOSSIM>. Acessado em: 01/10/2009.

WEISER, M. *The Computer for the 21st Century*. 1991. Disponível em: <http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html>. Acessado em: 15/01/2010.

XBEE ZNet 2.5/XBee-PRO ZNet 2.5 OEM RF Modules. [S.l.], 2008.

ZIGBEE ALLIANCE. *ZigBee Specification*. [S.l.], 2009. Disponível em: <http://www.zigbee.org/>. Acessado em: 3/12/2009.

Apêndice

Script para ns2 usado para obter os resultados deste trabalho. O mesmo foi construído com base no exemplo criado por Jianliang Zheng disponível gratuitamente online (RAMACHANDRAN, 2006).

```
# ======================================================================
# Definindo parametros iniciais
# ======================================================================
set val(chan)         Channel/WirelessChannel    ;# tipo de canal
set val(prop)         Propagation/TwoRayGround   ;# modelo de prop. de radio
set val(netif)        Phy/WirelessPhy/802_15_4   ;# tipo de interface de rede
set val(mac)          Mac/802_15_4               ;# tipo MAC ZigBee
set val(ifq)          Queue/DropTail/PriQueue    ;# tipo de fila
set val(ll)           LL                         ;# link layer type
set val(ant)          Antenna/OmniAntenna        ;# modelo de antena
set val(ifqlen)       100                        ;# maximo de pacotes na fila
set val(nn)           25                         ;# p1=17 e p2=25
set val(rp)           AODV                       ;# protocolo de roteamento
set val(x)            4                          ;# largura
set val(y)            20                         ;# comprimento
set val(allFFD)       1                          ;# todos sao FFD(allffd=1),
                                                  # nem todos sao FFD(allffd=0)
set val(nam)          saida_nam.nam              ;# arquivo de saida do nam

# Tempo decorrido ate que a simulacao finalize
#(3600s=1h)(7200s=2h)(14400s=4h)(28800 = 8h)(57600 = 16h)
set stopTime          100

#=============================================
#Metodo de definicao de fluxo de dados no meio
#=============================================
proc cbrtraffic { 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)
```

```
33      set cbr_($src) [new Application/Traffic/CBR]
34      eval \$cbr_($src) set packetSize_ 10  ;#tamanho dos pacotes em bytes
35      eval \$cbr_($src) set interval_ $interval
36      eval \$cbr_($src) set random_ 0
37      eval \$cbr_($src) attach-agent \$udp_($src)
38      eval $ns_ connect \$udp_($src) \$null_($dst)
39      $ns_ at $starttime "$cbr_($src) start"
40  }
41
42  #=================================
43  # Inicializar as variaveis globais
44  #=================================
45  set ns_        [new Simulator]
46  set tracefd     [open ./saida_tr.tr w]
47  $ns_ trace-all  $tracefd
48  set namtrace    [open ./$val(nam) w]
49  $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
50
51  #=====================================================
52  # Informa nam de que esse eh um arquivo trace para wpan
53  #(necessita de um cabecalho especial)
54  #=====================================================
55  $ns_ puts-nam-traceall {# nam4wpan #}
56  Mac/802_15_4 wpanCmd verbose on        ;#executa em modo detalhado
57  Mac/802_15_4 wpanNam namStatus on      ;#default = off - deve ser ativando
58                                          #antes de outros comandos 'wpanNam'
59  Mac/802_15_4 wpanNam ColFlashClr gold ;# default = gold (cor da colisao)
60
61
62  #=======================================================
63  #appTime define o momento em que a aplicacao sera iniciada
64  #=======================================================
65  for {set i 1} {$i< $val(nn)} {incr i} {
66    set appTime($i)     [expr 30+($i-1)]
67  }
68
69  #Limiar de potencia de deteccao de portadora (watt)
70  Phy/WirelessPhy set CSThresh_ 7.94328e-6;#Limiar de sensoriamento(watt)
71  Phy/WirelessPhy set RXThresh_ 7.94328e-6;#Limiar de potencia de recepcao(W)
72  Phy/WirelessPhy set CPThresh_ 10        ;#Limiar de captura (db)
73  Phy/WirelessPhy set freq_ 9.16e+8       ;#Frequencia (GHz)
74  Phy/WirelessPhy set L_ 1.0              ;#Fator de perda (-)
75  Phy/WirelessPhy set pt_ 0.001           ;#Potencia de transmissao (watt)
