COPPE/UFRJ

AJUSTE ADAPTATIVO DO SINAL DE TRANSMISSÃO DE MENSAGENS DE DESCOBERTA PARA CONSUMO EFICIENTE DE ENERGIA EM REDES TOLERANTES A ATRASOS E INTERRUPÇÕES.

Tiago Souza Azevedo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Luís Felipe Magalhães de Moraes

Rio de Janeiro
Junho de 2010

AJUSTE ADAPTATIVO DO SINAL DE TRANSMISSÃO DE MENSAGENS DE DESCOBERTA PARA CONSUMO EFICIENTE DE ENERGIA EM REDES TOLERANTES A ATRASOS E INTERRUPÇÕES.

Tiago Souza Azevedo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Luís Felipe Magalhães de Moraes, Ph. D.

Prof. Claudio Luis de Amorim, Ph. D.

Prof. Márcio Portes de Albuquerque, Ph. D.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2010

Azevedo, Tiago Souza

Ajuste Adaptativo do Sinal de Transmissão de Mensagens de Descoberta para Consumo Eficiente de Energia em Redes Tolerantes a Atrasos e Interrupções./Tiago Souza Azevedo. – Rio de Janeiro: UFRJ/COPPE, 2010.

XVIII, 134 p.: il.; 29, 7cm.

Orientador: Luís Felipe Magalhães de Moraes

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 67 – 77.

1. Redes Sem Fio. 2. Redes Ad Hoc. 3. Redes DTNs. 4. Consumo de Energia. I. Moraes, Luís Felipe Magalhães de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico este trabalho aos meus familiares, amigos e professores, que sempre me apoiaram em meus projetos.

Agradecimentos

Gostaria de agradecer aos meus pais, Carlos Roberto e Solange, a minha avó Yvone e aos meus irmãos Rafael e Natália, por toda dedicação, paciência e apoio. A minha namorada Jaqueline que com todo amor me ajudou a superar todos os percalços e desafios.

Gostaria de agradecer também ao Professor Luís Felipe pela oportunidade, pelo conhecimento e sabedoria transmitidos a nós alunos. Agradeço ainda aos meus amigos Airon Fontaneles, Bruno Astuto, Carlos Alberto Campos, Cláudia Silva, Danielle Lopes, Diogo Moreno, Eduardo Hargreaves, Fernando Veríssimo, Gustavo Dias, Jorge Peixoto, Júlio Nóbrega, Paulo Ditarso, Rafael Bezerra e Rafael Fernandes por toda ajuda na elaboração desta dissertação e principalmente, pela convívio e amizade.

Agradeço ainda a COPPE, ao Programa de Engenharia de Sistemas e Computação (PESC) e seus funcionários pelo suporte operacional e à FAPERJ pelo financiamento deste projeto.

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

AJUSTE ADAPTATIVO DO SINAL DE TRANSMISSÃO DE MENSAGENS DE DESCOBERTA PARA CONSUMO EFICIENTE DE ENERGIA EM REDES TOLERANTES A ATRASOS E INTERRUPÇÕES.

Tiago Souza Azevedo

Junho/2010

Orientador: Luís Felipe Magalhães de Moraes

Programa: Engenharia de Sistemas e Computação

Um dos principais desafios encontrados em Redes Tolerantes a Atrasos e Interrupções é o consumo eficiente de energia. Trabalhos indicam que em alguns cenários, este consumo ocorre principalmente no processo de descoberta de dispositivos vizinhos. Diversas são as estratégias para reduzir o consumo e garantir um aumento no tempo de vida operacional da rede, contudo, estas abordagens trazem consigo uma redução do número de contatos entre esses dispositivos, o que implica em um acréscimo no retardo e diminuição da capacidade da rede. Nesse sentido, este trabalho tem por objetivo identificar a relação custo/benefício entre número de contatos e o consumo de energia, durante o processo de descoberta de nós vizinhos. Para isso uma nova métrica, a probabilidade de perder oportunidades de contato é definida e um modelo analítico é proposto. Posteriormente este modelo é avaliado e validado a partir de resultados simulados. Por fim, a partir das análises deste modelo, um algoritmo para ajuste adaptativo da potência do sinal de transmissão é proposto e avaliado. Esse algoritmo, em cenários cujo intervalo de contato é distribuído exponencialmente, é capaz de reduzir o consumo a até metade do consumo total mantendo-se baixos índices de perda de oportunidades de contato.

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

ADAPTIVE ADJUST OF TRANSMISSION SIGNAL OF THE DISCOVERY MESSAGES FOR EFFICIENT ENERGY CONSUMPTION IN DELAY AND DISRUPTION TOLERANT NETWORKS.

Tiago Souza Azevedo

June/2010

Advisor: Luís Felipe Magalhães de Moraes

Department: Systems Engineering and Computer Science

One of the main challenges found in Delay and Disruption Tolerant Networks is the efficient consumption of energy. Many studies indicate that this consumption occurs mainly in the neighbour discovery process. There are many approaches to reduce the energy consumption and increase the lifetime of the network, however, all the approaches induce a decrease in number of contacts. In this sense, this work propose to identify the tradeoff between the amount of contacts and energy consumption during neighbour discovery process. Thus, a new metric, the probability of lose opportunities of contact is defined and a new analytic model is proposed. Later, this model is evaluated and validated through simulated results. Finally, from the analisys of this model, an adaptive algorithm was proposed and validated. This algorithm, in scenarios where the contact interval is exponentially distributed, make it possible to reduce the consumption by up to half the total consumption, while maintaining low rates of loss of opportunities of contact.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ATIM	Ad-Hoc Traffic Indication Maps, p. 16
BER	Bit Error Rate, p. 27
CAPM	Context Aware Power Management, p. 16
ChaNETS	Challenged Networks, p. 3
DSL	Digital Subscriber Line, p. 6
DTNRG	Delay and Disruption Tolerant Network Research Group, p. 3
DTN	Delay and Disruption Tolerant Networks, p. 2
GPS	Global Positioning System, p. 9
IEEE	Institute of Electrical and Electronics Engineers, p. 17
IPNSIG	Inter-Planetary Internet Special Interest Group, p. 3
IPN	Interplanetary Network, p. 3
IP	Internet Protocol, p. 2
IRTF	Internet Research Task Force, p. 3
MANET	Mobile Ad Hoc Networks, p. 2
MPAs	Mobile Access Points, p. 7
MULEs	Mobile Ubiquitous LAN Extensions, p. 9
PSM	Power Saving Mode, p. 16
SNR	Signal to Noise Ratio, p. 27
SWIM	Shared Wireless Infostation Model, p. 6
TCP	Transmission Control Protocol, p. 2

TSF	Time Synchronization Function, p. 16
VANET	Vehicular AdHoc NETworks, p. 2

Lista de Símbolos

$A_{n_i,ef}(p)$	Área de alcance efetivo do dispositivo n_i, p. 28
$A_{n_i,max}$	Área de alcance máximo do dispositivo n_i, p. 28
$C(p)$	Potência consumida na transmissão de uma mensagem de descoberta, p. 21
$C_{n,total}(t)$	O consumo total do dispositivo n até o instante t, p. 22
E	Área em estudo, p. 30
G_{rx}	Ganhos da antena de recepção em dBi, p. 27
G_{tx}	Ganhos da antena de transmissão em dBi, p. 27
N	Conjunto de dispositivos em estudo, p. 30
$Op_t(n_i, n_j)$	Variável aleatória indicadora que representa o evento da oportunidade de contato do nó n_i com outro nó qualquer n_j no instante t, p. 30
P_{rx}	Potências de recepção medida em Watts., p. 26
$P_{tx,max}$	Máxima potência de transmissão do rádio medida em W, p. 28
P_{tx}	Potências de transmissão medida em Watts., p. 26
$Pos(n_i, t)$	Posição do dispositivo n_i no instante de tempo t, p. 30
R_{ef}	Raio da área de alcance efetivo, p. 28
R_{max}	Raio da área de alcance máximo, p. 28
Δ	Variável aleatória que representa a duração de um intervalo de contato, p. 30
Γ	Variável aleatória indicadora do evento de perder uma oportunidade de contato, p. 32

Ω	Variável aleatória que representa o número de *probes* transmitidos em um intervalo de contato, p. 32
Θ	Variável aleatória que representa o intervalo de tempo entre contatos consecutivos, p. 30
Ξ	Nível de ruído ambiente medido em Watts(W), p. 27
β	Limiar de audição ou *sensitividade*, p. 28
λ	Comprimento da onda em metros, p. 27
ρ	Densidade de nós por área, p. 30
τ_i	Instante de início da oportunidade de contato i, p. 30
υ_i	Instante de término da oportunidade de contato i, p. 30
p	Potência efetiva de transmissão do rádio medida em W, p. 28
A	Diretividade da antena (*antenna aperture*), p. 27
L	Perda de sinal ou atenuação do sinal (*Path loss*), p. 26
S	Quantidade de potência por unidade de área, p. 26
d	Distância entre os dispositivos, p. 26
snr(d)	Relação Sinal-Ruído, p. 27

Listagem de Arquivos

Capítulo 1

Introdução

Nas últimas décadas temos presenciado um crescimento acelerado da Internet. Esse crescimento permitiu a consolidação desta rede como a infraestrutura principal de disseminação de conteúdos em escala mundial, sendo capaz de prover meios de informação, colaboração e interação entre indivíduos e seus dispositivos computacionais.

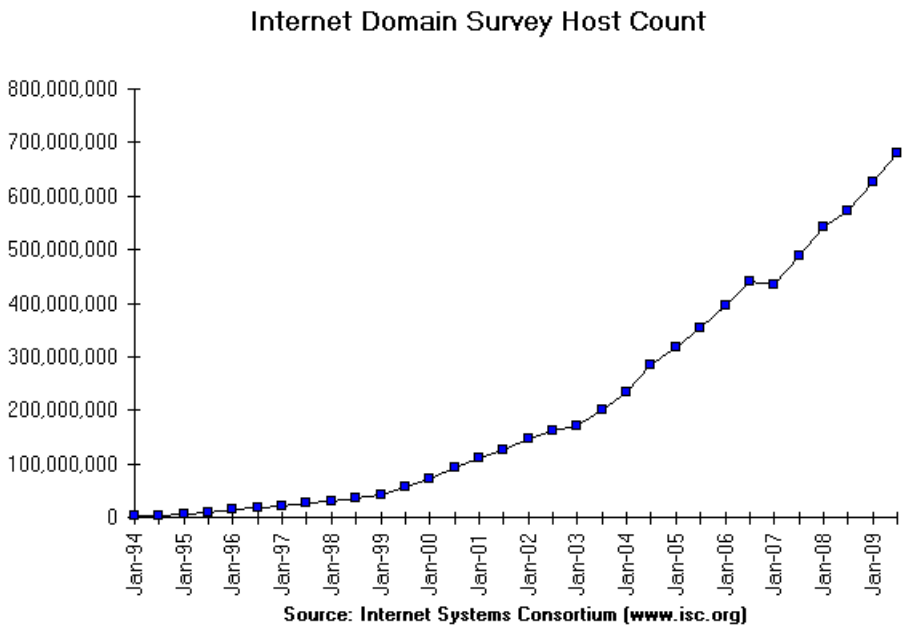

Figura 1.1: *Registro do número de hosts na Internet entre Janeiro de 2004 e Janeiro de 2009, feita pela Internet System Consortium [1].*

Este sucesso, em parte, é decorrente da Arquitetura[10] na qual foi desenvolvida, concebida para operar de forma independente da tecnologia de subrede permitindo assim maior flexibilidade, eficiência, robustez e capacidade de suportar diversas aplicações em diferentes cenários.

Contudo, é consenso entre os pesquisadores, que as premissas adotadas principal-

mente no desenvolvimento dos protocolos da camada de transporte não são capazes de suportar novos cenários de comunicação como: as Redes Móveis Ad Hoc [11]; as Redes de Sensores [12]; as Redes Subaquáticas [13]; as Redes Veiculares [14], dentre outras.

Isto ocorre, pois os principais protocolos Internet, TCP e IP, foram concebidos sob as suposições de conectividade fim-a-fim, atrasos de comunicação relativamente pequenos, baixas taxas de erros e mecanismos de retransmissão efetivos. Hipóteses que não ocorrem nos novos cenários, gerando portanto, o grande desafio de compatibilizar as tecnologias existentes com as demandas atuais.

Neste contexto encontramos as Redes Tolerantes a Atrasos e Interrupções(*Delay and Disruption Tolerant Networks - DTN's*). A Arquitetura DTN [15], tem por objetivo oferecer interoperabilidade entre redes. Esta pode ser vista como um meio de estender as capacidades da Arquitetura Internet(TCP/IP), permitindo comunicação mesmo quando a conectividade entre dispositivos for intermitente ou nula.

1.1 Redes Tolerantes a Atrasos e Interrupções

1.1.1 Breve Histórico

O conceito de Redes Tolerantes a Atrasos e Interrupções (DTN) surgiu na década de 90. Embora seja considerado mais abrangente atualmente, a ideia partiu da necessidade de implementar uma Rede Interplanetária (IPN)[16] baseada na suíte de protocolos TCP/IP, sob a supervisão do Grupo de Trabalho *IPNSIG*. Ao operar no espaço, esta rede deveria ser capaz de suportar os altos retardos e conexões intermitentes inerentes às transmissões por distâncias espaciais, necessitando portanto, de uma nova arquitetura e novos protocolos capazes de suportar tais condições [16].

Posteriormente em 2003, este conceito foi generalizado no artigo [17]. Nesse trabalho, os autores identificaram que as suposições assumidas em redes convencionais não caracterizavam bem alguns cenários como: comunicações sem fio, comunicações entre dispositivos móveis e com restrições de energia, comunicações rurais, subaquáticas e também comunicações interplanetárias. Essas redes de comunicação possuem em sua essência problemas para manter uma comunicação fim-a-fim com baixa latência e pequena perda de pacotes e por isso, foram denominadas Redes Desafiadoras (ChaNETS).

Em 2004, foi criado pelo IRTF, um grupo de trabalhos para estudo e desenvolvimento de protocolos alternativos de comunicação, o DTNRG [18]. Logo após, em 2006, a Arquitetura DTN começou a ser definida em um *Internet Draft*, que descrevia as soluções encontradas para resolver as incompatibilidades entre as arquiteturas existentes. Tais incompatibilidades podem ser visualizadas na Tabela 1.1.

Atualmente, diversas pesquisas e projetos vem sendo realizados. Tais trabalhos serão explorados na Seção 1.2.

1.1.2 Conceitos e Características

A Arquitetura DTN é composta de *Redes Regionais*. Uma Rede Regional corresponde a um agrupamento de dispositivos com características semelhantes. Em cada região assume-se que existe um caminho fim-a-fim entre os nós. Caso seja necessária a comunicação entre regiões, esta é conduzida por um dispositivo especial denominado *gateway DTN*, responsável por resolver problemas de compatibilidade.

Para permitir essa agregação de Redes Regionais é definida na Arquitetura DTN uma sobrecamada, situada abaixo da camada de aplicação, denominada camada de agregação (*Bundle Layer*), que pode ser visualizada na Figura 1.2. Nessa camada são definidas as funcionalidades que permitem aos dispositivos operar de forma independente do tipo de cenário, resolvendo os problemas inerentes às ChaNETs como

Premissas	Internet	Redes Desafiadoras
Atraso	**Baixo -** Por causa da baixa taxa de erros e da existência de caminhos com pouca variação com relação à transmissão, o atraso é baixo.	**Alto -** Devido aos problemas relacionados às taxas de erros e a assimetria de caminhos, o atraso pode ser elevado.
Conectividade	**Fim-a-fim -** entre quaisquer pares de dispositivos que se comunicam na rede, existe pelo menos um caminho bi-direcional fim-a-fim.	**Intermitente -** a conectividade entre dispositivos pode sofrer de desconexões devido a fatores como falta de energia, problemas de rádio, alta mobilidade.
Taxas de Transmissão	**Simétrica -** A taxa de transmissão é assumida como aproximadamente simétrica, ou seja, as taxas de recepção e de transmissão possuem a mesma ordem de grandeza.	**Assimétrica -** Os enlaces são assimétricos, podendo muitas vezes ser unidirecionais.
Taxa de Perda de Pacotes	**Baixas -** Embora os pacotes possam ser perdidos devido a congestionamentos ou falhas de enlaces as taxas de perdas são baixas.	**Altas -** Devido à natureza inerente do meio sem-fio presente nestas redes, altas taxas de erro podem ocorrer.

Tabela 1.1: *Comparação entre as premissas utilizadas na arquitetura Internet(TCP/IP) e em Redes Desafiadoras.*

ausência parcial ou total de comunicação fim-a-fim, alta latência e alta taxa de perda de pacotes.

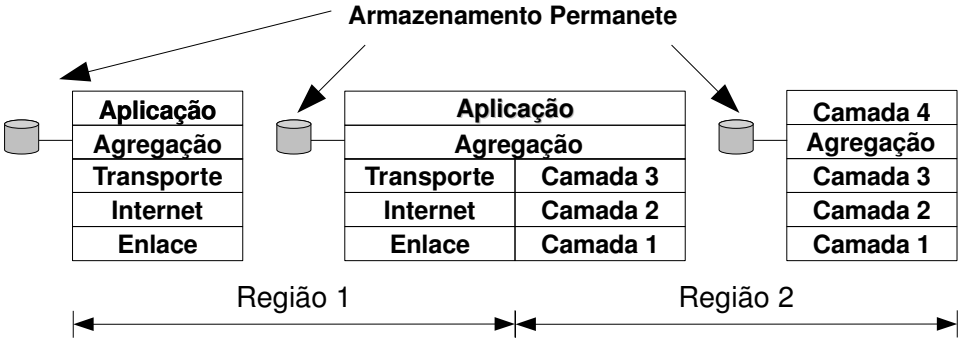

Figura 1.2: *Exemplo da arquitetura DTN. A camada de agregação é responsável por garantir a interoperabilidade dos dispositivos em diferentes regiões.*

Além das caracteristicas anteriormente citadas, a Arquitetura DTN ainda especifica:

- **Armazenamento Permanente:** Por ser caracterizada por longos atrasos e interrupções, a arquitetura DTN prevê o armazenamento permanente de informações, em disco rígido, durante seu trânsito pela rede;

- **Gateways DTN:** Conjunto de nós responsáveis pela interoperabilidade entre redes. Geralmente estão localizados em pontos de interconexão e são capazes de fazer a conversão entre protocolos distintos;

- **Resolução de Nomes:** Para rotear as mensagens entre diversas redes, na especificação da arquitetura DTN foi definido um novo espaço de endereçamento baseado em tuplas de nomes. Uma tupla consiste de duas partes, a primeira é globalmente única e hierarquicamente estruturada responsável por identificar uma região. A segunda parte identifica o dispositivo interno à região;

- **Encaminhamento por Custódia:** Uma das maneiras encontradas para prover confiabilidade fim-a-fim através de múltiplos saltos. Os nós delegam aos dispositivos encaminhadores, *relays*, a responsabilidade de entrega ao destino. Isto é feito através da troca de mensagens de reconhecimento (*acks*) a cada salto e armazenamento persistente.

- **Classes de serviço:** Também foi previsto na especificação classes de serviços permitindo diferenciar alocação de *buffers*, capacidade de enlaces, tempo de processamento etc.

Diferentemente da Arquitetura Internet, em uma DTN, nem sempre existem caminhos entre dois dispositivos. Estes caminhos podem surgir com o tempo, a medida que os dispositivos se movimentam e permanecem próximos o suficiente para que seus rádios se comuniquem, como também podem ser desfeitos, quando os nós se afastam e seus rádios perdem a comunicação. O estabelecimento de enlaces e caminhos entre dispositivos é dependente da disposição espacial dos nós, de sua mobilidade, de seu nível de energia etc.

Quando os dispositivos estão em condições de estabelecer uma comunicação, é dito que estes possuem uma *oportunidade* de comunicação. Em DTN, esta condição favorável é denominada *contato*. Na Figura 1.3, são apresentadas três imagens tomadas em instantes distintos da evolução de uma DTN. Os dispositivos são representados por círculos preenchidos, os círculos tracejados representam o alcance de seus respectivos rádios e as setas indicam a direção e intensidade da velocidade dos nós.

No instante de tempo t_0, os dispositivos **a** e **b** estão próximos o suficiente e estabelecem contato. Esse contato é mantido durante o intervalo $[t_0, t_1]$. Logo após o dispositivo **b** se afasta de **a** e se aproxima de **c**, estabelecendo um novo enlace. No instante t_2, os dispositivos **b** e **c** afastam-se e o contato é perdido, contudo, o nó **c** aproxima-se de **d** e um novo contato é estabelecido.

O intervalo de tempo em que os nós permanecem em contato, denominado *tempo de contato* ou *intervalo de contato*, é altamente dependente da mobilidade e da

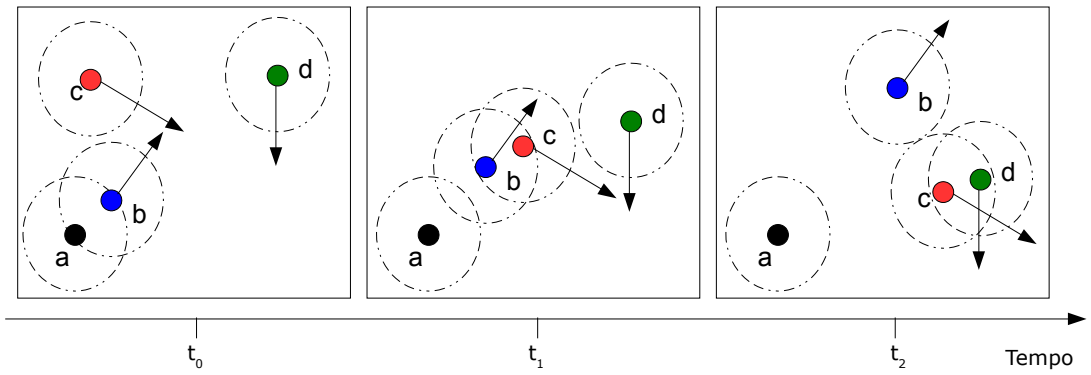

Figura 1.3: *Dinâmica de uma DTN, devido a mobilidade dos nós, a cada instante de tempo contatos antigos são desfeitos e novos contatos são estabelecidos.*

velocidade com a qual se movimentam e influencia de forma direta a quantidade de informação que pode ser trocada entre os dispositivos.

O intervalo de tempo medido desde o fim de um contato até o início do próximo é denominado *tempo entre contatos* ou *intervalo entre contatos*. Este intervalo também é dependente da mobilidade e possui grande influência no atraso das entregas de mensagens. É importante observar que o tempo entre contatos pode ser caracterizado para um par específico de nós, ou em relação a um dispositivo para todos os demais.

O conjunto de instantes em que os contatos ocorrem é denominado *agendamento de contatos*. Este agendamento pode ocorrer de diferentes formas e a Arquitetura DTN classifica-os em: persistente, sob demanda, programado e oportunista.

Os contatos persistentes são aqueles que estão sempre disponíveis, como é o caso de uma enlace DSL contratado para conexão com a Internet. Os contatos sob demanda, necessitam de algum evento para que ocorram, como por exemplo a requisição do usuário em uma conexão discada.

Os contatos programados ocorrem de forma agendada, com horários pré-estabelecidos, como por exemplo em Redes Interplanetárias, em que o movimento dos astros no espaço é determinado. Já os contatos oportunistas ocorrem de forma aleatória, como por exemplo os contatos entre dispositivos móveis (Palms e telefones celulares) em uma conferência.

1.1.3 Aplicações da Arquitetura DTN

São vários os cenários e projetos em que os conceitos da Arquitetura DTN foram aplicados. No modelo SWIM, proposto em [19], os autores buscavam a obtenção de dados biológicos de colônias de baleias em alto mar. Com o objetivo de superar as

limitações impostas pela comunicação intermitente e restrições de energia, os dispositivos foram programados para funcionarem como nós encaminhadores (*relays*), realizando roteamento através de múltiplos saltos.

Outro projeto interessante é o ZebraNET [2]. Nesse trabalho, os autores investigaram os problemas inerentes às redes de sensores e à computação móvel em um cenário que possuía o objetivo de obter dados biológicos e de movimentação de zebras em uma reserva natural do Quênia. Dentre os desafios encontrados estão: o consumo eficiente de energia dos sensores, roteamento das informações, problemas relacionados à mobilidade como conectividade e isolamento de dispositivos etc. Em face a estes problemas, alguns protocolos foram propostos e implementados, servindo como base para diversos trabalhos nesta área.

Figura 1.4: *Projeto Zebranet documentado em [2]. Na figura, uma zebra com um colar responsável pela coleta de dados e encaminhamento de informações.*

O projeto Daknet[20] é outro trabalho no qual encontramos a aplicação de conceitos da Arquitetura DTN. Para alcançar o objetivo de prover acesso à Internet para áreas rurais que não dispunham de infaestrutura necessária, os autores propuseram aproveitar a infraestrutura de transporte local, como por exemplo ônibus e táxis, utilizando-os como Pontos de Acesso Móveis (MPAs), através dos quais a informação poderia ser distribuída, conforme a Figura 1.5.

Embora o acesso à Internet em tempo real não tenha sido contemplado por este modelo, muitas aplicações anteriormente inexistentes puderam ser implementadas, como por exemplo, o correio eletrônico.

Outros projetos interessantes relacionados à Arquitetura DTN podem ser encontrados em [21] e em [22].

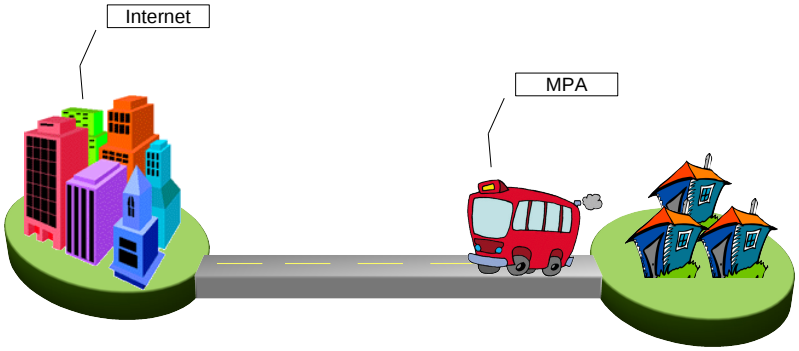

Figura 1.5: *Representação da rede Daknet. Neste modelo, os MPAs são responsáveis por levar a informação da região sem infraestrutura, para a região com infraestrutura Internet e vice-versa.*

1.2 Problemas Relacionados às Redes Tolerantes a Atrasos e Interrupções

As Redes Tolerantes a Atrasos e Interrupções apresentam diversos desafios que não estão presentes nas redes tradicionais. Muitos desses proveem da necessidade de se lidar com enlaces intermitentes, recursos escassos como energia e poder computacional, armazenamento restrito; além dos problemas anteriormente citados como altas taxas de perda de pacote, inexistência de caminhos fim-a-fim e alta latência.

Estes fatores impactam na interação entre os dispositivos causando problemas de roteamento, problemas relacionados à mobilidade, problemas de conservação de energia, dentre outros.

1.2.1 Roteamento na Arquitetura DTN

Como visto na Seção 1.1.2, os nós possuem ao longo do tempo oportunidades de contato determinadas por suas posições e alcance dos rádios. A medida que os dispostivos se encontram, caminhos são estabelecidas e dados são encaminhados através de múltiplos saltos. A função dos protocolos de roteamento é de determinar o melhor caminho entre fonte e destino, baseado em alguma métrica, e desta forma, saber se uma informação deve ou não ser encaminhada quando os dispositivos estão em contato.

Dentre as principais métricas de roteamento em uma DTN estão:

- **Probabilidade de Entrega**: calculada como a razão das mensagens entregues pelo número total de mensagens geradas na rede;
- **Retardo**: o tempo despendido desde o instante em que a mensagem foi gerada, até o momento de sua entrega;
- **Overhead**: medido como o número de mensagens excedentes geradas para efetuar o roteamento. Esta medida, *overhead*, é de grande importância pois cada transmissão extra consome energia adicional e portanto, em diversos estudos é adotada como uma aproximação do consumo de energia.

Segundo [23], os protocolos de roteamento podem ser classificados em três categorias: **(i)** mecanismos baseados em conhecimento sobre a rede, **(ii)** mecanismos de entrega designada e **(iii)** mecanismos oportunísticos.

Os protocolos de roteamento baseados em conhecimento [24, 25], utilizam dados históricos sobre a operação da rede ou a suposição de oráculos capazes de fornecer informações sobre esta dinâmica. A partir desses dados, são aplicados os algorítmos tradicionais como *Dijkstra* e *Bellman-Ford* para o estabelecimento de rotas.

Um exemplo pode ser encontrado em [26]. Neste trabalho, os autores estudaram uma rede veicular de alta mobilidade e assumiram que cada nó conhecia sua posição através da utilização de um dispositivo de posicionamento global,GPS. Também foi assumida a existência de um único nó destino, cuja posição era conhecida por todos. A partir destas suposições os autores propuseram estratégias de roteamento de forma a obter altas taxas de sucesso na entrega de mensagens.

Os protocolos baseados em entrega designada utilizam dispositivos programados a percorrer caminhos específicos, sendo responsáveis por coletar e encaminhar os dados. Como exemplo, encontra-se na literatura o trabalho [27]. Neste trabalho os autores verificaram que a existência da infraestrutura de transportes locais, como ônibus e táxis, poderia ser utilizada para estender a conectividade de uma região, ao custo do aumento do retardo na entrega das mensangens. Para isso, os dipositivos desta infraestrutura de transporte foram configurados como nós encaminhadores e passam a ser denominados MULEs, provendo uma maior capacidade de comunicação aos dispositivos da região por onde se movimentavam.

Outro exemplo é encontrado em [28]. Neste trabalho, foi proposta uma abordagem que utilizava um conjunto especial de nós, denominados *ferries*, responsáveis por percorrer caminhos fixos e capazes de prover serviços de comunicação aos demais nós da rede. A principal ideia na utilização desses dispositivos é de adicionar determinismo à rede e explorar tal característica para aumentar a probabilidade de sucesso na entrega de mensagens.

O roteamento oportunístico, por outro lado, utiliza um procedimento diferente. Neste, os nós encaminham uma ou mais cópias da mesma mensagem para os nós

vizinhos, sem utilizar suposições sobre o conhecimento da rede. A técnica mais simples a seguir essa premissa é o roteamento epidêmico [29, 30]. Em uma rede com roteamento epidêmico, a cada contato entre dispositivos, é trocada uma lista de mensagens que estão armazenadas em suas memórias locais. Em seguida, os nós identificam as mensagens que não possuem e requisitam-nas de seus pares. Ao final de um contato, ambos os dispositivos possuem a mesma relação de mensagens. Este roteamento é ótimo em termos da métrica Retardo, contudo, gera muitas cópias da mesma mensagem na rede(*Overhead*), consumindo muitos recursos como energia e espaço de armazenamento.

Um outro exemplo de roteamento oportunístico pode ser encontrado em [31]. Neste trabalho, os autores verificaram que o roteamento epidêmico é muito custoso e propuseram um novo algoritmo, com um número reduzido de cópias da mesma mensagem, denominado *Spray and Wait*. Este protocolo é constituido de duas fases: na fase *Spray*, para cada mensagem originada no nó fonte, L cópias são encaminhadas para nós *relays* nos primeiros contatos. Em seguida, na fase *Wait*, caso a mensagem ainda não tenha sido entregue diretamente, os nós *relays* aguardam até que um contato seja estabelecido com o nó destino e finalmente entregam a mensagem. Com este algoritmo é possível adaptar o parâmetro L para a rede em estudo, alcançando um retardo na entrega próximo ao ótimo enquanto mantem-se o *overhead* na rede baixo.

1.2.2 Problemas Relacionados à Mobilidade

O estudo da mobilidade e sua influência na dinâmica de um DTN é de grande importância na avaliação de performance nestas redes. Pesquisas recentes [32] demonstraram que a mobilidade é capaz de incrementar a capacidade em MaNET's, um subgrupo das Redes Desafiadoras. Além disso, a mobilidade é grande responsável pelas oportunidades de comunicação, ou seja, pelos momentos em que os dispositivos estão próximos o suficiente para estabelecer contato. Portanto, a acurácia da avaliação de performance depende de quão próximo o modelo de mobilidade utilizado está da realidade.

Um dos primeiros trabalhos a explorar o impacto da mobilidade em uma Rede Tolerante a Atrasos e Interrupções é encontrado em [33]. Neste trabalho, os autores utilizaram 54 *iMotes* distribuídos entre atendentes, durante a conferência IEEE Infocom 2005 em Miami. Esses dispositivos estavam programados para fazer varreduras utilizando rádios *bluetooth*, à procura de outros dispositivos em sua vizinhança.

A partir dos dados gerados, foi possível traçar a Função de Distribuição Cumulativa Empírica (CDFE) dos *tempos de contatos* e dos *tempos entre contatos* de pares de dispositivos. Ao avaliar a distribuição dessas duas métricas, os autores

demonstraram que tanto o tempo de contato quanto o tempo entre contatos possuem distribuições aproximadas por leis de potência, além de exibirem a propriedade de cauda pesada.

Uma das consequências deste resultado é que os modelos de mobilidade sintéticos, como o *Random WayPoint* e *Random Walk*, comumente usados na avaliação de performance, não são adequados, pois não se comportam dessa maneira. Além disso, a propriedade de cauda pesada encontrada nessas distribuições tem impacto negativo em alguns protocolos de encaminhamento, como demonstrado em [34] e [33].

Este resultado também é obtido em [35]. Neste trabalho, foi observado que nenhum modelo de mobilidade individual se assemelha à realidade de um cenário DTN, onde em geral, as pessoas movem-se de forma dependente umas das outras. Por isso, os autores propuseram um novo modelo de mobilidade em grupo, baseado em redes sociais e verificaram através de simulações, que sua proposta representava melhor a mobilidade real.

Em um outro artigo [36], os autores conduziram uma pesquisa similar a [33], mas com conclusões diferentes. Neste trabalho os autores identificaram uma dicotomia: a distribuição cumulativa empírica dos tempos entre contatos segue uma lei de potência até um certo período de tempo, denominado *tempo característico* e a partir deste valor a distribuição possui um decaimento exponecial.

Os autores verificaram ainda que este tempo característico é da ordem de 12 horas e que isso indicaria a periodicidade do comportamento humano. Além disso demonstraram que esta propriedade está presente em distribuições dos tempos entre contato obtidas de traces sintéticos, geradas pelo modelo Random Walk. Desta forma, eles concluem que os modelos de mobilidade, *Random WayPoint* e *Random Walk*, poderiam ser usados para avaliação de desempenho em DTN's.

Essa aparente incoerência é explicada em [37] e [38]. Nesses artigos os autores demonstram que são os limites da área em estudo dos cenários simulados e reais, que moldam os tempos de contato e entre contato à semelhança de distribuições exponenciais. Ainda nestes trabalhos, os autores demonstram que manipulando os parâmetros da simulação, de forma a evitar os limites, é possível reproduzir resultados com tempos de contato e entre contato distribuidos segundo leis de potência.

Outro trabalho interessante é encontrado em [39]. Nesse, os autores identificam um erro na obtenção das distribuições dos tempos entre contato em diversos estudos. Esse erro é devido a obtenção das amostras dos intervalos de contato e entre contatos em estudo. Os tempos entre contatos maiores que o tempo do experimento não são capturados, assim como tempos entre contatos que são menores que o intervalo de amostragem.

De forma a corrigí-los, os autores propuseram um algoritmo baseado em uma

técnica denominada *Estimador de Kaplan-Meier* ou *Estimador de Produto Limite* [39]. Após a correção dos erros, os autores traçaram as distribuições empíricas para o tempo entre contatos e caracterizaram-na como auto-similar.

Em [40], os autores caracterizaram os caminhos entre dispositivos fontes e destinos. Um caminho é uma sequência de dispositivos pelo qual a mensagem é transmitida, que se inicia no dispositivo fonte e termina no nó de destino. Devido a natureza móvel dos dispositivos, estes caminhos surgem e desaparecem. Ao caracterizar estes caminhos, os autores descobriram uma *"Explosão de caminhos"*, ou seja, após a mensagem chegar ao destino pelo caminho de menor retardo, vários outros caminhos são formados, possuindo retardos próximo ao ótimo. Este resultado explicaria porque algoritmos de encaminhamento que utilizam diferentes técnicas possuem avaliações de desempenho parecidas.

1.2.3 Conservação de Energia

A conservação de energia dos dispositivos, em cenários DTN, é também um dos grandes problemas desta arquitetura. Isto ocorre pois em geral, estes ambientes demadam a utilização de dispositivos com pouca capacidade de energia e grande mobilidade.

Estas restrições acabam por implicar em uma relação custo/benefício entre conservar energia e otimizar a comunicação. Por exemplo, para elevar o número de contatos, reduzindo o retardo na entrega de mensagens é preciso ampliar o alcance do rádio dos dispositivo consumindo mais energia.

Por outro lado, para minimizar o consumo de energia é preciso reduzir o alcance do rádio, ao custo de diminuir o número de contatos, consequentemente aumentando o retardo.

Este problema é um dos grandes desafios da Arquitetura DTN, frente aos novos dispositivos computacionais de dimensões reduzidas e alta capacidade de processamento. Diversas são as estratégias e técnicas com objetivo de diminuir o consumo de energia, estas serão exploradas no Capítulo 2.

1.3 Organização do Texto

Este capítulo teve como objetivo apresentar a Arquitetura DTN, sua origem, características e os desafios encontrados em sua implementação. No Capítulo 2 mais detalhes sobre o problema da conservação de energia serão explorados. Também serão discutidos os trabalhos relacionados a este problema e as soluções adotadas na literatura. Além disso, serão apresentados o escopo deste trabalho e a estratégia adotada na resolução do problema proposto.

No Capítulo 3 serão descritos os modelos analíticos utilizados como ferramentas na resolução do problema. Em seguida, no Capítulo 4, serão apresentadas as análises dos resultados obtidos através destes modelos. Também serão apresentados comparativos entre os resultados obtidos analiticamente e por simulações. Posteriormente, serão propostos e avaliados algoritmos para ajuste adaptativo da potência do sinal de transmissão de mensagens de descoberta.

Finalmente, no Capítulo 5, serão apresentadas as principais contribuições e conclusões deste trabalho, além de propostas para atividades futuras.

Capítulo 2

Consumo de Energia em Redes Tolerantes a Atrasos e Interrupções

Neste capítulo serão apresentados trabalhos relacionados ao problema da conservação de energia na Arquitetura DTN. Diversas técnicas serão abordadas, bem como suas características e classificações. Também serão apresentados a formalização do problema e os objetivo a serem alcançados.

2.1 Trabalhos Relacionados

O consumo de energia é um fator crítico na análise da performance de sistemas computacionais. Estudos como [41] e [42], indicam que o subsistema de comunicação é o maior responsável pelo consumo de energia de dispositivos utilizados em comunicações sem fio. E este consumo ocorre principalmente durante a transmissão de dados, seguido pelo processo de descoberta de dispositivos. Tal fato é agravado em Redes Tolerantes a Atrasos e Interrupções, pois estas redes são geralmente caracterizadas por cenários que demadam a utilização de dispositivos com pouca capacidade de energia e grande mobilidade.

Em [42], os autores demonstraram que o dreno de energia de um dispositivo celular Nokia 6600 é de: (i) 38,61mA para a o envio de uma mensagem de descoberta de dispositivos através da interface *Bluetooth*, (ii) 9,33mA para responder a uma mensagem de descoberta, (iii) 51,47mA para a transferência de dados pela interface *Bluetooth* e (iv) 38,68mA para uma ligação telefônica.

Para resolver esse problema, aumentando assim a autonomia dos nós e o tempo de vida operacional da rede, foram propostas diversas técnicas pela comunidade científica. Tais técnicas fazem parte do que denomina-se *Gerenciamento de Energia* e podem ser classificadas segundo seu princípio de operação, sendo as principais: (i)técnicas baseadas em mobilidade, (ii)técnicas direcionadas por dados, (iii) técnicas baseadas em ciclos de trabalho e (iv) técnicas baseadas no ajuste do alcance do rádio. É importante notar que estas classes de técnicas não são mutuamente excludentes, podendo haver técnicas com características de classes distintas.

2.1.1 Técnicas baseadas em Mobilidade

A mobilidade permite que os dispositivos se comuniquem com vizinhos mais próximos através de potências de transmissão menores, utilizando desta forma menos energia. Entretando, esta premissa pode nem sempre ser verdadeira, uma vez que ao reduzir a potência de transmissão o número de saltos necessários para que os dados cheguem ao destino aumenta. As técnicas de conservação de energia baseadas em mobilidade tem por objetivo encontrar um equilíbrio nessa relação. Nesta classe encontram-se as redes MULEs e *Message Ferries*, descritas na seção 1.2.1.