```

```
76
77 #============================
78 # Configuracao da topografia
79 #============================
80 set topo       [new Topography]
81 $topo load_flatgrid $val(x) $val(y)
82
83 #==========
84 #Create God
85 #==========
86 set god_ [create-god $val(nn)]
87
88 #=============================================================================
89 #Cria/Reserva um canal (faixa de frequencia) para comunicacao entre nos
90 #=============================================================================
91 set chan_1_ [new $val(chan)]
92
93 #=============================================================
94 # Configuracao para o no observador ou coordenador da PAN.
95 #=============================================================
96 $ns_ node-config -adhocRouting $val(rp) \
97           -llType $val(ll) \
98           -macType $val(mac) \
99           -ifqType $val(ifq) \
100          -ifqLen $val(ifqlen) \
101          -antType $val(ant) \
102          -propType $val(prop) \
103          -phyType $val(netif) \
104          -topoInstance $topo \
105          -agentTrace OFF \
106          -routerTrace OFF \
107          -macTrace ON \
108          -movementTrace OFF \
109          -channel $chan_1_
110          # Parametros de energia requeridos somente quando existe um
111          #modelo requerido
112               #-energyModel "EnergyModel" \
113               #-initialEnergy 1000 \
114               #-rxPower 35.28e-3 \
115               #-txPower 31.32e-3 \
116               #-idlePower 712e-6 \
117               #-sleepPower 144e-9 \
118
```

```
#=========================================
# Cria especificacao do no e atrela ao canal
#=========================================
for {set i 0} {$i < 1 } {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0       ;# disable random motion
}
#==================
# Cria os demais nos
#==================
$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 OFF \
            -routerTrace ON \
            -macTrace ON \
            -movementTrace OFF \
                 -energyModel "EnergyModel" \
                 -initialEnergy 1000 \
                 -rxPower 30e-3 \
                 -txPower 81e-3 \
                 -idlePower 3e-6 \
                 -sleepPower 45e-6 \
                 -channel $chan_1_


for {set i 1} {$i < $val(nn) } {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0      ;# disable random motion
}

#================================
#configurando a topologia da rede
#================================
if { $val(nn) == 17 } {
      set k 0
      set j $val(x)
```

```
162
163       for {set i 1} {$i < $val(nn)} {incr i} {
164        if {$k > $val(y) } {
165          set k 0
166        }
167        if {$j > $val(x) } {
168          set j 0
169          $node_($i) set X_ 0
170          $node_($i) set Y_ $k
171          $node_($i) set Z_ 0
172          set j $val(x)
173          set k [expr $k + 2.5]
174        } else {
175          $node_($i) set X_ 4
176          $node_($i) set Y_ $k
177          $node_($i) set Z_ 0
178          incr j
179        }
180       }
181   } else {
182       set j 0
183       for {set i 1} {$i < $val(nn)} {incr i} {
184        if {$j > $val(x)} {
185         set j 0
186        }
187
188        for {set k 0} {$k < $val(y)} {} {
189         $node_($i) set Z_ 0
190         $node_($i) set Y_ $k
191         $node_($i) set X_ $j
192         set k [expr $k + 2.5]
193         if {$k < $val(y)} {
194          incr i
195         }
196         if {$i > [expr $val(nn) -1]} {
197          set k [expr $val(y) + 1]
198         }
199        }
200        set j [expr $j + 2]
201       }
202 }
203 $node_(0) set X_ [expr $val(x)/2]
204 $node_(0) set Y_ $val(y)
```

```
205 $node_(0) set Z_ 0
206
207 #=====================================================
208 #Aqui defino o no 0 como o coordenador da PAN
209 #Lembrando que este no nao possui restricoes de energia.
210 #=====================================================
211 $ns_ at 0.0   "$node_(0) NodeLabel \"PAN Coor\""
212
213 #cmd usado para iniciar uma nova PAN com o correspondente no como 'PanCoor'
214 #$node sscs startPANCoord <txBeacon=1><beaconOrder=3><SuperframeOrder=3>