2.1.2 Técnicas direcionadas por Dados

A classe de técnicas direcionadas por dados está baseada na ideia de que em geral, a transmissão de dados é mais despendiosa em energia do que o processamento [43] e que o consumo de energia está diretamente ligado a quantidade de bits transmitidos. Por isto, esta classe tem como objetivo evitar a aquisição e principal-

mente a transmissão de dados desnecessários. Este objetivo é alcançado através do uso de compressão e codificação de dados[44]. Contudo, ao codificar a mensagem no nó fonte e decodificá-la no destino, é necessário um maior processamento dos dados e portanto, busca-se um equilíbrio entre codificação e consumo de processamento.

2.1.3 Técnicas baseadas em Ciclos de Trabalho

A classe de técnicas baseada em ciclos de trabalho(*duty cycle*), segue o princípio de que uma parcela considerável de energia pode ser economizada ao colocar os dispositivos em modo de conservação de energia, durante alguns períodos de tempo. Nesta abordagem, o principal problema está em como balancear os períodos de atividade e inatividade de modo a não degradar a performance da rede.

Encontramos na literatura diversos trabalhos que possuem o objetivo de resolver este problema. Segundo [23] e [45], estes mecanismos podem ser subclassificados em: **(a)** síncronos, **(b)** assíncronos, **(c)** baseados em controle de topologia e **(d)** sob demanda.

Mecanismos Síncronos

No mecanismo síncrono, os nós são desligados e religados em instantes sincronizados. Um exemplo é o PSM, muito utilizado em Redes Sem Fio convencionais e definido na especificação do protocolo IEEE 802.11 [46]. O PSM é constituido de dois componentes, o TSF(*Time Synchronization Function*) e um mecanismo de conservação de energia baseado na indicação de tráfego. O TSF utiliza um algoritmo distribuído para executar a sincronização. Este algoritmo está baseado na criação de pacotes de anúncio denominados *beacons*, responsáveis por anunciar a presença de um nó na rede e sincronizar os dispositivos. Essa sincronização é feita da seguinte forma: os nós disputam o meio segundo um protocolo padrão de *back-off*. Apenas uma estação adquire o meio e todas as demais ajustam seus relógios em função desta estação vencedora. Uma vez sincronizadas, as estações trocam pacotes denominados ATIM (*Ad-Hoc Traffic Indication Maps*), que são pacotes responsáveis por informar ao nós destinos para que permaneçam com seus rádios ligados quando há tráfego a ser transmitido.

Mecanismos Assíncronos

Nos mecanismos assíncronos, os dispositivos são programados para permanecer ativos durante períodos determinados de forma que estes períodos de atividades estejam sobrepostos aos períodos de atividades dos nós vizinhos. Com este objetivo em [23] foi proposto um protocolo denominado CAPM (*Context Aware Power Management*). Neste mecanismo os nós possuem períodos variáveis nos quais permanecem

ativos ou inativos. Assim que iniciam seu período de atividades, os dispositivos enviam mensagens contendo informações de identificação e o período pelo qual permanecerão ativos. Ao ajustar os intervalos de atividade e inatividade de forma assíncrona, os autores verificaram através de simulação, que este mecanismo é capaz de conservar energia enquanto mantém outras métricas, como taxa de entrega de dados, com valores semelhantes aos cenários em que o mecanismo não é aplicado.

Em [47] os autores estudaram uma rede de sensores com alta densidade de nós, na qual os dispositivos utilizavam um subsistema de comunicação baseado nos circuitos integrados CC2420[9], fabricados em acordo com o padrão IEEE 802.15.4. Foram executadas diversas medições com o objetivo de avaliar o consumo de energia em cada estado do rádio. A partir destes valores, os autores calcularam o consumo total para envio de um pacote de dados levando em consideração todas as fases do protocolo (aquisição do meio, envio do pacote, retransmissões em caso de colisões e recebimento do reconhecimento(ACK) da transmissão) incluindo também o gasto de energia com as transições de estado. Baseado nestes resultados, foi proposta uma política para ativação do subsistema de comunicação capaz de reduzir o consumo.

Mecanismos baseados em Controle da Topologia

Os mecanismos baseados em controle de topologia estão geralmente associados a redundância e alta densidade de nós. Desta forma é possível dinamicamente adaptar a topologia da rede, permitindo sua operação com o menor número de nós ativos e desta forma conservar energia.

Em [48], os autores analisaram a rede através de um grafo $G = (V, U)$. Neste modelo, os vértices $v_i \in V$ representavam os dispositivos, as arestas $e_{i,j} \in U$ representavam os enlaces entre estes dispositivos e os pesos das arestas $P(e_{i,j})$, a potência do sinal utilizado para estabelecer o enlace $e_{i,j}$. A partir desse modelo, foi proposto um algoritmo distribuido para encontrar a árvore geradora mínima $T = (X, W)$. Este subgrafo, possui os mesmos vértices que o grafo G, mas seu conjunto de arestas W é um subconjunto de U de forma que $\sum_{\forall e_{i,j} \in W} P(e_{i,j})$ seja mínimo.

Em [49] foi proposto um protocolo com objetivo de conservar energia, mantendo apenas alguns nós ativos, enquanto garantia a conectividade da rede. Para isto, os autores propuseram um algoritmo distribuído em que a área da rede era dividida em células e dentro de cada célula os dispositivos elegiam um dispositivo coordenador, responsável por manter a conexão com as células vizinhas.Esta eleição de nós coordenadores é baseado na quantidade de energia disponível para este nó e realizada periodicamente de forma a homogenizar o consumo de energia dentre todos os nós da célula.

Mecanismos sob Demanda

Os mecanismos sob demanda, tem por objetivo ativar o rádio apenas quando em contato com nós vizinhos. Para isso geralmente empregam um rádio auxiliar de menor potência e consumo, que permanece sempre ativo, sendo capaz de detectar contatos e ativar o rádio principal quando necessário.

Em [23], os autores verificaram que ao utilizar esta técnica em redes esparsas, com baixa densidade de nós por área, muitos contatos eram perdidos. Por isso propuseram utilizar o rádio principal em conjunto com o rádio auxiliar para a descoberta de contatos, ativando-o com menos freqüência que o rádio auxiliar. Através de simulações, ficou demonstrado que mesmo com a utilização de ambos os rádios é possível reduzir o consumo de energia.

2.1.4 Técnicas baseadas no Ajuste do Alcance do Rádio

É consenso entre os pesquisadores que o consumo de energia do subsistema de comunicação é uma função da potência do sinal empregado na transmissão. Portanto, ajustar o alcance dos rádios dos dispositivos é uma alternativa dentre os métodos de conservação de energia.

Fundamentadas nesta premissa, encontram-se as técnicas baseadas no ajuste do alcance do rádio. Ajustar o alcance dos rádios garante não apenas a redução no consumo de energia como também, uma otimização no reuso espacial, gerando menos ruído e interferências. Contudo, quanto menor a potência do sinal transmitido, menores serão as oportunidades de contato entre dispositivos. Isto resulta em altos retardos e baixa capacidade de transmissão.

Em [50] encontramos um dos primeiros estudos sobre o alcance dos rádios de dispositivos em uma rede sem fio, anteriormente denominada *"Packet Radio Networks"*. Neste trabalho, os autores propuseram uma rede de dispositivos sem fio distribuidos uniformemente em uma área com densidade homogênea λ. O tempo foi dividido em segmentos e os dispositivos foram considerados fixos (sem mobilidade) com raios de transmissão homogêneos. O protocolo de acesso ao meio utilizado era o ALOHA segmentado. Cada dispositivo tinha probabilidade de transmissão p por segmento e um alcance circular de raio r. Também foi suposta uma matriz de tráfego uniforme, ou seja, cada dispositivo comunicava-se com os demais uniformemente. Baseado nestas premissas, os autores propuseram uma nova métrica, *progresso médio de um pacote por segmento* e demonstram que o raio de alcance medido em termos de número de dispositivos vizinhos que maximizava esta métrica, era aproximadamente 6.

Em [51] e [52] foram propostas técnicas baseadas no fato de que uma vez estabelecido o enlace de comunicação entre os nós é possível calcular o indicador do nível

de sinal recebido (RSSI) a cada troca de mensagens e desta forma estimar a distância entre os dispositivos. Assim é possível ajustar o alcance do sinal de transmissão, consumindo menos energia tanto pela redução da intensidade do sinal transmitido, quanto pela redução do número de retransmissões devido a colisões. Contudo, o problema principal destas técnicas consiste em determinar com maior acurácia esse indicar e por consequência, a distância entre os nós.

Em [53], os autores estudaram o impacto da variação da potência de transmissão na capacidade, conectividade e no consumo em uma Rede Ad Hoc Móvel. Foi verificado, assim como em [54], que existe um valor para o menor raio comum de transmissão, R_{min}^{com}, capaz de manter toda a rede conectada. Além disso, também foi demonstrado que a capacidade da rede é inversamente proporcional ao alcance de transmissão e que portanto, em DTN's, casos em que a rede não está totalmente conectada, é possível aumentar a capacidade e simultaneamente reduzir o consumo.

Em [55], os autores propuseram uma nova métrica, a *eficiência de distância-energia de primeiro salto*, calculada como a razão entre o progresso médio de uma mensagem durante a primeira transmissão e o consumo de energia desta transmissão. Para o cálculo do progresso médio de uma mensagem, foi proposta uma estratégia de transmissão, baseada na ideia de que o envio de uma mensagem só deve ocorrer: **(i)** entre os nós fonte e destino, ou então; **(ii)** entre um nó fonte e um nó vizinho, desde que este seja o nó mais próximo do destino.

Após derivar a métrica proposta analiticamente e validá-la através de simulações, os autores demonstram que o raio ótimo varia em função da densidade dos nós quando a potência de transmissão decai em função da distância por um fator de 2. Quando este fator é 4, praticamente não há variação no raio ótimo. Além disso, os autores concluem que, para redes com grandes distâncias, o valor do raio de transmissão que maximiza a métrica proposta é aproximadamente constante.

Com o objetivo de encontrar o alcance do sinal de transmissão com boa performance, em [56], os autores propuseram uma nova métrica, *joule por bit-metro*. Tal medida é definida como a energia requerida para mover uma unidade de informação por uma unidade de distância na direção do dispositivo receptor.

Ao aplicar o modelo proposto a cenários de redes de sensores, os autores demonstraram que a distância ótima, em termos da métrica definida, é controlada pelo fator de decaimento do sinal em relação a distância. Portanto, pequenos saltos são mais apropriados e alta densidade de nós é requerida para elevar a performance. Além disso, foi demonstrado que a posição ideal do nó sorvedouro é próximo ao centro da rede. E no caso desse nó ser móvel, modificar a topologia de forma que sejam formados pequenos aglomerados ao invés de uma grande rede conectada, pode reduzir o consumo.

2.2 Definição do Problema e Proposta de Resolução

Como visto, existem na literatura diversos estudos com o objetivo de reduzir o consumo de energia. Entretanto, um problema pouco explorado ocorre quando o enlace de comunicação ainda não foi estabelecido e os contatos acontecem de forma oportunista. Neste cenário, é necessário decidir qual a melhor potência de sinal a ser usada na transmissão de mensagens de descoberta.

Por exemplo, em redes *bluetooth* o processo de descoberta ocorre da seguinte forma: o dispositivo mestre, aquele que busca por dispositivos vizinho, passa para o estado de *inquiry*. Neste estado são enviadas mensagens de descoberta, a cada $312,5\mu s$ em pares de frequência de forma a cobrir 32 dos 79 canais possíveis. Após percorrer um par de canais, o dispositivo aguarda nestes canais por um período de $312,5\mu s$, por eventuais respostas, como pode ser visto na Figura 2.1.

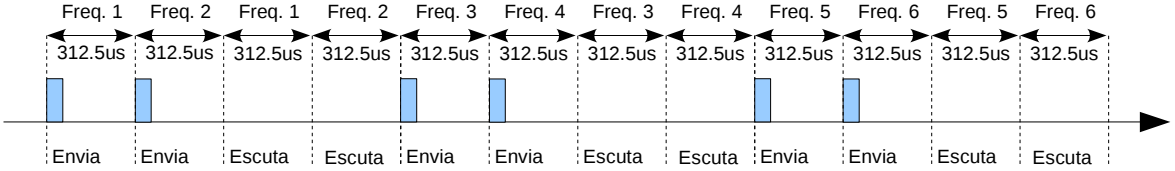

Figura 2.1: *Mecanismo de descoberta de dispositivos vizinhos em uma rede Bluetooth.*

Ess processo é repetido, de forma a percorrer todos os canais. Assim em todo o processo de descoberta podem ser enviados até 1600 mensagens de descoberta e todo o processo pode demorar até 10,24 segundos.

De forma a avaliar a porcentagem de energia consumida no processo de descoberta diversos estudos foram realizados. Em [42], foi verificado que este processo consome uma parcela considerável de energia, que pode chegar a até aproximadamente 75% do consumo de uma transmissão de 1MB em um cenários com dois dispositivos *bluetooth*. Em [57] foi desenvolvido um aparato para medição do consumo de dispositivos *bluetooth* em vários cenários. Tais experimentos demonstraram que o consumo durante os estados de *inquiry* e *inquiry scan*, momentos em que o dispositivo envia mensagens de *probe* e escuta por *acks*, não podem ser desconsiderados.

Em [6], os autores desenvolveram um dispositivo para mensurar o consumo de rádios *bluetooth* em cada estado de operação. Os resultados podem ser visualizados na Tabela 2.1. Como pode ser visto, os estados resposáveis pelo processo de descoberta de dispositivos vizinhos, *inquiry* e *inquiry scan* são os principais responsáveis pelo consumo de energia, chegando a $36,76\%$ do consumo total no cenário analisado.

Essa situação é ainda mais grave em alguns cenários de Redes Tolerantes à Atrasos e Interrupções, caracterizados por redes esparsas com baixa densidade de nós

Estado de Operação Bluetooth	Consumo
Startup and initialization	66 mW
Idle	6.6 mW
Inquiry	231 mw
Inquiry Scan	139 mW
Page	208 mW
Page Scan	149 mW
Transmission and Reception(Master)	56 mW
Transmission and Reception(Slave)	152 mW

Tabela 2.1: *Consumo dos diversos estados de operação em dispositivos bluetooth, como observado em [6].*

por área e formados por dispositivos com baixa capacidade de armazenamento e energia. Por exemplo, um dispositivo isolado, programado para enviar uma mensagem de descoberta em intervalos de tempo regulares, pode exaurir toda energia apenas com esse tipo de transmissão.

Como visto anteriormente, uma alternativa seria desligar o rádio durante estes períodos, entretando, como não existe conhecimento sobre a rede, contatos importantes podem ser perdidos. Outra alternativa, seria ajustar os intervalos de envio de mensagens de descobertas, como proposto em [42].

No presente trabalho, o problema abordado será a redução do consumo durante o processo de descoberta de dispositivos vizinhos, através da redução da intensidade do sinal de transmissão. Portanto, o termo *consumo de energia* irá referir-se ao consumo desse processo.

Os dispositivos considerados serão dispositivos sem fio com uma composição típica conforme a Figura 2.2, sendo constituídos de pelo menos três subsistemas principais: **(i)** subsistema de processamento, incluindo microcontrolador e memória para processamento local; **(ii)** subsistema de comunicação, consistindo de rádio e antena; e **(iii)** um subsistema de armazenamento de energia, consistindo de uma bateria. Dependendo da especificidade da rede, outros componentes podem existir, como no caso das Redes de Sensores que demandam um subsistema sensorial.

Formalmente, seja $C(p)$ a potência consumida na transmissão de uma mensagem de descoberta(*probe*) com a intensidade de sinal p. Suponha ainda que o tempo de transmissão de um *probe* seja de 1 segundo e que portanto, o consumo de energia

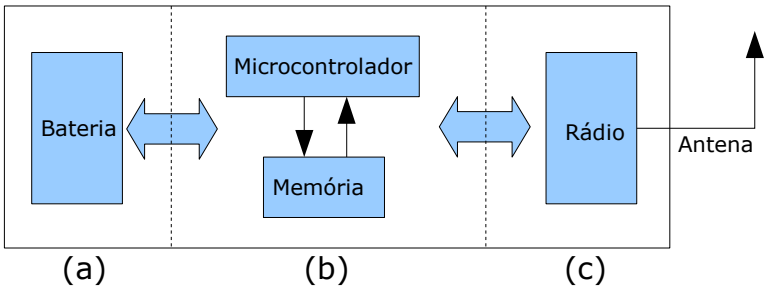

Figura 2.2: *Composição típica de dispositívos sem fio. (a)Subsistema de energia, (b)Subsistema de processamento, (c)Subsistema de Comunicação.*

da transmissão de uma mensagem de descoberta, dado em joules, é igual:

$$C(p) = p \tag{2.1}$$

Assim é assumido que o consumo para a transmissão de um *probe* com a potência de sinal p é uma função linear de p. Desta forma, quanto maior for a intensidade do sinal sendo transmitido, maior será o consumo do dispositivo. Tal relação está de acordo com dipositivos reais, como pode ser visto na Figura 2.3. A figura apresenta dados de consumo de um dispositivo de comunicação, CC1000[3], obtidos de sua especificação.

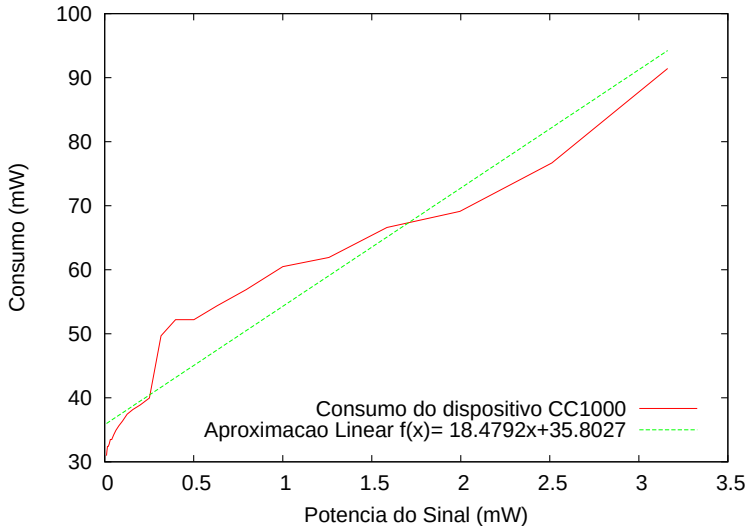

Figura 2.3: *Consumo do dispositivo CC1000[3] em função da potência do sinal de transmissão. Os valores foram obtidos utilizando voltagem de 3.6 V e frequência do sinal de 868 MHz.*

Seja t um instante de tempo qualquer durante a evolução de uma rede e suponha que durante o intervalo $[0, t]$, foram executados P *probes* pelo dispositivo n. O

consumo total de n até o instante t será dado por :

$$C_{n,total}(t) = \sum_{i=0}^{P} C(p_i) \tag{2.2}$$

Segundo a Equação 2.2, o consumo total é função da quantidade de *probes* executados no período de análise, como também da intensidade do sinal utilizado para transmitir tais mensagens de descoberta. Desta forma, existem duas possibilidades para reduzir o consumo: **(i)** diminuir a intensidade do sinal transmitido ou, **(ii)** diminuir o número de *probes*, aumentando o intervalo de tempo entre estes.

Como visto na Seção 2.1.4, diversos estudos foram feitos com o objetivo poupar energia através da redução do alcance do rádio dos dispositivos. Entretanto, as técnicas apresentadas visavam otimizar a potência do sinal de transmissão em tempo de design da rede e não previam adaptação do alcance do rádio. Além disso as métricas definidas, por serem calculadas geométricamente e levarem em consideração a evolução da transmissão dos dados na direção do dispositivo receptor, supunham que os dispositivos conhecessem as posições de seus vizinhos e do nó destino, o que pode não ocorrer em redes reais.

A redução do consumo através do estudo do intervalo entre mensagens de descoberta foi abordada em [42]. Neste trabalho, os autores propuseram um modelo analítico com o objetivo de investigar a relação entre intervalo entre *probes* e a probabilidade de perder oportunidades de contato. A partir desse modelo, também foi proposto um algoritmo adaptativo, capaz de decidir qual o melhor intervalo entre mensagens de descoberta, durante a evolução de uma DTN.

O objetivo desta dissertação é investigar a relação entre o consumo de energia e as oportunidades de contato e desta forma, propor novas maneiras para reduzir o consumo de energia, utilizando-se da redução da intensidade do sinal de transmissão de mensagens de descoberta.