215 $ns_ at 0.0   "$node_(0) sscs startPANCoord 1 6 6"
216
217 #cmd usado para iniciar um roteador ou dispositivo final
218 #$node sscs startDevice <isFFD=1><assoPermit=1><txBeacon=0><beaconOrder=3><
    SuperframeOrder=3>
219 if { $val(allFFD) == 0 } {
220       if { $val(nn) == 25 } {
221         for {set i 1} {$i <= 8} {incr i} {
222          $ns_ at [expr 5+$i-1]   "$node_($i) sscs startDevice 0"
223         }
224
225         for {set i 9} {$i <= 16} {incr i} {
226          $ns_ at [expr 5+$i-1]   "$node_($i) sscs startDevice 1 1 0 6 2"
227         }
228
229         for {set i 17} {$i <= 24} {incr i} {
230          $ns_ at [expr 5+$i-1]   "$node_($i) sscs startDevice 0"
231         }
232       } else {
233         for {set i 1} {$i < [expr $val(nn)]} {incr i} {
234          $ns_ at [expr 5+$i-1]   "$node_($i) sscs startDevice 1 1 0 6 2"
235          incr i
236          $ns_ at [expr 5+$i-1]   "$node_($i) sscs startDevice 0"
237          incr i
238          $ns_ at [expr 5+$i-1]   "$node_($i) sscs startDevice 0"
239          incr i
240          $ns_ at [expr 5+$i-1]   "$node_($i) sscs startDevice 1 1 0 6 2"
241         }
242       }
243 } else {
244       if { $val(allFFD) == 1 } {
245         for {set i 1} {$i < [expr $val(nn)]} {incr i} {
246           $ns_ at [expr 5+$i-1]   "$node_($i) sscs startDevice 1 1 0 6 2"
```

```
247         }
248       }
249 }
250
251 #============================================
252 #Configuracao para a interface de animacao Nam
253 #============================================
254 Mac/802_15_4 wpanNam PlaybackRate 3ms
255
256 $ns_ at $appTime(1) "puts \"\nTransmitting data ...\n\""
257
258 #=========================================
259 # Configurar o fluxo de Ã¡trfico entre os nos
260 #=========================================
261 puts "\nTraffic: cbr"
262 #acks da camada MAC para os pacotes da camada superior(upper layer),
263 #por default vem ligado Mac/802_15_4 wpanCmd ack4data off
264 puts [format "Acknowledgement for data: %s" [Mac/802_15_4 wpanCmd ack4data
      ]]
265 $ns_ at $appTime(1) "Mac/802_15_4 wpanNam PlaybackRate 0.20ms"
266 $ns_ at [expr $appTime(1) + 0.5] "Mac/802_15_4 wpanNam PlaybackRate 1.5ms"
267
268 #Configura trafego de dados para os nos da borda da topologia da rede
269 if { $val(nn) == 25 } {
270       for {set i 1} {$i <= 8} {incr i} {
271          cbrtraffic $i 0 3 $appTime(1)
272       }
273       for {set i 17} {$i <= 24} {incr i} {
274          cbrtraffic $i 0 3 $appTime(1)
275       }
276 } else {
277       for {set i 1} {$i < $val(nn)} {incr i} {
278          cbrtraffic $i 0 3 $appTime(1)
279       }
280 }
281
282 $ns_ at $appTime(1) "$ns_ trace-annotate \"(at $appTime(1)) cbr traffic
      from node n to node 0\""
283
284 #Define cor de fluxo a fim de discernir quais sao os tipos de pacotes
285 Mac/802_15_4 wpanNam FlowClr -p AODV -c tomato
286 Mac/802_15_4 wpanNam FlowClr -p ARP -c green
287 Mac/802_15_4 wpanNam FlowClr -p MAC -s 0 -d -1 -c yellow
```

```
288 Mac/802_15_4 wpanNam FlowClr -p tcp -s 0 -d 1 -c blue
289 Mac/802_15_4 wpanNam FlowClr -p tcp -s 1 -d 0 -c blue
290 Mac/802_15_4 wpanNam FlowClr -p ack -s 1 -d 0 -c cyan4
291
292 #=============================
293 # Define o tamanho do no no nam
294 #=============================
295 for {set i 0} {$i < $val(nn)} {incr i} {
296      $ns_ initial_node_pos $node_($i) 1
297 }
298
299 #=======================================
300 # Diz aos nos quando a simulacao terminou
301 #=======================================
302 for {set i 0} {$i < $val(nn) } {incr i} {
303      $ns_ at $stopTime "$node_($i) reset";
304 }
305
306 $ns_ at $stopTime "stop"
307 $ns_ at $stopTime "puts \"NS EXITING...\n\""
308 $ns_ at $stopTime "$ns_ halt"
309
310 proc stop {} {
311    global ns_ tracefd appTime1 val env
312    $ns_ flush-trace
313    close $tracefd
314
315    foreach index [array names env] {
316        puts "$index: $env($index)"
317        if { ("$index" == "DISPLAY") && ("$env($index)" != "") } {
318      set hasDISPLAY 1
319        }
320    }
321    if { ("$val(nam)" == "saida_nam.nam") && ("$hasDISPLAY" == "1") } {
322        exec nam saida_nam.nam &
323    }
324 }
325 puts "\nStarting Simulation..."
326 $ns_ run
```