Para isto, será formalizada uma generalização do modelo proposto em [42], abrangendo a relação do alcance do rádio com a probabilidade de perder oportunidades de contato. Esta métrica, a *Probabilidade de Perder uma Oportunidade de Contato*, será calculada através de um novo modelo analítico. A partir desta nova medida, será possível avaliar a relação entre o consumo de energia e as oportunidades de contato e encontrar o menor consumo para um dado nível de perda.

A métrica proposta possui diversas vantagens em relação as demais métricas definidas em [55] e [56], pois, as hipóteses adotadas estão mais próximas de um cenário real DTN, não necessitando de conhecimento da posição espacial dos nós e permitindo uma adaptação da potência do sinal durante a evolução da rede.

É importante notar que consideramos que o melhor intervalo entre *probes*, como calculado em [42], já está sendo observado e estaremos interessados apenas na re-

dução da intensidade do sinal de transmissão das mensagens de descoberta.

Os resultados obtidos pelo modelo proposto, serão comparados a resultados oriundos de cenários DTN simulados. Esta comparação tem como objetivo, verificar a correção deste modelo e ajustá-lo a diferentes cenários.

Por fim, a partir das conclusões obtidas, será proposto um algoritmo capaz de identificar e adaptar a intensidade de sinal de transmissão durante a evolução da rede para reduzir o consumo e desta forma, aumentar a autonomia dos dispositivos.

Capítulo 3

Modelagem de uma Rede Tolerante a Atrasos e Interrupções

São vários os modelos necessários para representar a dinâmica de uma Rede Tolerante a Falhas e Interrupções. Neste trabalho identificamos como principais: **(a)** o modelo de propagação do sinal e **(b)** o modelo de contato entre dispositivos. Este capítulo tem por objetivo, apresentar esses modelos.

3.1 Modelos de Propagação do Sinal

A comunicação entre dispositivos em um meio sem fio requer a trasmissão por ondas de rádio, uma forma de radiação eletromagnética. Esta radiação está sujeita às condições do ambiente, sendo influenciada por umidade, obstáculos, multipercurso, mobilidade, altura da antena entre outros. Tais condições acabam por gerar fenômenos como desvanecimento (*fading*), interferências e ruídos.

Modelar de forma correta esses fenômenos é um fator crítico ao analisar redes sem fio, uma vez que determinar quando uma comunicação é realizada com sucesso é dependente desses fatores. Contudo, como cada enlace de comunicação pode encontrar condições diferentes, seria intratável definir em uma única equação toda a atenuação durante o caminho(*path loss*). Por isso, são encontrados na literatura [58] diferentes modelos, para diferentes tipos de canais, sob diversas condições.

Neste trabalho será utilizado o modelo de Friis[59], não com o objetivo de caracterizar completamente a propagação do sinal, mas sim, determinar o alcance do rádio de um dispositivo, permitindo desta forma calcular quando dois ou mais nós são capazes de estabelecer um enlace de comunicação.

De maneira geral, a perda de sinal ou atenuação do sinal (*Path loss*) L, é a redução da intensidade de potência da onda eletromagnética transmitida, que propaga através do espaço. Esta perda pode ocorrer por diversos fatores dentre os quais atenuação no espaço-livre(*free-space loss*), refração, reflexão, absorção em obstáculos e geralmente é dada em decibéis(dB) pela equação:

$$L = 10 \log_{10} \frac{P_{rx}}{P_{tx}} \tag{3.1}$$

onde P_{rx} e P_{tx} são respectivamente as potências de recepção e transmissão medidas em Watts.

A atenuação em espaço-livre[1] é a única prevista no modelo de Friis, proposto em [59], um dos modelos mais simples encontrados na literatura. Neste modelo, a potência do sinal recebido a uma distância d do ponto emissor pode ser calculada por:

$$P_{rx}(d) = P_{tx} S(d) A \tag{3.2}$$

onde P_{rx} e P_{tx} são respectivamente as potências de recepção e transmissão medidas em miliWatts(mW), S é a quantidade de potência por unidade de área (Wm^{-2})

[1] O espaço-livre é definido como um meio homogêneo: possui constituição física identica em todos os pontos; isotrópico: que possui características de permissividade, permeabilidade magnética e condutividade constantes em relação a direção de propagação e sem obstaculos capazes de absorver a energia da onda sendo propagada, sendo a única perda, a causada pela expansão natural da energia da onda de rádio no espaço.

e A é a diretividade da antena (*antenna aperture*) medida em unidade de área (m^2). A quantidade de potência por unidade de área, a uma distância d do ponto emissor, pode ser calculada por:

$$S(d) = \frac{1}{4\pi d^2} \tag{3.3}$$

A diretividade da antena (*antenna aperture*), é a área efetiva sob a qual a antena é capaz de absorver a energia de uma onda eletromagnética incidente e é definida como:

$$A = G_{tx} G_{rx} \frac{\lambda^2}{4\pi} \tag{3.4}$$

onde G_{tx} e G_{rx} são respectivamente os ganhos das antenas em dBi e λ é o comprimento da onda em metros. Em antenas isotrópicas, uma abstração matemática de uma antena que irradia potência de maneira uniforme em todas as direções, o ganho em direcionalidade é $G_{tx} = G_{rx} = 1$. Desta forma, a potência do sinal recebido a uma distância d do ponto emissor, utilizando-se antenas isotrópicas, segundo o modelo de Friis é dado por:

$$P_{rx}(d) = P_{tx}(\frac{\lambda}{4\pi d})^2 \tag{3.5}$$

ou em Decibéis(dBm) como:

$$P_{rx}(d) = P_{tx} - 20log_{10}(\frac{4\pi d}{\lambda}) \tag{3.6}$$

Além da suposição da antena isotrópica, este modelo ainda supõe: (i)a existência de uma linha de visão entre as antenas(*line-of-sight*), (ii)que não existe o efeito de multi-caminho, e (iii) que as antenas estejam devidamente alinhadas e polarizadas.

Uma vez determinada a forma com a qual o sinal se propaga, é necessário verificar se este alcança o nível de serviço ou nível de sinal necessário na recepção, de forma a determinar se a informação recebida pode ser diferenciada do ruído. Este nível de sinal é determinado através da relação entre Sinal-Ruído, (*Signal to Noise Ratio* - SNR) recomendada pelos órgãos normativos para um determinado serviço de comunicação, sendo estabelecida em função de fatores como frequência de operação, modulação e principalmente taxa de erro de bits (*Bit Error Rate* - BER). Esta relação é dada por:

$$snr(d) = \frac{P_{rx}(d)}{\Xi} \tag{3.7}$$

onde Ξ, medido em Watts(W), indica o nível de ruído ambiente. Esta relação

geralmente é expressa em Decibéis(dB):

$$snr(d) = 10log_{10}(\frac{P_{rx}(d)}{\Xi}) \tag{3.8}$$

O limiar de audição β, representa o menor valor para a relação Sinal-Ruído, a partir da qual é possível distinguir o sinal recebido do ruído do ambiente e geralmente é expresso em função da taxa de erro de bits(BER), da frequência de operação e da modulação utilizada. Este valor pode ser também encontrado nas especificações dos dispositivos de comunicação na forma de menor potência de sinal distinguível de ruído, usualmente denominada de *Sensitividade* e dado em mW ou dBm.

O modelo de propagação de Friis basicamente representa a potência do sinal em função da distância através de zonas círculares equipotentes centradas no dispositivo emissor. Quanto maior a distância em relação ao dispositivo emissor, menor será a intensidade do sinal recebido.

Desta forma, é possível estabelecer uma *área de alcance máximo* para um dispositivo n_i, $A_{n_i,max}$, determinada pela sensitividade β e pela máxima potência de transmissão empregada pelo rádio $P_{tx,max}$:

$$A_{max} = \pi R_{max}^2 \tag{3.9}$$

onde R_{max} é dado por:

$$R_{max} = \frac{\lambda}{4\pi}\sqrt{\frac{P_{tx,max}}{\beta}} \tag{3.10}$$

Além da área de alcance máximo, é possível estabelecer também a área de alcance efetivo para um dispositivo n_i, $A_{n_i,ef}(p)$, que pode ser obtida de forma semelhante ao alcance máximo, através da utilização das equações 3.9 e 3.10, substituindo apenas a máxima potência de transmissão $P_{tx,max}$ pela potência p efetivamente empregada em um dado instante.

$$A_{ef} = \pi R_{ef}^2 \tag{3.11}$$

onde R_{ef} é dado por:

$$R_{ef} = \frac{\lambda}{4\pi}\sqrt{\frac{P_{tx,ef}}{\beta}} \tag{3.12}$$

As regiões definidas pelo modelo de propagação, área de alcance máximo e área de alcance efetivo, pode ser visualizadas na Figura 3.1. Estas regiões possuem grande importância pois a partir delas, serão definidos os processos de contato entre dispositivos na Seção 3.2.

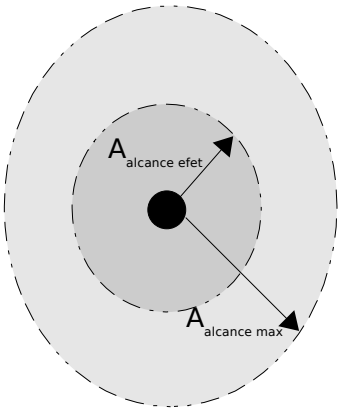

Figura 3.1: *Área de alcance máximo delimita a máxima região na qual é possível estabelecer uma conexão. Contudo, como é permitido ao nó reduzir a potência de transmissão para uma potência efetiva $p < P_{max}$, a região na qual a conexão pode ser estabelecida, pode ser reduzida para a àrea de alcance efetivo.*

3.2 Modelo de Contatos entre Dispositivos

Um dos principais processos a ser avalido em uma Rede Tolerantes a Atrasos e Interrupções é o processo de contato entre dispositivos. É a partir desse processo, que são obtidas diversas métricas importantes na avaliação de desempenho de tais redes.

Por se tratar de um tema recente, ainda não existe consenso quanto a definição de contato. Em sua forma mais usual, encontrada em [34] e [36], o termo *contato* é definido como o intervalo de tempo no qual os dispositivos trocam mensagens ou são capazes de identificar seus pares. Embora diversos trabalhos adotem esta abordagem, são encontrados na literatura definições variadas.

Em [60], os autores definem contato como a coexistencia de inscrições de alunos em disciplinas em uma universidade, desta forma, se dois ou mais alunos estão matriculados em cursos no mesmo horário, estes estarão em contato. Em [61], os autores utilizam *traces* de dispositivos em uma rede IEEE 802.11 com infraestrutura, obtidos em um campus da Universidade de Dartmounth para identificar o processo de contato entre os nós. Neste trabalho, os autores definem que dois dispositivos estão em contato caso estejam simultaneamente conectados ao mesmo ponto de acesso.

Embora sejam encontradas diferentes definições, o processo de contato em diversos estudos parece apresentar resultados semelhantes: as distribuições dos tempos de contato e dos tempos entre contatos são caracterizadas por distribuições

exponenciais[36] e leis de potência [34] e [60].

Neste trabalho, o termo *contato* terá uma nova abordagem. Seja $N = \{n_1, n_2, \ldots, n_k\}$ o conjunto de dispositivos em estudo, de forma que $|N| = k$ seja a quantidade total de dispositivos. Suponha que os dipositivos sem fio n_i estejam dispostos segundo uma distribuição uniforme na área em estudo E de tamanho finito $|E|$ e que a densidade de nós por área seja $\rho = \frac{k}{|E|}$.

Seja $Pos(n_i, t)$ a posição do dispositivo n_i no instante de tempo t. Suponha ainda que o movimento dos dispositivos é definido por um modelo de mobilidade que mantenha esta distribuição uniforme.

Assuma que todo dispositivo explora o ambiente segundo algum algoritmo de descoberta. Quando um dispositivo n_i envia uma mensagem de descoberta (um *probe*), todos os demais dispositivos que recebem esta mensagem respondem com outra mensagem, de reconhecimento. Baseado nesta troca de dados, os dispositivos decidem se devem estabelecer uma conexão.

Em um instante de tempo t, um dispositivo n_i estabelece uma *Oportunidade de Contato* com outro dispositivo qualquer n_j, quando n_j encontra-se na área de alcance máximo de n_i, $A_{n_i,max}$, definida pela utilização da máxima potência de transmissão $P_{n_i,max}$. Posto de outra forma, seja $Op_t(n_i, n_j)$ a variável aleatória que representa o evento da oportunidade de contato do nó n_i com outro nó qualquer n_j no instante t:

$$Op_t(n_i, n_j) = \begin{cases} 1 & \text{se } Pos(n_j, t) \in A_{n_i,max}; \\ 0 & \text{caso contrário.} \end{cases} \tag{3.13}$$

Portanto, uma *Oportunidade de Contato* indica quando uma comunicação pode ocorrer, contudo, para que os dispositivos realmente estabeleçam um enlace de comunicação é necessário que durante esta oportunidade, troquem mensagens de descoberta utilizando potências de transmissão compatíveis com a distância na qual estão seus pares.

Para um dado par (n_i, n_j), seja τ_i e υ_i os instantes em que uma oportunidade de contato é respectivamente iniciada e finalizada. Estes intervalos, $\Delta_i = \upsilon_i - \tau_i$, são denominados *intervalos de contato*. Suponha que as durações desses intervalos Δ_i sejam variáveis aleatórias independentes e identicamente distribuídas, com *Função de Distribuição Cumulativa (FDC)*, $F_\Delta(x) = P[\Delta < x] = \int_0^x f_\Delta(x)d_x$ e média $\mathbb{E}[\Delta] = \frac{1}{\mu}$.

Seja ainda Θ_i o intervalo de tempo entre dois contatos consecutivos, Δ_i e Δ_{i+1}, ou seja, $\Theta_i = \tau_{i+1} - \upsilon_i$. Estes intervalos são denominados *intervalo entre contatos*. Suponha que as durações destes intervalos sejam também variáveis aleatórias independentes e identicamente distribuídas, com FDC $F_\Theta(x) = P[\Theta < x] = \int_0^x f_\Theta(x)d_x$ e média $\mathbb{E}[\Theta] = \frac{1}{\nu}$.

A Figura 3.2 ilustra estes instantes e intervalos de tempo. Entre os instantes t_0 e t_2, os dispositivos n_i e n_j permanecem em um intervalo de contato Δ_i. Logo após, os nós se separam e permanecem separados até o instante t_4, quando iniciam um novo intervalo de contato.

É importante notar que por serem variáveis *independentes e identicamente distribuidas (i.i.d)*, os índices i de Δ_i e Θ_i podem ser desconsiderados sem perda de generalidade. Além disso, como supomos também independência entre dispositivos, estas variáveis são válidas para qualquer par de nós.

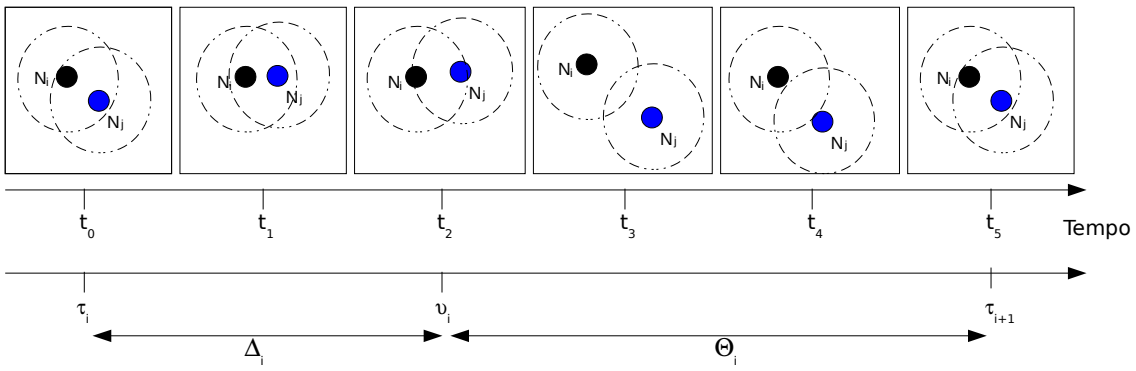

Figura 3.2: *Representação dos intervalos de contato e entre contatos.*

Como visto, um nó n_i identificará uma oportunidade de contato com o nó n_j, se durante esta oportunidade, Δ, uma mensagem de descoberta com potência de transmissão p for enviada e $Pos(n_j, t) \in A_{n_i,ef}(p)$, onde $A_{n_i,ef}(p)$ é a área de alcance efetivo de n_i, obtida através da utilização de uma potência $0 \leq p \leq P_{max}$ no instante t.

Na Figura 3.3, o dispositivo n_i possui uma oportunidade de contato com os dispositivos n_j e n_k. Entretanto, como a potência efetiva utilizada na transmissão da mensagem de descoberta é menor do que a potência máxima, apenas o dispositivo n_j, que está no interior da área de alcance efetivo, é capaz de recebê-la.

Desta forma, também é possível definir uma variável aleatória $D_{t,p}(n_i, n_j)$, que representa o evento de descoberta do nó n_j pelo nó n_j, dado que estes dispositivos estão em um intervalo de contato:

$$D_{t,p}(n_i, n_j) = \begin{cases} 1 & \text{se } Pos(n_j, t) \in A_{n_i,ef}(p); \\ 0 & \text{caso contrário.} \end{cases} \tag{3.14}$$

Existem diversas formas de executar *probes* em um intervalo de tempo, contudo o objetivo deste trabalho é encontrar a potência utilizada no envio dessas mensagens de descoberta que possibilite consumir a menor quantidade de energia possível, com

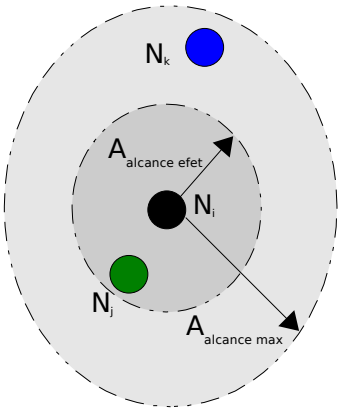

Figura 3.3: *Representação da oportunidade de contato e do evento de descoberta dos nós n_i, n_j e n_k. Emboras os nós estejam em contato, apenas o dispositivo n_j será descoberto.*

a menor probabilidade de perder uma oportunidade de contato.

O problema de identificar o melhor instante para executar *probes* foi tratado em [42]. Neste trabalho os autores demonstraram que dentre todas as estratégias de execução de *probes* com a mesma taxa média, a estratégia que minimiza a perda de oportunidades de contato é aquela em que o intervalo de envio de mensagens de descoberta é constante. Portanto, dentre todas as estratégias de descoberta, considere aquela em que os *probes* são executados em intervalos de tempos constantes T. Além disso, suponha que cada *probe* seja transmitido com uma potência $0 \leq p_{ef} \leq P_{max}$.

Seja Γ a variável aleatória que indica o evento de perder uma oportunidade de contato, ou seja:

$$\Gamma = \begin{cases} 1 & \text{se o contato é perdido;} \\ 0 & \text{caso contrário.} \end{cases} \tag{3.15}$$

Como visto, uma oportunidade de contato é perdida quando esta ocorre entre dois instantes de *probe*, ou quando todos os *probes* executados durante este intervalo de contato não possuem potência suficiente para alcançar outro nó. Portanto, para encontrar $P[\Gamma = 1]$, é preciso determinar a quantidade de *probes* que ocorrem em um intervalo de contato.

Sob a política de *probes* constantes, seja Ω a variável aleatória que representa o número de *probes* executados durante o intervalo de contato Δ, por um dos dispositivos. É possível encontrar a distribuição de Ω, $P[\Omega = n]$, ao se condicionar esta

probabilidade no tamanho do intervalo Δ e observar que $\Omega \approx \frac{\Delta}{T}$. Desta forma:

$$P[\Omega = n] = \int_0^\infty P[\Omega = n \mid x \leq \Delta < x + d_x] f_\Delta(x) d_x \tag{3.16}$$

A probabilidade $P[\Omega = n \mid x \leq \Delta < x + d_x]$ pode ser encontrada aplicando-se a mesma técnica utilizada no trabalho [62]. Por ser uma variável inteira, o valor exato de Ω será dado por uma aproximação de $\Omega \approx \frac{x}{T}$. Dependendo do instante em que o contato for iniciado, a variável Ω pode assumir os valores, $\Omega = \lfloor \frac{x}{T} \rfloor$ ou $\Omega = \lfloor \frac{x}{T} \rfloor + 1$, como pode ser visto na Figura 3.4.

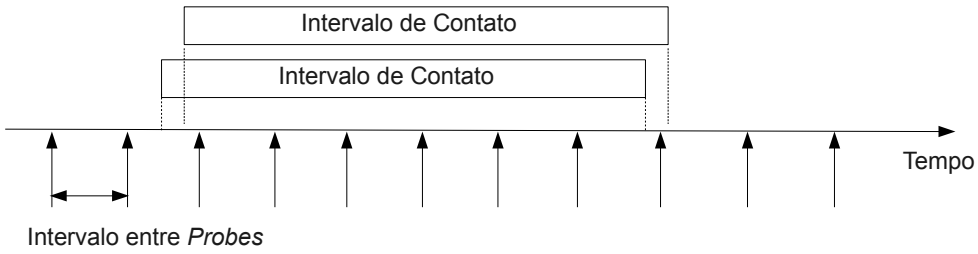

Figura 3.4: *Aproximação do valor da variável inteira* $\Omega = \frac{C}{T}$. *Dependendo do instante em que o contato for iniciado, a variável* Ω *será* $\lfloor \frac{x}{T} \rfloor$ *ou* $\lfloor \frac{x}{T} \rfloor + 1$.

Portanto, condicionando novamente esta probabilidade, ao inicio do intervalo de contato, e supondo que este instante de início é distribuido uniformemente entre dois intervalos de *probe*, ou seja, no intervalo $[iT, (i+1)T] | i \in \mathbb{N}$, temos que:

$$P[\Omega = n \mid x \leq \Delta < x + d_x] = \begin{cases} 0 & \text{se } \frac{x}{T} < (n-1); \\ \frac{x}{T} - n + 1 & \text{se } (n-1) \leq \frac{x}{T} < n; \\ n - \frac{x}{T} + 1 & \text{se } n \leq \frac{x}{T} \leq (n+1); \\ 0 & \text{se } \frac{x}{T} > n+1 \ . \end{cases} \tag{3.17}$$

Substituindo a Equação 3.17 na Equação 3.16 e observando que a $P[\Omega = n \mid x \leq \Delta < x + d_x]$ é diferente de zero apenas no intervalo $[n-1, n+1]$, teremos que a distribuição do número de *probes* em um intervalo de contato será dada por:

$$P[\Omega = n] = \int_0^\infty P[\Omega = n \mid x \leq \Delta < x + d_x] f_\Delta(x) d_x \tag{3.18}$$

$$= \int_{n-1}^{n} (\frac{x}{T} - n + 1) f_\Delta(x) d_x + \int_n^{n+1} (n - \frac{x}{T} + 1) f_\Delta(x) d_x \tag{3.19}$$

É importante observar que a Equação 3.17 é válida para $n > 0$ e portanto, precisamos considerar o caso $n = 0$, separadamente. A variável Ω será igual a zero

apenas quando o intervalo de contato for menor que um intervalo de probes, $\Delta \leq T$, portanto:

$$P[\Omega = 0] = \frac{1}{T} \int_0^T (T - x) f_\Delta(x) d_x \tag{3.20}$$

Uma vez encontrada a distribuição de Ω, é possível determinar a probabilidade de perder uma oportunidade de contato para um intervalo de contato qualquer. Para isto, basta condicionar $P[\Gamma = 1]$ no número de *probes* executados no intervalo Ω:

$$P[\Gamma = 1] = \sum_{n=0}^{\infty} P[\Gamma = 1 \mid \Omega = n] P[\Omega = n] \tag{3.21}$$

$$P[\Gamma = 1] = P[\Gamma = 1 \mid \Omega = 0] P[\Omega = 0] + \sum_{n=1}^{\infty} P[\Gamma = 1 \mid \Omega = n] P[\Omega = n] \tag{3.22}$$

A probabilidade de perder uma oportunidade de contato, dado que nenhum *probe* é executado, $P[\Gamma = 1 \mid \Omega = 0] = 1$, ou seja, se não houver um *probe*, o contato sempre será perdido.

Caso $\Omega \geq 1$, o contato não será detectado se em cada *probe* a potência de transmissão não for suficiente para alcançar o dispositivo vizinho. Como supomos que os eventos de envio de *probes* são independentes, $P[\Gamma = 1 \mid \Omega = n]$, para $n \geq 1$ é dado por:

$$P[\Gamma = 1 \mid \Omega \geq 1] = \sum_{n=1}^{\infty} P[\Gamma = 1 \mid \Omega = n] P[\Omega = n] \tag{3.23}$$

$$= \sum_{n=1}^{\infty} \binom{n}{0} P[D = 1]^0 P[D = 0]^{n-0} P[\Omega = n] \tag{3.24}$$

$$= \sum_{n=1}^{\infty} P[D = 0]^n P[\Omega = n] \tag{3.25}$$

Portanto a probabilidade de perder uma oportunidade de contato, será igual a:

$$P[\Gamma = 1] = P[\Omega = 0] + \sum_{n=1}^{\infty} P[D = 0]^n P[\Omega = n] \tag{3.26}$$

Para encontrar $P[D = 0]$ é necessário observar que, dado que os dispositivos estão em um intervalo de contato, uma mensagem de descoberta do dispositivo n_i não alcançará o nó n_j se e somente se $Pos(n_j, t) \notin A_{n_i,ef}(p)$. Como a distribuição dos dispositivos pela área de estudo E é considerada uniforme, a probabilidade de uma descoberta de n_i não ocorrer é dada por:

$$
\begin{aligned}
P[D_{t,p_{ef}} = 0] &= 1 - \frac{A_{n_i,ef}(p_t)}{A_{n_i,max}} \\
P[D_{t,p_{ef}} = 0] &= 1 - \frac{p_{ef,t}}{P_{max}} \qquad (3.27)
\end{aligned}
$$

Assim, substituindo a Equação 3.27 na Equação 3.26 obtemos:

$$
P[\Gamma = 1] = P[\Omega = 0] + \sum_{n=1}^{\infty} (1 - \frac{p_{ef,t}}{P_{max}})^n P[\Omega = n] \qquad (3.28)
$$

É importante notar que este modelo é uma generalização do modelo proposto em [42]. Nesse trabalho os autores também calculam a probabilidade de perder uma oportunidade de contato, denominada por eles P_{miss}, contudo, em seu cenário, não é possível variar a potência de transmissão utilizada na descoberta de um dispositivo vizinho.

Comparando-se os dois modelos e restringindo-se a variação da potência de transmissão à potência máxima, a probabilidade de não haver descoberta $P[D_{t,p} = 0]$ passa a ser zero e portanto $P[\Gamma = 1] = P_{miss}$, uma vez que:

$$
\begin{aligned}
P[\Gamma = 1] &= \frac{1}{T} \int_0^T (T - x) f_\Delta(x) d_x \qquad (3.29) \\
P[\Gamma = 1] &= \frac{T - \mathbb{E}[\Delta | \Delta \leq T]}{T} F_\Delta(T) \qquad (3.30)
\end{aligned}
$$

Nesta seção foram definidos os passos necessários para o cálculo da probabilidade de perder uma oportunidade de contato para um par de dispositivos qualquer, quando é utilizada a estratégia de *probes* constantes.

Esta probabilidade é uma métrica importante na avaliação de desempenho de Redes Tolerantes a Atrasos e Interrupções, principalmente, no estudo da redução do consumo de energia.

Pela Equação 3.26, verifica-se que a probabilidade de perder uma oportunidade de contato está relacionada com a distribuição dos tempos de contato Δ e com a potência de transmissão efetiva, p_{ef}, utilizada durante a descoberta de dispositivos vizinhos.

Além disso, analisando o modelo proposto, é nítida a existência de uma relação custo/benefício entre a probabilidade de perder uma oportunidade de contato e o consumo de energia.

Quando o consumo é máximo, ou seja, a potência de transmissão utilizada é máxima, a probabilidade de não haver descobertas é zero, $P[D = 0] = 0$, e portanto a probabilidade de perder uma oportunidade de contato é mínima $P[\Gamma = 1] =$

$P[\Omega = 0]$, ocorrendo apenas para contatos com durações menores que o intervalo entre *probes*.

Por outro lado, quando o consumo é mínimo, ou seja, a potência de transmissão utilizada é zero, a probabilidade de não ocorrer descoberta é máxima, $P[D = 0] = 1$, e a probabilidade de perder uma oportunidade de contato será máxima $P[\Gamma = 1] = P[\Omega = 0] + \sum_{n=1}^{\infty} P[\Omega = n] = 1$.

Estudar esta relação e encontrar mecanismos capazes de reduzir o consumo de energia mantendo-se um certo nível de contato é o principal objetivo deste trabalho.

Capítulo 4

Aplicação do Modelo Proposto

Neste capítulo serão analisados os resultados da aplicação do modelos proposto, descritos no Capítulo 3, ao supor distribuições Exponenciais e de Pareto para o intervalo de contato entre dispositivos. Além disso, os resultados do modelo analítico serão comparados aos resultados obtidos por simulação, com o objetivo de testar sua acurácia.

A distribuição exponencial tem sido adotada para representar o tempo de contato e entre contatos em diversos estudos [32] e [63], principalmente por tornar os modelos e cálculos mais tratáveis. Além disso, essa suposição é suportada por simulações numéricas, com o modelo de mobilidade *Random Waypoint*, realizados em[64].

A distribuição de Pareto, tem sido observada em outros estudos empíricos [34], [36] e [60]. Nestes estudos, os *traces* reais de movimentação e de contato são caracterizados e indicam que a Função de Distribuição Cumulativa Complementar, ou seja, a cauda da Função de Distribuição Cumulativa se assemelha às leis de potência.

Essa aparente incoerência é explicada em [37] e [38]. Nestes artigos os autores demonstram que são os limites da área em estudo dos cenários simulados e reais, que moldam os tempos de contato e entre contato à semelhança de distribuições exponenciais. Ainda nestes trabalhos, os autores demonstram que manipulando os parâmetros da simulação, de forma a evitar os limites, é possível produzir resultados com tempos de contato e entre contato distribuidos segundo leis de potência.

4.1 Distribuição Exponencial

Supondo que o intervalo de contato entre dispositivos seja distribuído exponencialmente, teríamos que $F_{\Delta}(x) = P[\Delta < x] = 1 - e^{-\mu x}$. Utilizando a equação 3.26, a probabilidade de perder uma oportunidade de contato $P[\Gamma = 1]$ seria dada por:

$$P[\Gamma = 1] = P[\Omega = 0] + \sum_{n=1}^{\infty} P[D = 0]^n P[\Omega = n] \quad (4.1)$$

$$P[\Gamma = 1] = \frac{e^{-\mu T} - 1 + \mu T}{\mu T} + \frac{2(cosh(\mu T) - 1)}{\mu T} \frac{e^{-\mu T}(1 - \frac{p_{ef}}{P_{Max}})}{1 - (e^{-\mu T}(1 - \frac{p_{ef}}{P_{Max}}))} \quad (4.2)$$

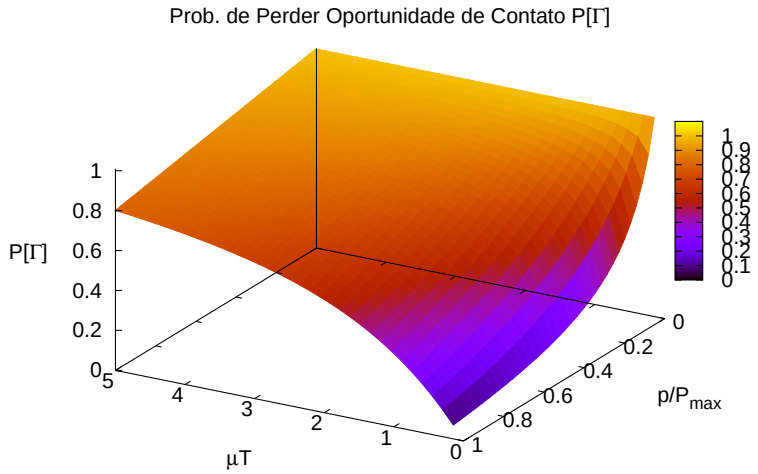

Figura 4.1: *Probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função de* μT *e* $\frac{p_{ef}}{P_{max}}$.

Como discutido na Seção 2.1, a diminuição do consumo de energia durante a descoberta de nós vizinhos possui duas abordagens principais: **(i)** diminuir o número de *probes* aumentando o intervalo de tempo entre estes ou, **(ii)** diminuir a intensidade do sinal transmitido. Isto é evidenciado pela Equação 4.1 em que é possível verificar que a probabilidade de perder uma oportunidade de contato $P[\Gamma = 1]$ é uma função de μT e da relação entre a potência efetiva p_{ef} e potência máxima P_{max}.

Pela abordagem **(i)**, o consumo de energia poderia ser reduzido, aumentando-se o intervalo entre *probes*. Para uma taxa de contatos μ constante, isto implica em aumentar a relação μT, que representa o número médio de contatos que ocorre em um intervalo entre *probes*, e consequentemente implica também em aumentar a probabilidade de perder um contato.

Na Figura 4.2 é possível observar o comportamento de $P[\Gamma = 1]$ em função de μT, para diferentes valores de p_{ef}/P_{max}. Para pequenos valores de μT, quando o intervalo de contato é maior que o intervalo entre *probes*, $P[\Gamma = 1]$ está próximo de

zero. Isto ocorre pois serão executados diversos *probes* em um intervalo de contato e portanto, a probabilidade de perder essa oportunidade diminui. É possível notar ainda que quanto maior a relação $\frac{p_{ef}}{P_{max}}$ menor a probabilidade de perder um contato.

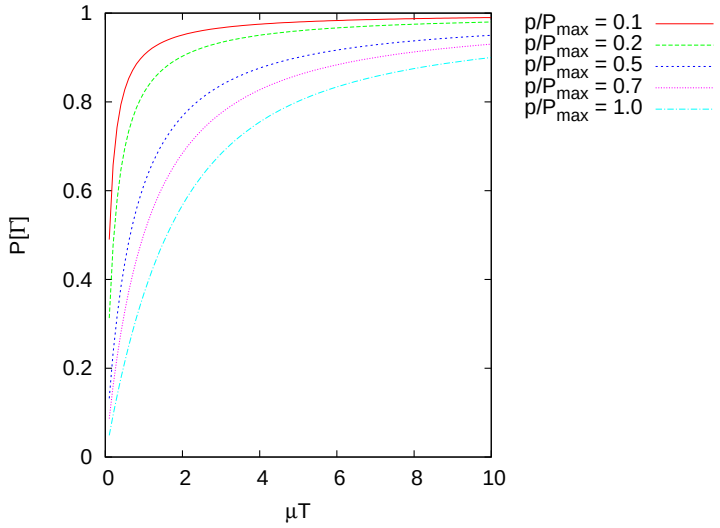

Figura 4.2: *Probabilidade de perder oportunidade de contato* $P[\Gamma = 1]$ *em função de* μT*, para diferentes valores de* $\frac{p_{ef}}{P_{max}}$.

Pela abordagem **(ii)**, o consumo de energia pode ser reduzido, diminuindo-se a potência efetiva de transmissão do sinal de uma mensagem de descoberta, utilizando-se apenas uma fração da potência máxima P_{max}. Quanto menor for essa relação, menor o consumo e maior a probabilidade de perder uma oportunidade de contato.

Na Figura 4.3, é possível verificar esta relação. Além disso, é possível notar que para pequenos valores de μT, a curva $P[\Gamma = 1]$ apresenta um 'joelho' acentuado que vai sendo desfeito a medida que os valores μT aumentam.

Este comportamento possibilita reduzir o consumo de energia sem que a probabilidade de perda aumente demasiadamente. Na Figura 4.4 é possível verificar que para um valor de $\mu T = 0.10$, em que o tempo médio de contato é dez vezes maior que o intervalo entre *probes*, uma redução da potência efetiva para 0.8 vezes o valor da potência total, o que equivale aproximadamente a redução do consumo em 20%, aumentaria a probabilidade de perder novas oportunidades de contato em apenas $2,1\%$. Quando μT possui um valor de 1.00, um redução de consumo de 20% aumenta a probabilidade de perder novas oportunidades em aproximadamente $8,5\%$.

4.1.1 Validação por Simulação - Metodologia e Resultados

Com o objetivo de validar o modelo proposto, um simulador orientado a eventos discretos foi desenvolvido. A implementação deste novo simulador foi necessária, pois os simuladores já existentes, NS-2[65] e *The ONE*[66], não são capazes de suportar as variações nas potências de transmissão de seus dispositivos, característica

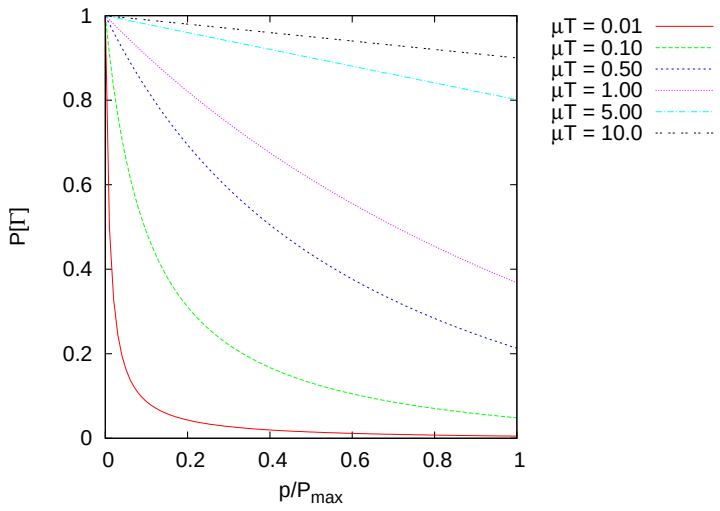

Figura 4.3: *Probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função de* $\frac{p_{ef}}{P_{max}}$*, para diferentes valores de* μT*.*

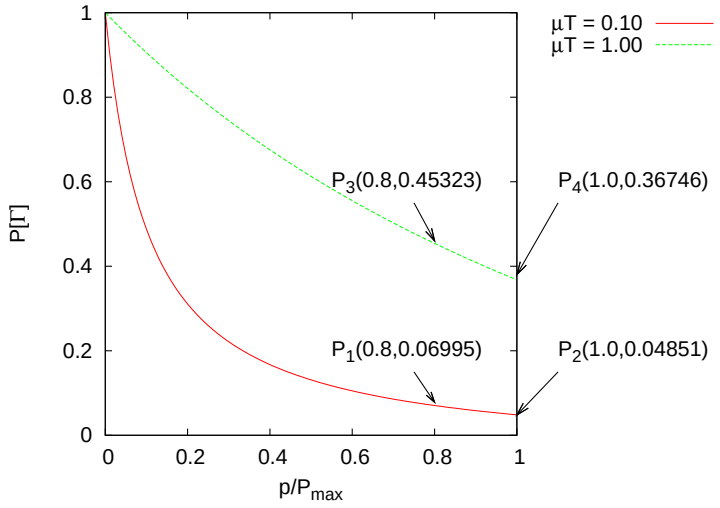

Figura 4.4: *Probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função de* $\frac{p_{ef}}{P_{max}}$*. Quando* $\mu T = 0.10$*, é possível observar que uma redução de consumo de* 20% *implica em um aumento da probabilidade de perder novas oportunidades de contato em apenas* $2,1\%$*. Quando* μT *possui um valor de* 1.00*, um redução de consumo de* 20% *aumenta a probabilidade de perder novas oportunidades em aproximadamente* $8,5\%$*.*

essencial ao estudo. Mais detalhes sobre a implementação do simulador, seu código-fonte e funcionamento podem ser encontradas no Apêndice A.

Ao simular a dinâmica de uma DTN, diversos parâmetros são necessários para caracterizar o cenário em estudo como: número de dispositivos móveis, dimensões da região, o modelo de mobilidade dos nós, características do rádio como potência máxima de transmissão, sensitividade do rádio, ruído do ambiente etc. Estes parâmetros devem ser escolhidos cuidadosamente pois podem comprometer os re-

sultados obtidos.

Para escolher os parâmetros do cenário de forma que estes estejam o mais próximo de um cenário real, foi aplicada a técnicalabel de ajuste de parâmetros desenvolvida em [7], em dados de movimentação reais obtidos em [8]. Os parâmetros de caracterização do rádio foram baseados nas especificações do transmissor CC2420[9] integrado em diversos dispositivos móveis, dentre eles o Tmote[67], amplamente utilizado em cenários de Redes de Sensores. Todos os parâmetros utilizados nas simulações podem ser visualizados na Tabela 4.1.

Parâmetros da Simulação			
Característica do Cenário		**Características do Rádio e Sinal**	
Tempo de Simulação:	3000 s	**Potência Máxima de Tx.:**	0,1 mW
Modelo de Mobilidade:	RandomWaypoint	**Sensitividade da antena:**	-85 dBm
Velocidade Máxima:	1,58 m/s	**Frequência:**	2,4 GHz
Velocidade Mínima:	1,52 m/s	**Comprimento de Onda:**	0,125 m
Tempo de Pausa:	0 s	**Alcance máximo**	57,43 m
Dimensões da região:	600x800 m²		
Número de Dispositivos:	100		

Tabela 4.1: *Parâmetros utilizados na simulação. Estes parâmetros foram obtidos atravéz da técnica de ajuste de parâmetros desenvolvida em [7], em dados de movimentação reais obtidos em [8]. Os parâmetros de caracterização do rádio foram obtidos das especificações do transmissor [9].*

A principal métrica a ser obtida é a probabilidade de perder uma oportunidade de contato. Para obter esta métrica, foram gerados 10 *traces* de movimentação, utilizando o programa *MobiSim*, [68]. Cada *trace* corresponde a uma rodada de simulação. Desta forma, garantimos que os resultados não serão dependentes de uma movimentação específica. Em cada rodada de simulação, foram avaliados os contatos entre todos os dispositivos, par-a-par, gerando portanto $\frac{n!}{(n-2)!}$ amostras para n dispositivos avaliados.

Contudo, é importante notar que para simplificar a análise, supomos que os dispositivos durante toda a simulação possuíam a mesma potência máxima de transmissão e utilizavam a mesma potência efetiva. Desta forma, se um dispositivo N_i está em contato com outro N_j, então N_j também está em contato com N_i, e portanto $P[\Gamma_{i,j} = 1] = P[\Gamma_{j,i} = 1]$. Assim, é necessário analisar apenas $\frac{n!}{2(n-2)!}$ amostras.

Também supomos independência entre os dispositivos, bem como independência

entre intervalos de contato e entre contatos, possibilitando desta forma calcular a média amostral da probabilidade de perder uma oportunidade de contato, $\mathbb{E}[\hat{\Gamma}]$, a partir da Equação 4.3:

$$\mathbb{E}[\hat{\Gamma}] = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} F(i,j)}{2n} \tag{4.3}$$

Onde a função F é definida como:

$$F_(i,j) = \begin{cases} P[\Gamma_{i,j}] & \text{se } i \neq j; \\ 0 & \text{caso contrário.} \end{cases} \tag{4.4}$$

Após concluir todas as rodadas, um nova média foi gerada. Tal procedimento foi realizado para os diversos valores da potência de transmissão efetiva dos dispositivos e de intervalo entre *probes*, gerando a superfície visualizada na Figura 4.5.

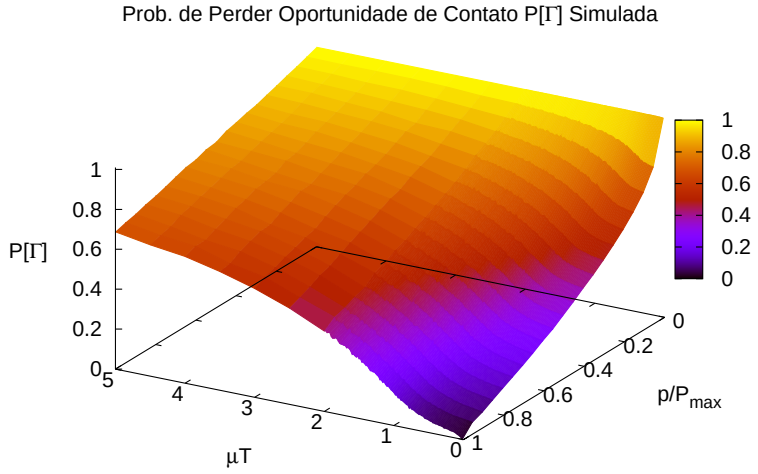

Figura 4.5: *Probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função de* $\frac{p}{P_{max}}$*, obtida através de simulação.*

Como pode ser visto na Figura 4.5, o resultado da simulação é muito semelhante ao que fora previsto no modelo, Figura 4.1. Esta semelhança fica ainda mais clara, quando observamos as projeções de $P[\Gamma = 1]$ sobre os eixo $\frac{p_{ef}}{P_{max}}$ e μT nas Figuras 4.6 e 4.7.

As pequenas diferenças entre os resultados do modelo e os obtidos por simulação, podem ser explicadas pelas suposições realizadas no modelo. Por exemplo, embora a escolha dos parâmetros tenha sido feita de forma a ajustar o *trace* sintético ao *trace* real, não foi possível impor uma distribuição dos tempos de contato entre dispositivos à movimentação dos nós, sendo esta avaliação feita após a simulação.

Para tal avaliação foram obtidos o tempo médio de contato e a Função de Distribuição Cumulativa Empírica dos tempos de contato. O tempo de contato médio

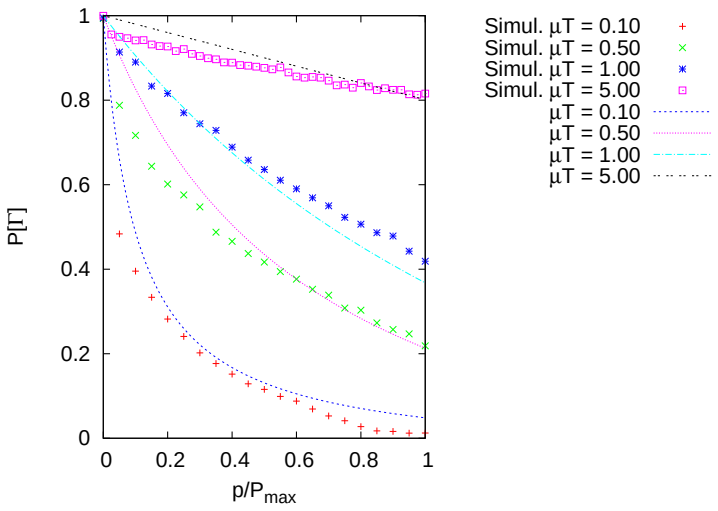

Figura 4.6: *Probabilidade de perder oportunidade de contato* $P[\Gamma = 1]$ *simulada, em função de* $\frac{p_{ef}}{P_{max}}$*, para diferentes valores de* μT*.*

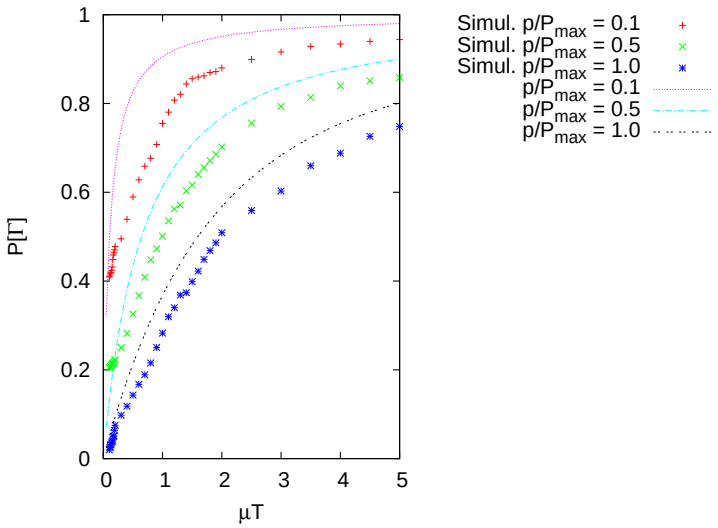

Figura 4.7: Probabilidade de perder oportunidade de contato $P[\Gamma = 1]$ simulada, em função de μT, para diferentes valores de $\frac{p_{ef}}{P_{max}}$.

foi de 46.1298 segundos, tendo portanto, taxa de 0.02167796 contatos por segundo.

Analisando a Função de Distribuição Cumulativa Empírica obtida de uma das rodadas, verificamos que esta é semelhante a uma distribuição exponencial, como pode ser visto na Figura 4.8. Porém, uma das diferenças observadas entre a Distribuição Exponencial e a ditribuição empírica é que esta última apresenta probabilidades maiores para contatos de pequena duração, 1 segundo aproximadamente. Também apresenta probabilidades maiores para contatos com duração próxima da média, entre 40s e 50s.

Além da suposição de tempos de contatos com distribuições exponenciais, outra hipótese importante do modelo é a de uma distribuição espacial uniforme dos nós

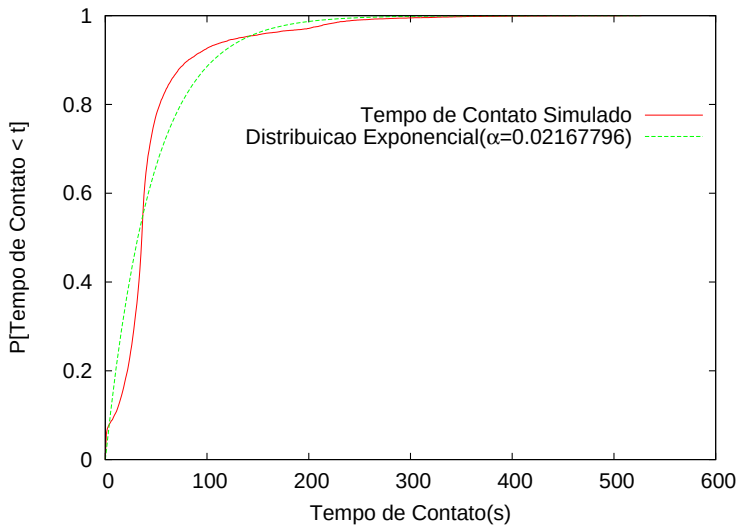

Figura 4.8: *Função de Distribuição Cumulativa Empírica dos tempos de contato entre dispositivos.*

pela área de simulação. Em conformidade com outros estudos [4], a distribuição espacial obtida com o modelo de mobilidade *Random Waypoint*, apresentou uma concentração no centro da região de simulação, como pode ser visto na Figura 4.9. Tal fato também serve para explicar as variações encontradas entre os resultados analíticos e simulados. Entretanto, tais variações não chegam a invalidar o modelo proposto.

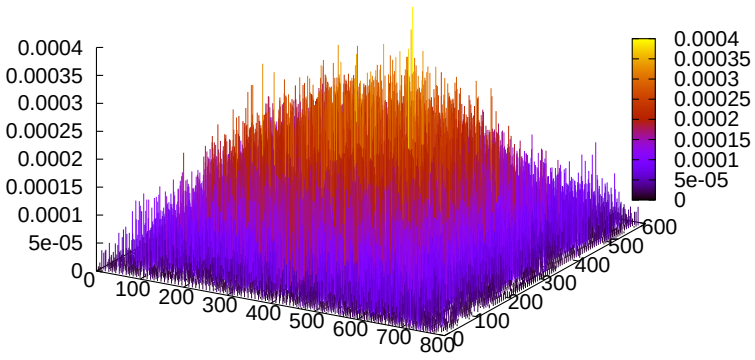

Figura 4.9: *Distribuição dos dispositivos pela área em estudo. Em conformidade com outros estudos [4], a distribuição espacial obtida com o modelo de mobilidade* Random Waypoint, *apresenta uma concentração no centro da região de simulação.*

4.2 Distribuição de Pareto

Como visto na Seção 3.2, diversos estudos demonstraram que as distribuições complementares dos tempos de contato e dos tempos entre contatos são caracterizadas por leis de potência [34] e [60].

Portanto, de forma a garantir a compatibilidade do modelo formulado com as características encontradas em dados reais, suponha que o intervalo de contato entre dispositivos seja distribuído segundo uma Distribuição de Pareto:

$$F_{\Delta}(x) = P[\Delta < x] = 1 - \frac{x_{min}}{x}^{k}$$

para $k > 1$, com função densidade de probabilidade $f_{\Delta}(x)$ dada por:

$$f_{\Delta}(x) = \frac{k x_{min}^{k}}{x^{k+1}}$$

.

Esta distribuição implica em uma distribuição complementar segundo uma Lei de Potência da forma:

$$P[\Delta > x] = \frac{x_{min}}{x}^{k}$$

Utilizando a equação 3.26, $P[\Gamma = 1]$ seria dada por:

$$P[\Gamma = 1] = P[\Omega = 0] + \sum_{n=1}^{\infty} P[D = 0]^{n} P[\Omega = n] \tag{4.5}$$

Onde $P[\Omega = 0]$ e $\sum_{n=1}^{\infty} P[D = 0]^{n} P[\Omega = n]$ são dados respectivamente pelas Equações 4.6 e 4.7:

$$P[\Omega = 0] = 1 + \frac{k x_{min}}{T(1-k)} - \frac{x_{min}^{k}}{T^{k}(1-k)} \tag{4.6}$$

$$\begin{aligned} \sum_{n=1}^{\infty} P[D = 0]^{n} P[\Omega = n] &= \sum_{n=1}^{\infty} [(1-n)(\frac{x_{min}}{(n-1)})^{k} - (n+1)(\frac{x_{min}}{(n+1)})^{k} + 2n(\frac{x_{min}}{n})^{k} + \\ &+ \frac{k}{T}(\int_{n-1}^{n} (\frac{x_{min}}{x})^{k} dx - \int_{n}^{n+1} (\frac{x_{min}}{x})^{k} dx)] * (1 - \frac{p_{ef}}{P_{max}})^{n} \end{aligned} \tag{4.7}$$

A Equação 4.5 não possui forma fechada e portanto deve ser resolvida numericamente, entretanto, para realizar tal tarefa é necessário determinar os parâmetros k e x_{min}.

4.2.1 Validação por Simulação - Metodologia e Resultados

Para identificar os parâmetros necessários, novas rodadas de simulação foram realizadas. Estas simulações utilizavam os mesmos parâmetros da Tabela 4.1, com uma modificação na dimensão da área simulada, de 600×800 para 2000×2000 metros quadrados. Esta modificação teve por objetivo eliminar os efeitos da borda e obter uma distribuição dos tempos de contato segundo uma lei de potência, como previsto em [37] e [38].

Como pode ser visto na Figura 4.10, a cauda da distribuição dos tempos de contato se assemelha a uma lei de potência com parâmetros $k = -2$ e $x_{min} = 27$, para os valores de intervalo de contato entre 30 e 300 segundos. Porém, é possível notar que nos intervalos abaixo e acima destes valores, as curvas são bem diferentes.

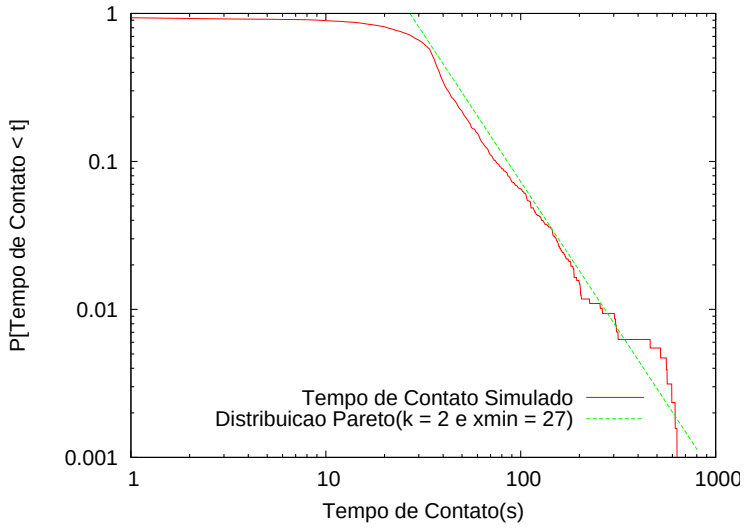

Figura 4.10: *Função de Distribuição Cumulativa Complementar Empírica dos tempos de contato entre dispositivos.*

Uma vez determinado os parâmetros da distribuição, foi desenvolvido um programa para efetuar o cálculo numérico da probabilida de perder uma oportunidade de contato $P[\Gamma = 1]$ para diversos pontos de p_{ef} e T, com erro $\epsilon < 0.0001$. O resultado pode ser visualizado na Figura 4.11.

Novamente, é possível verificar que os métodos de redução do consumo de energia condizem com o esperado, ou seja, é possível reduzir o consumo ao diminuir a potência do sinal de transmissão, ou ao aumentar o intervalo entre *probes*. Contudo, diferentemente das distribuições exponenciais, é possível visualizar na Figura 4.12 que os cenários cujo tempo de contato sejam distribuidos segundo Pareto não exibem a forma de 'joelho' em suas probabilidades de perder uma oportunidade de contato.

Porém, ainda assim é possível reduzir o consumo e obter uma redução não-linear da probabilidade de perder novos contatos. Na Figura 4.14 é possível verificar que para um valor de $T = 50s$, uma redução da potência efetiva para 0.8 vezes o valor

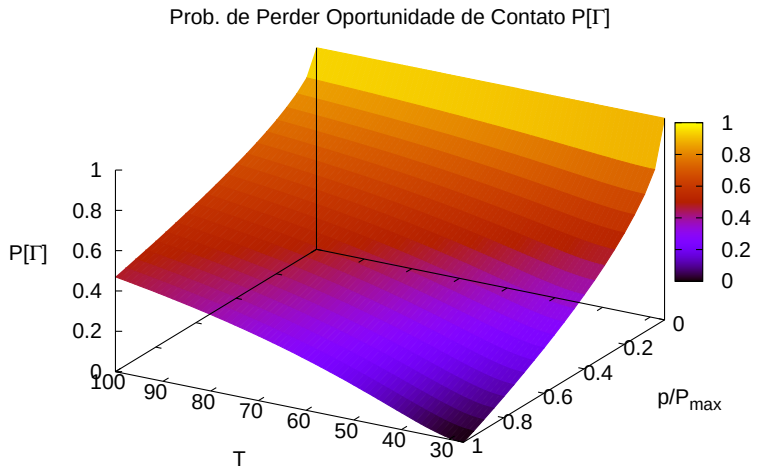

Figura 4.11: *Probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função de* T *e* $\frac{p_{ef}}{P_{max}}$.

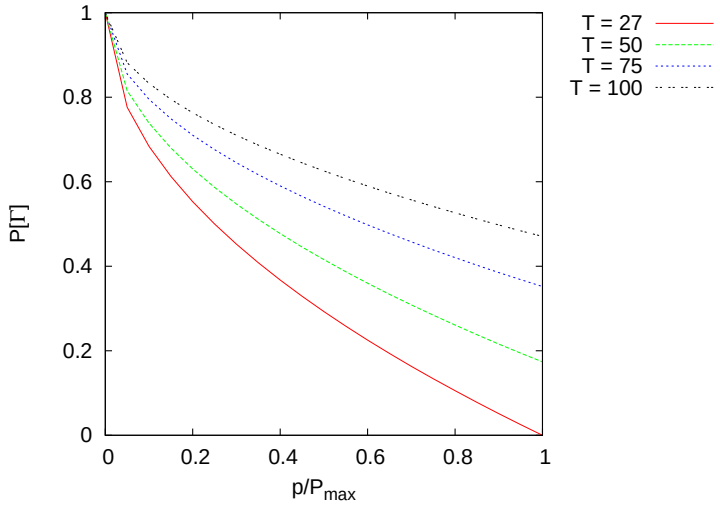

Figura 4.12: *Probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função* $\frac{p_{ef}}{P_{max}}$ *para diversos valores de* T.

da potência total, o que equivale aproximadamente a redução do consumo em 20%, aumentaria a probabilidade de perder novas oportunidades de contato em $8,7\%$. Quando T possui um valor de $100s$, um redução de consumo de 20% aumenta a probabilidade de perder novas oportunidades em aproximadamente $5,5\%$.

Ao comparar as distribuições de $P[\Gamma = 1]$, obtidas analiticamente e por simulação, é possível verificar algumas diferenças, conforme Figuras 4.15 e 4.16. Os resultado obtidos pelas simulações são maiores que os resultados obtidos analiticamente para pequenas potências. A medida que a intensidade do sinal utilizado tende a potência máxima, a probabilidade de perder contatos simulada diminui.

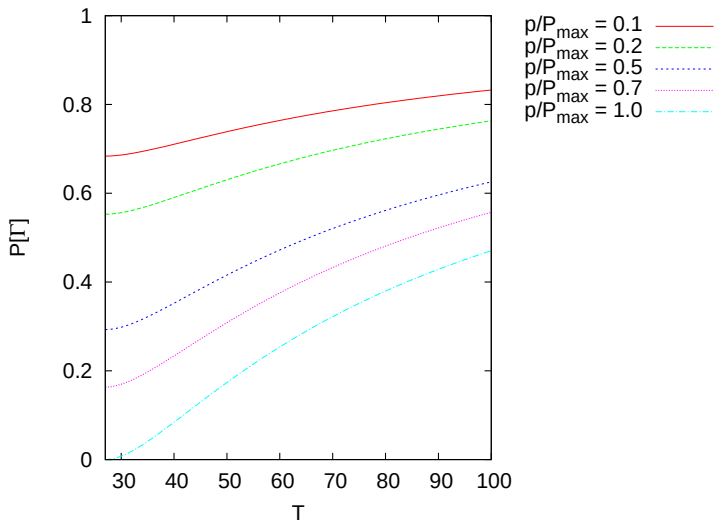

Figura 4.13: *Probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função* T *para diversos valores de* $\frac{p_{ef}}{P_{max}}$.

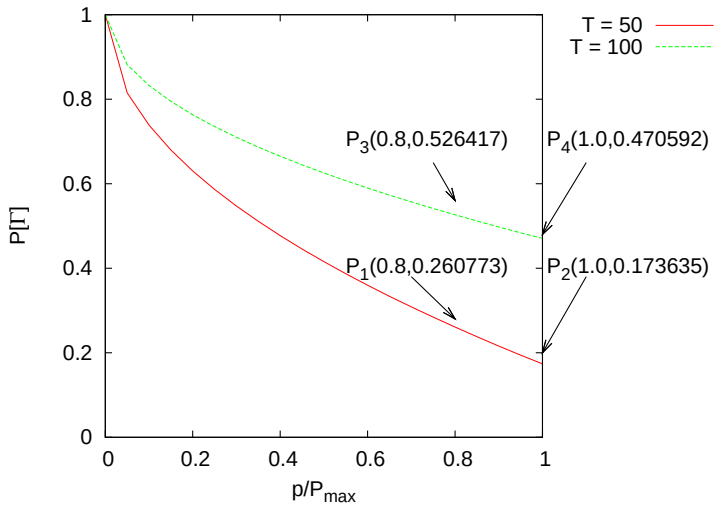

Figura 4.14: *Probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função de* $\frac{p_{ef}}{P_{max}}$. *Quando* $T = 50s$ *é possível observar que uma redução de consumo de* 20% *implica em um aumento da probabilidade de perder novas oportunidades de contato em apenas* $8,7\%$. *Quando* T *possui um valor de* $100s$, *um redução de consumo de* 20% *aumenta a probabilidade de perder novas oportunidades em aproximadamente* $5,5\%$.

4.3 Ajuste Adaptativo da Potência de Transmissão de Mensagens de Descoberta

Nas seções anteriores, verificamos que as suposições e o modelo proposto para o cálculo analítico da probabilidade de perder oportunidades de contato representam de forma aproximada a dinâmica de uma Rede Tolerante a Atrasos e Interrupções,

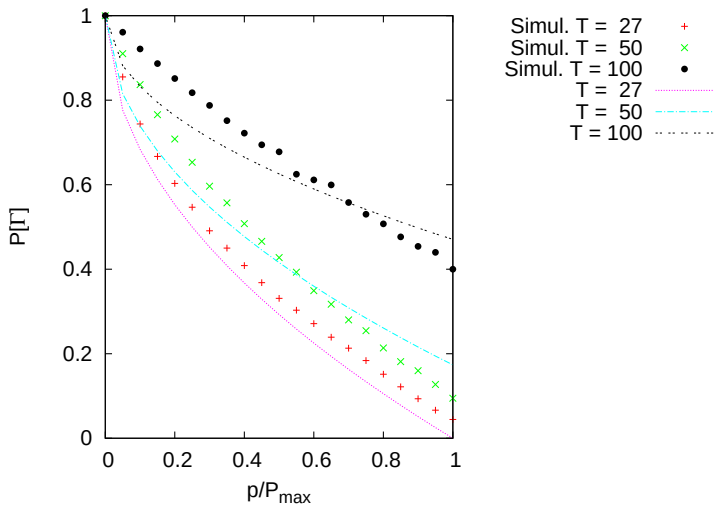

Figura 4.15: *Comparação entre os resultados obtidos analiticamente e por simulação para a probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função* $\frac{p_{ef}}{P_{max}}$ *para diversos valores de* T.

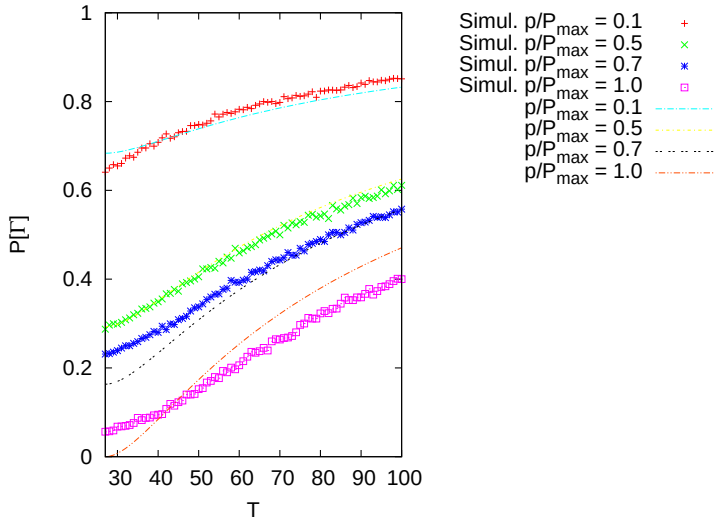

Figura 4.16: *Comparação entre os resultados obtidos analiticamente e por simulação para a probabilidade de perder uma oportunidade de contato* $P[\Gamma = 1]$ *em função* T *para diversos valores de* $\frac{p_{ef}}{P_{max}}$.

tanto para intervalos de contato distribuídos exponencialmente quanto para intervalos distribuídos segundo Pareto.

Também verificamos, visualmente, que existe uma relação de custo/benefício entre o consumo e a perda de novas oportunidades de contato. Além disso, foi demonstrado que é possível reduzir o consumo de energia dos dispositivos sem contudo, aumentar demasiadamente a probabilidade de perder novas oportunidades de contato, tanto para intervalos de contato distribuídos exponencialmente quanto para intervalos distribuídos segundo Pareto.

Nesta seção algumas modificações serão consideradas no modelo proposto de forma a torná-lo mais próximo do mundo real. Também será proposto um algoritmo capaz de ajustar a potência do sinal de transmem relação a duraçãoissão de *probes* a partir das ideias adquiridas com o modelo. Por fim, este algoritmo será avaliado através de simulações e comparado aos algoritmos *AIMD* e *MIAD*, que são algoritmos adaptativos comumente utilizados em protocolos de rede.

4.3.1 Problemas de Otimização Multiobjetivo

Até este momento havíamos feito diversas simplificações com relação ao processo de contato entre dispositivos. Dentre as principais, supomos que o tamanho médio dos intervalos de contatos $\frac{1}{\mu}$ era constante durante todo o período analisado.

Contudo, no mundo real tal suposição não ocorre. Intuitivamente esperamos que o padrão de intervalos de contato durante a noite seja bastante diferente dos contatos que ocorrem durante o dia, por exemplo, no horário de trabalho. Portanto, estamos interessados em investigar como é possível reduzir o consumo de energia quando esse parâmetro varia. na Figura 4.17, é possível visualizar a variação do intervalo de contato médio, percebido por um dos dispositivos da simulação, durante todo o tempo de simulação.

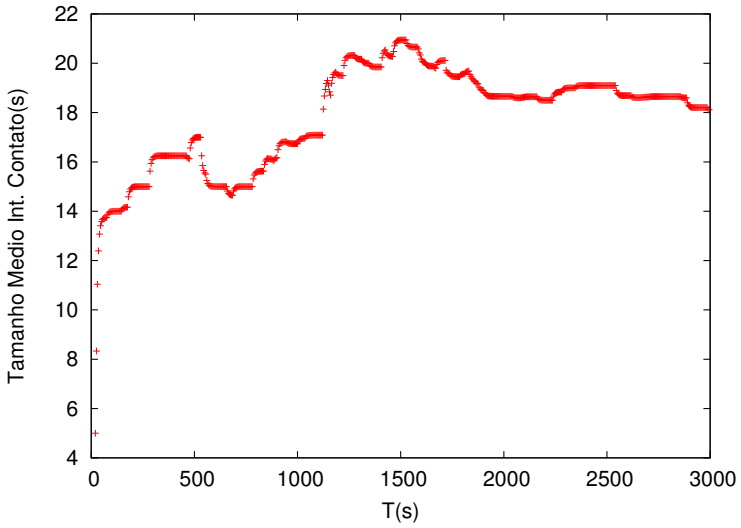

Figura 4.17: *Variação do intervalo de contato médio, percebido por um dos dispositivos da simulação, durante todo o tempo de simulação.*

Supondo que o tamanho médio dos intervalos de contatos, $\frac{1}{\mu}$ possa variar e sob a hipótese de que o intervalo entre *probes* T já tenha sido otimizado, desejamos encontrar a 'melhor' potência para transmitir mensagens de descoberta. Neste caso, escolher a melhor potência de transmissão significa utilizar uma potência que possa ao mesmo tempo reduzir o consumo do dispositivo enquanto mantenha uma baixa

probabilidade de perder novos contatos, ou seja, desejamos encontrar uma potência que otimize estes parâmetros.

Este problema pode ser entendido como um problema de otimização. Contudo, diferentemente dos problemas de otimização escalar, nos quais o objetivo é encontrar uma solução ótima global, neste caso, desejamos minimizar o consumo e ao mesmo tempo a probabilidade de perder novas oportunidades de contato, parâmetros conflitantes. Esta classe de problemas de otimização recebe o nome de *Problemas de Otimização Multiobjetivo*, ou *Otimização Vetorial*.

Os problemas de otimização multiobjetivo são problemas que apresentam uma coleção de objetivos a serem extremizados que são conflitantes entre si, ou seja, a melhoria de algum(uns) objetivo(s) causa(m) consequentemente a deterioração de outro(s).

Para esses problemas não existe uma solução ótima única e sim um conjunto de soluções. As soluções contidas nesse conjunto são ótimas porque não existem outras soluções melhores no espaço de busca, quando são considerados simultaneamente todos os objetivos [69]. Este conjunto de soluções é conhecido como *Soluções Ótimas de Pareto* ou *Fronteira de Pareto*.

Existem vários métodos para encontrar as solução dos problemas de otimização multiobjetivo. Um deles consiste em escalarizar os diversos objetivos, transformando o problema multiobjetivo em um problema de otimização de uma função escalar substituta. Este procedimento pode ser feito otimizando apenas uma das funções e restringindo as demais, ou gerando uma única função objetivo a partir de uma soma ponderada das demais funções objetivo.

Uma vez realizada esta transformação, é possível encontrar a solução ótima através de métodos convencionais como a programação linear e seus métodos de resolução como o algoritmo Simplex.

Um método alternativo consiste na obtenção dos valores pertencentes a fronteira de Pareto através de algoritmos genéticos. Os algoritmos genéticos são baseados no mecanismo de seleção natural. A partir de valores iniciais e procedimentos de cruzamento, mutação e seleção, são gerados a cada iteração do algoritmo, novos resultados cada vez mais próximos da Fronteira de Pareto.

Formalmente temos as seguintes definições:

- *Dominância*
  Um ponto $\mathbf{A}$, domina outro ponto $\mathbf{B}$ se: $\forall i \in 1, 2, \ldots, n | f_i(\mathbf{A}) \leq f_i(\mathbf{B})$ e $\exists j \in 1, 2, \ldots, n | f_j(\mathbf{A}) < f_j(\mathbf{B})$. Onde n é o número de objetivos e $f_i(x)$ são as funções a serem extremizadas.

- *Solução ótima de Pareto*

Um ponto **A** é uma solução não-dominada e pertence à fronteira de Pareto se não existe nenhum outro ponto **B**, tal que **B** domine **A**.

Para exemplificar os conceitos apresentados, considere a Tabela 4.2. Desta, podemos concluir os seguintes pontos:

- A solução $x = 3$ é dominada por $x = 4$: Podemos observar que $f_1(4) \leq f_1(3)$, $f_2(4) \leq f_2(3)$ e $f_2(4) < f_2(3)$;
- A solução $x = 5$ é dominada por $x = 1$: Podemos observar que $f_1(1) \leq f_1(5)$, $f_2(1) \leq f_2(5)$ e tanto $f_1(1) < f_1(5)$ quanto $f_2(1) < f_2(5)$;
- As soluções $x = 1, x = 2, x = 4$, não são dominadas e portanto pertencem a Fronteira de Pareto;

x	$f_1(x)$	$f_2(x)$
1	10	5
2	8	10
3	7	12
4	7	11
5	13	6

Tabela 4.2: *Exemplo de otimização multiobjetivo.*

4.3.2 Otimização Multiobjetivo aplicada a Conservação de Energia

Como visto desejamos encontrar a intensidade do sinal que otimize o consumo e a probabilidade de perder novas oportunidades de contato, formalmente desejamos resolver o seguinte problema:

$$MIN : \begin{cases} P_{\Gamma=1}(x); \\ C(x) = \frac{p_{eft}}{P_{max}}. \end{cases} \tag{4.8}$$

$$s.a. : \begin{cases} 0 \leq P_{\Gamma=1}(x) \leq 1; \\ 0 \leq \frac{x}{P_{Max}} \leq 1. \end{cases} \tag{4.9}$$

Para o problema definido na Equação 4.8, é possível demonstrar que dado T, o conjunto $\mathbf{S} = \{(x, P_{\Gamma=1}(x)) | 0 \leq x \leq 1\}$ é um conjunto de soluções ótimas de Pareto. Para isso basta demonstrar que não existe ponto $(x, P_{\Gamma=1}(x))$ dominante em **S**.

Essa prova pode ser realizada por contradição. Suponha que em **S** exista um ponto dominante $A = (x_a, P_{\Gamma=1}(x_a))$. Portanto deve existir um ponto $B = (x_b, P_{\Gamma=1}(x_b))$ onde:

$$P_{\Gamma=1}(A) \leq P_{\Gamma=1}(B) \tag{4.10}$$

$$C(A) \leq C(B) \tag{4.11}$$

Além disso, uma das seguintes situações deve ocorrer:

$$C(A) < C(B) \tag{4.12}$$

$$P_{\Gamma=1}(A) < P_{\Gamma=1}(B) \tag{4.13}$$

É possível observar que a função $P_{\Gamma=1}(x)$ é monotonicamente descrescente para qualquer distribuição dos tempos de contato e a função $C(x)$ é monotonicamente crescente. Portanto, a Equação 4.10 implica que $A \geq B$. Entretanto, se considerarmos o caso 4.12, esta equação implica que $A < B$. O mesmo pode ser observado para o caso 4.13. Desta forma, temos uma contradição e portanto, não existe um ponto $(A, P_{\Gamma=1}(A)) \in \mathbf{S}$ que seja dominante.

Assim, todos os pontos em **S** são soluções para o problema de otimização proposto na Equação 4.8. A escolha de qual ponto desta curva é 'melhor' está a critério do decisor e de qual objetivo se quer dar prioridade. Por exemplo, pode-se decidir que reduzir a probabilidade de perder novas oportunidades de contato deva ter um importância menor que reduzir o consumo, ou vice-versa.

4.3.3 Definição do Algoritmo de Ajuste Adaptativo do Sinal de Transmissão

Uma vez demonstrado que o conjunto $\mathbf{S} = \{(x, P_{\Gamma=1}(x)) | 0 \leq x \leq 1\}$ é um conjunto de soluções ótimas de Pareto é possível construir um algoritmo capaz de gerar estes pontos através da utilização do modelo proposto. Desta forma temos uma melhoria em relação aos algoritmos genéticos, pois ao gerar os pontos pertencentes a Fronteira de Pareto, não é mais necessário buscar soluções por todo espaço de solução.

Porém, mesmo tento observado que todos os pontos gerados pela aplicação do modelo já fazem parte da solução do problema, ainda é necessário identificar qual intensidade de sinal que possui a melhor relação custo/benefício entre reduzir consumo e aumentar probabilidade de perdas.

Intuitivamente, este ponto é próximo ao 'joelho' da curva $P_{\Gamma=1}$. Isto ocorre

pois, nesta região, uma melhora mesmo que pequena em qualquer um dos objetivos implica em uma degradação considerável no outro objetivo.

Para identificar esta região, propomos uma adaptação do algoritmo evolutivo de otimização multiobjetivo desenvolvido em [5]. Este algoritmo é uma modificação do algoritmo evolutivo NSGA-II (*Non-dominated Sorting Genetic Algorithm - II*)[70] e tem como objetivo encontrar 'joelhos' nas soluções que se encontram na Fronteira de Pareto, mantendo contudo baixa complexidade computacional.

O algoritmo proposto, denominado *Knee*, é executado por cada dispositivo a cada instante t de envio de *probes*. O algoritmo funciona da seguinte forma: todos os dispositivo estimam a duração dos seus intervalos de contato, $\hat{X}_t$, baseado na duração de contatos anteriores, realizando uma média entre os contatos antigos, $\hat{X}_{t-1}$, e as novas amostras ι, como pode ser visto na Equação 4.14.

$$\hat{X}_t = \frac{\iota + \hat{X}_{t-1}}{2} \tag{4.14}$$

Isto pode ser realizada através da manutenção de uma tabela mantida por cada nó e variáveis de controle do tempo. A cada contato identificado, o nó determina a duração deste contato através das variáveis de controle de tempo e ao final do contato atualizam a tabela, calculando a média.

Uma vez determinado $\hat{X}$, são gerados os pontos x_i pertencentes a fronteira de Pareto, através das Equações 4.1 ou 4.5. Como existem infinitos pontos neste intervalo, é preciso identificar um parâmetro de entrada do algoritmo, δ, de forma a gerar os pontos $x_{i+1} = x_i + \delta$.

Este parâmetro determina a granularidade dos pontos gerados. Quanto menor o valor de δ, mais pontos serão gerados e mais próximo estará da potência ideal, a potência de transmissão gerada pelo algoritmo. Porém, maior será o tempo e principalmente o consumo de energia para realizar esse processamento.

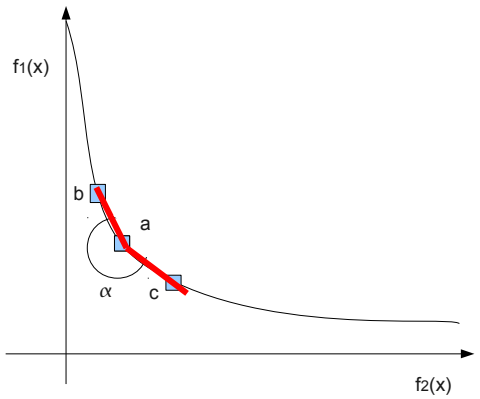

Figura 4.18: *Exemplo do cálculo proposto em [5] para calculo do 'joelho' em uma fronteira de Pareto para as funções $f_1(x)$ e $f_2(x)$.*

Por fim, para cada ponto x_i, calcula-se o ângulo α_i entre este e seus vizinhos x_{i-1} e x_{i+1}, como visto na Figura 4.18. O ponto de maior ângulo relativo deverá estar na região de 'joelho' da curva e é escolhido como tendo a melhor relação custo/benefício. O fluxograma de execução do algoritmo *Knee* pode ser visualizada na Figura 4.19.

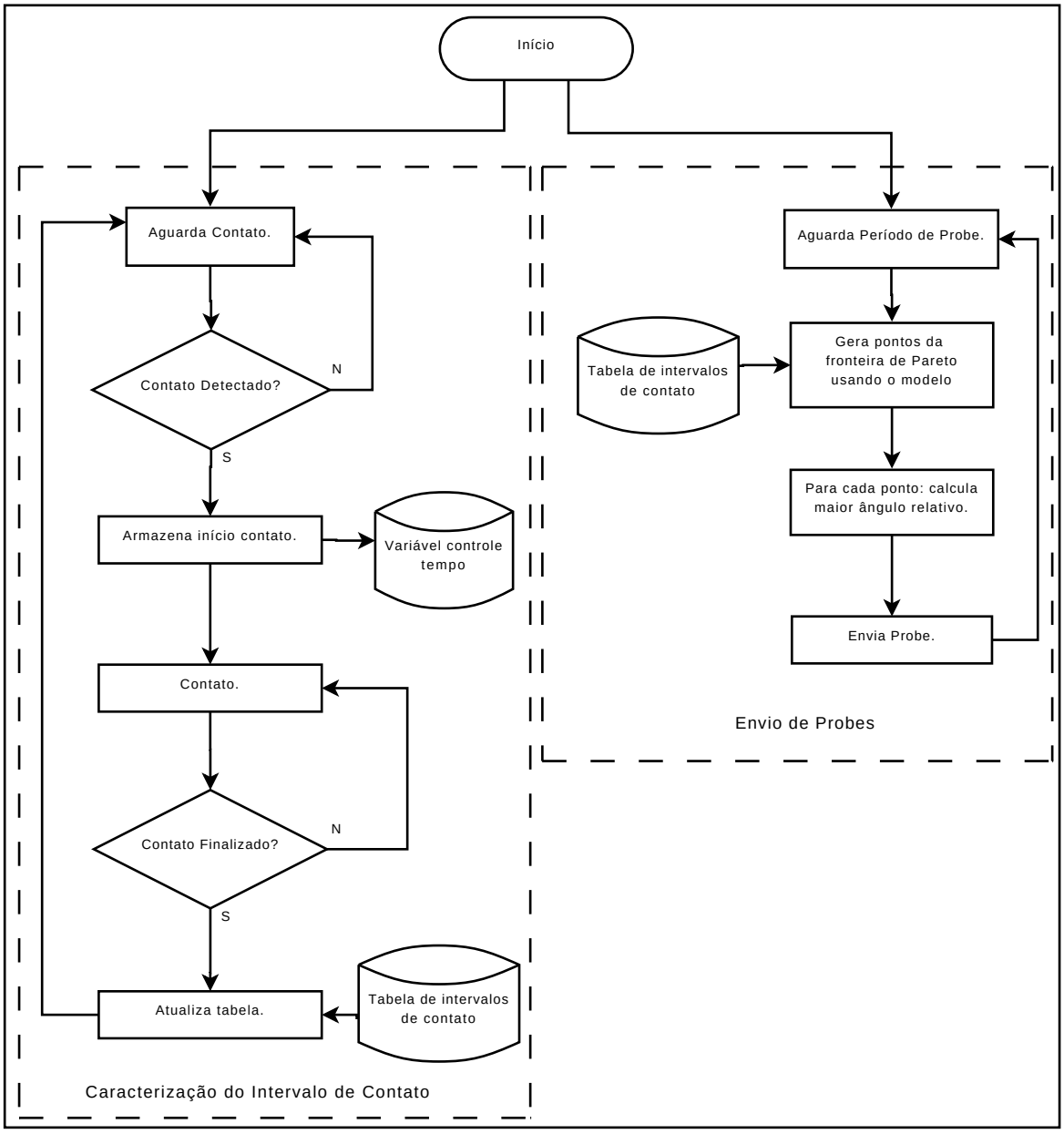

Figura 4.19: *Fluxograma de execução do algoritmo adaptativo* Knee.

Este algoritmo tem a característica de ser adaptativo pois é capaz de estimar os contatos realizados entre os dispositivos e utilizar esta informação e as ideias obtidas com o modelo proposto na Seção 3.2 para calcular a potência de envio de mensagens de descoberta que garanta boa performance. Possui ainda baixa complexidade computacional, da ordem de $O(\frac{1}{\delta})$, onde δ é o parâmetro que determina a granularidade dos pontos gerados. Contudo, possui alta complexidade de memória,

da ordem de $O(|N|)$, onde $|N|$ é a quantidade de dispositivos em estudo. Isto ocorre pois é necessário para o algoritmo armazenar as estimativas de intervalos de contato para cada dispositivo na rede.

Para validar os resultados obtidos, o algoritmo *Knee* proposto foi comparado aos algoritmos *Additive Increase, Multiplicative Decrease*(AIMD) e *Multiplicative Increase, Additive Decrease*(MIAD). Esta comparação com algoritmos genéricos adaptativos foi necessária pois, por ser uma proposta nova, não há na literatura algoritmos semelhantes.

O *AIMD* é um algoritmo adaptativo muito robusto comumente utilizado em protocolos de rede, como o TCP, no qual é responsável por ajustar o tamanho da janela de congestão. Neste trabalho, o algoritmo ajusta a potência de transmissão de mensagens de descoberta baseado na presença ou ausência de novos contatos. Se nenhum contato é detectado, o algoritmo linearmente aumenta a potência de transmissão de *probes* até seu valor máximo, tentando desta forma ampliar seu alcance de transmissão, diminuindo a probabilidade de perder contatos. Caso contrário, ao detectar a presença de um nó vizinho, este poderá ser utilizado para realizar troca de dados, não havendo portanto necessidade de se descobrir novos vizinhos. Por isso, a intensidade do sinal é reduzida rapidamente por um fator de 2 com o objetivo de economizar energia.

O algoritmo *MIAD* é uma variação do algoritmo *AIMD*. Neste, a cada transmissão de um *probe*, se nenhum dispositivo vizinho for detectado, a potência do sinal de transmissão é rapidamente aumentada, por um fator de 2, até a potência máxima de transmissão. Quando um contato é detectado, o algoritmo linearmente diminui a intensidade do sinal das mensagens de descoberta para reduzir o consumo de energia.

4.3.4 Simulação - Metodologia e Resultados

Para efetuar a comparação entre os algoritmos, estes foram implementados no simulador orientado a eventos discretos descrito na Seção 4.1.1.

As simulações foram executadas em duas etapas distintas. Na primeira etapa consideramos intervalos de contato distribuídos exponencialmente. Na segunda etapa, realizamos uma modificação no tamanho da área simulada de forma a obter tempos de contato distribuidos segundo Pareto.

Para cada etapa, novas rodadas de simulação foram realizadas. Exceto pelo tamanho da área simulada, os parâmetros utilizados foram os mesmos da Tabela 4.1. Também foi permitido que cada dispositivo variasse sua potência de transmissão efetiva de *probes*, de acordo com o algoritmo utilizado. Além disso supomos que $\delta = 0.01$ e que portanto o ponto gerado pelo algoritmo estará apenas a essa distância,

ou menor, do ponto ideal.

As métricas preteridas desta vez são: a probabilidade de perder novas oportunidades de contato e o consumo de energia. Estas métricas foram obtidas para cada um dos três algoritmos descritos anteriormente e para o caso de maior consumo, quando os nós mantem a potência efetiva de transmissão constante e igual a potência máxima.

Para obtermos a probabilidade de perder novas oportunidades de contato, seguimos a mesma metodologia descrita na Seção 4.1.1. Para obtermos o consumo, foi realizado o seguinte método: a cada rodada de simulação, e para cada dispositivo da rede, foram contabilizados os consumos das mensagens de descoberta de acordo com a Equação 4.15:

$$C(p) = p_{ef} \tag{4.15}$$

Onde $0 \leq p_{ef} \leq P_{max}$ é a potência efetiva de transmissão. O consumo total de cada dispositivo foi calculado como a soma dos consumos individuais de cada *probe*. Ao final de cada rodada de simulação, obtivemos uma média do consumo de todos os dispositivos. Posteriormente, ao final de todas as rodadas, uma para cada *trace* de movimentação, obtivemos o consumo médio observado. Estas métricas foram obtidas variando-se o intervalo entre *probes* T.

Os resultados obtidos na primeira etapa, quando o intervalo de contato é distribuido exponencialmente, podem ser visualizado a seguir. Na Figura 4.20, podemos observar que o caso extremo e de menor probabilidade de perda ocorreu quando foi mantido constante a potência máxima de transmissão, durante todo o período da simulação.

Contudo, este é também o caso de maior consumo. O algoritmo proposto obteve menor probabilidade de perdas, quando comparado aos algoritmos *AIMD* e *MIAD*. Sendo a diferença entre estes maior para pequenos valores de T, casos em que a curva de $P[\Gamma = 1]$ apresenta um 'joelho' acentuado.

Na Figura 4.21 podemos observar que todos os algoritmos são capazes de reduzir o consumo de energia. Essa redução, para pequenos valores de T, chega a até 85% do consumo máximo para o algoritmo *AIMD* e a até 45% para os algoritmos *MIAD* e *Knee*.

Uma redução maior obtida pelo algoritmo *AIMD* em relação aos demais algoritmos era esperada, pois este algoritmo eleva a potência de forma linear e o reduz exponencialmente. Contudo, por reduzir drasticamente a potência de transmissão após a descoberta de um contato, sua probabilidade de perder novas oportunidades aumenta.

Pelas Figuras 4.20 e 4.21 é difícil analisar qual algoritmo exerce melhor sua

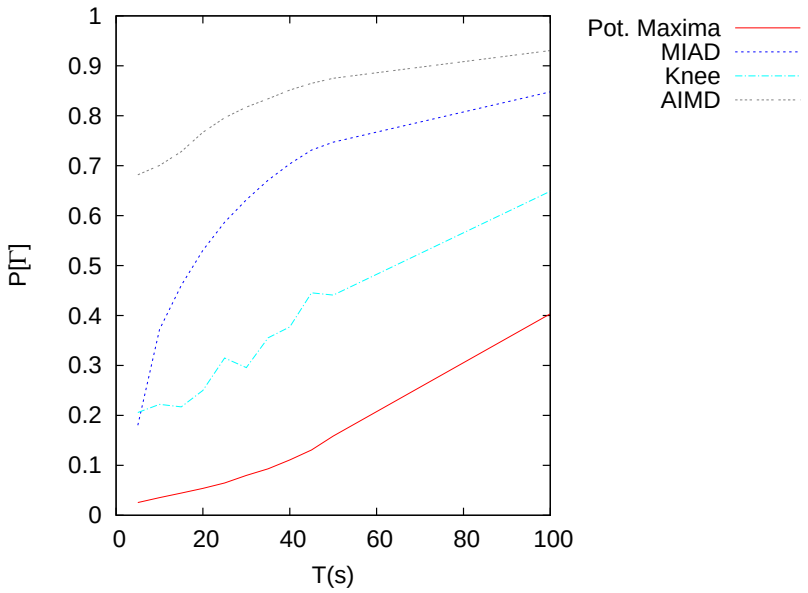

Figura 4.20: *Probabilidade média de perder oportunidades de contato em função de T para os algoritmos analisados e para intervalos de contato distribuídos exponencialmente.*

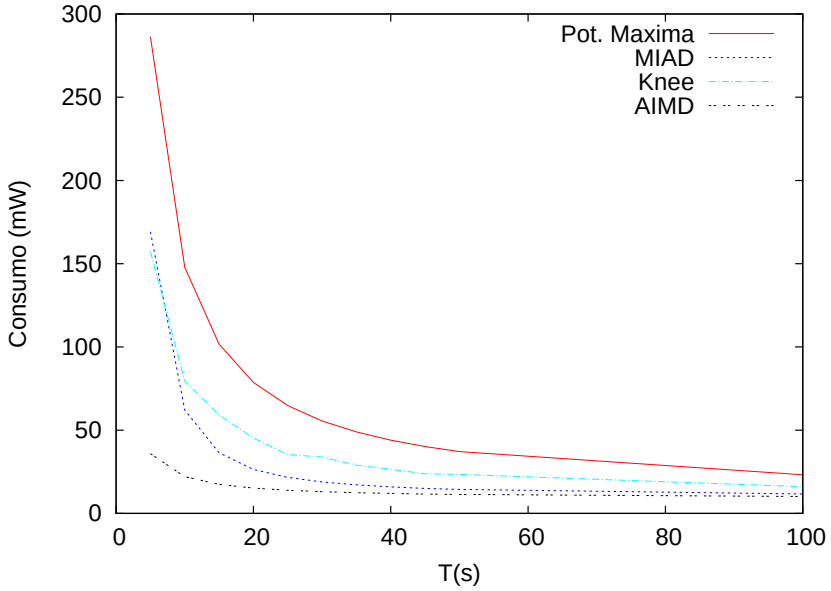

Figura 4.21: *Consumo médio de energia em função de T para os algoritmos analisados e para intervalos de contato distribuídos exponencialmente.*

função, uma vez que o algoritmo *AIMD* é o que possui maior redução do consumo de energia, enquanto o algoritmo *Knee* é o que possui o menor acréscimo na probabilidade de perder oportunidades de contato.

Como forma de compará-los, foi executado o seguinte procedimento: para cada algoritmo, calculamos a redução do consumo e o consequente acréscimo da probabilidade em relação ao caso extremo, em que é empregada a potência máxima durante todo período de análise. Posteriormente encontramos a relação entre estas difer-

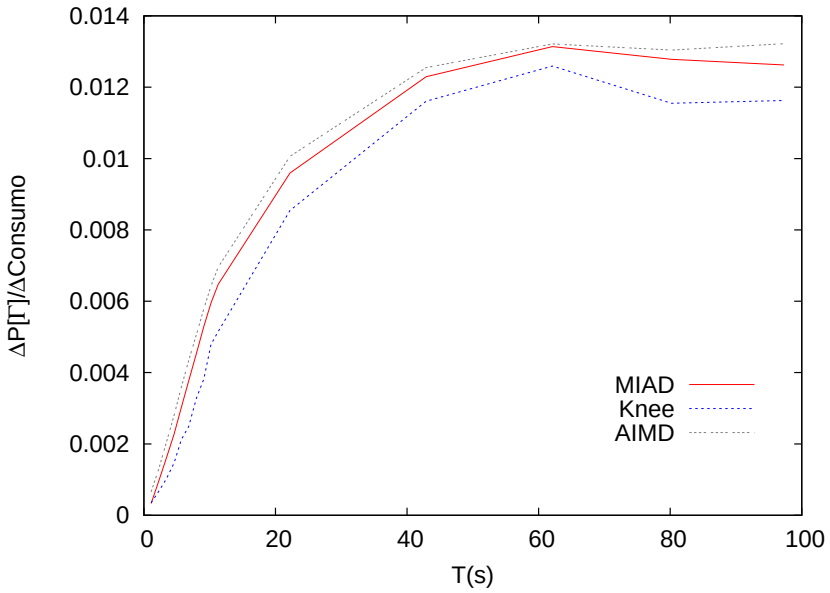

Figura 4.22: *Comparativo do acréscimo de* $P[\Gamma = 1]$ *por redução do consumo para intervalos de contato distribuídos exponencialmente.*

enças, o acréscimo da probabilidade por redução consumo, que pode ser visualizada na Figura 4.22.

Dentre os algoritmos avaliados, o algoritmo proposto *Knee*, é o que possui o menor acréscimo da probabilidade por redução do consumo, ou seja, para um mesmo acréscimo da probabilidade de perda este implica na maior redução de consumo. E para uma dada redução do consumo, o algoritmo implica no menor acréscimo da probabilidade de perda.

Portanto, é possível concluir que quando a duração do intervalo de contato é distribuida exponencialmente, o algoritmo proposto é uma boa escolha para a otimizar a relação entre consumo e probabilidade de perder novas oportunidades de contato.

Os resultados obtidos durante a segunda etapa, na qual o intervalo de contato entre dispositivos é distribuido segundo Pareto, podem ser visualizados a seguir.

Pela Figura 4.23 é possível verificar que o algoritmo *Knee* é o que apresenta maior probabilidade de perder novas oportunidades de contato. Tal fato ocorre, pois a distribuição de $P[\Gamma = 1]$ para intervalos de contato distribuídos segundo Pareto, não apresenta o formato de 'joelho', como visto na Figura 4.12. Além disso, os valores de $P[\Gamma = 1]$ sofrem um acréscimo para pequenos valores de $\frac{p_{eft}}{P_{max}}$ e o algoritmo *Knee* acaba por identificar esses pontos como o joelho da curva, implicando em probabilidades maiores de perda.

Ainda analisando a Figura 4.23, verificamos que o algoritmo que possui menor probabilidade de perda é o *MIAD*.

Novamente verificamos que todos os algoritmos são capazes de reduzir o consumo de energia como visto na Figura 4.24. Essa redução, para pequenos valores de T,

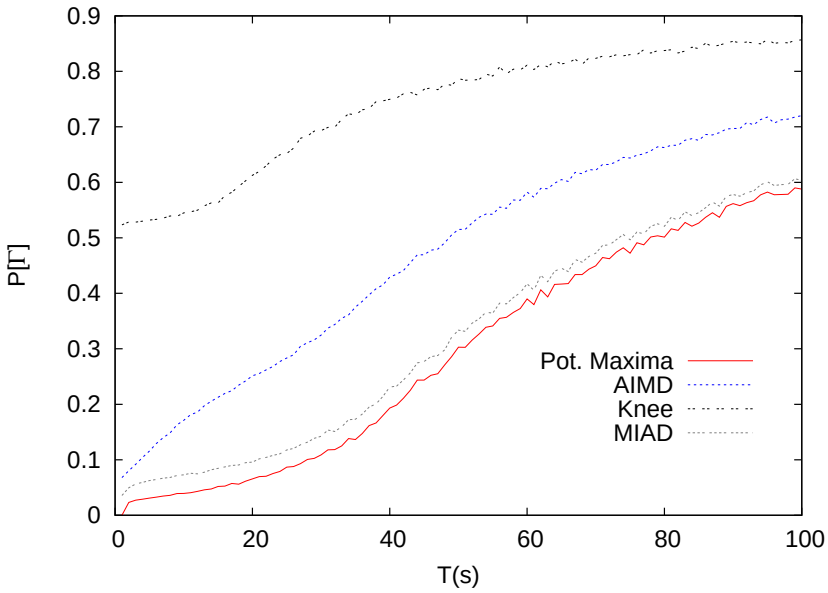

Figura 4.23: *Probabilidade média de perder oportunidades de contato em função de T para os algoritmos analisados e para intervalo de contato distribuído segundo Pareto.*

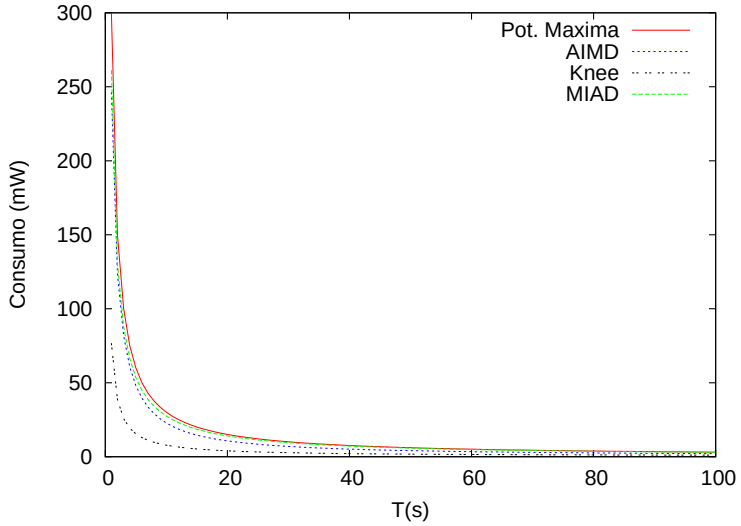

Figura 4.24: *Consumo médio de energia em função de T para os algoritmos analisados e para intervalos de contato distribuídos segundo Pareto.*

chega até a aproximadamente 17% para os algoritmos *Knee* e *AIMD*, e até 75% para o algoritmo *MIAD*.

Realizando a análise da relação, acréscimo da probabilidade por redução consumo, descobrimos que o algoritmo *MIAD* é o que minimiza essa métrica para intervalos entre mensagens de descoberta de até aproximadamente 50s, a partir desse valor, os três algoritmos possuem performance semelhante, como pode ser visto na Figura 4.25.

Portanto, concluímos que quando a duração do intervalo de contato é distribuida

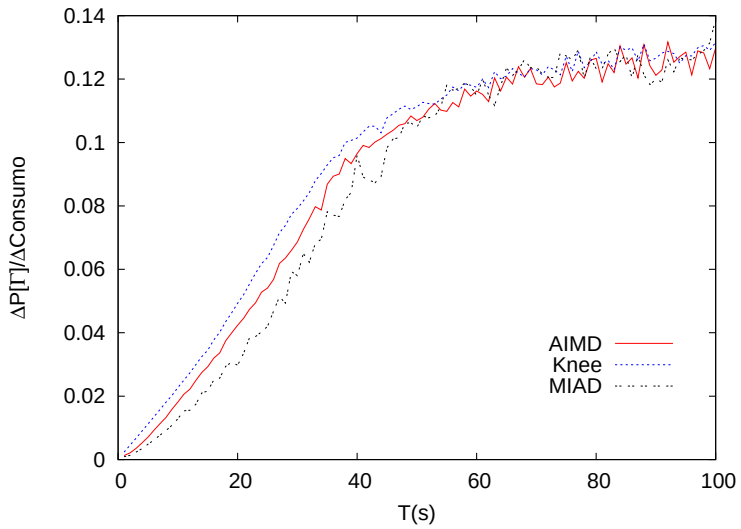

Figura 4.25: *Comparativo do acréscimo de* $P[\Gamma = 1]$ *por redução do consumo para intervalos de contato distribuídos segundo Pareto.*

segundo Pareto, o algoritmo *MIAD* é a melhor escolha para otimizar a relação entre consumo e probabilidade de perder novas oportunidades de contato. O algoritmo proposto, embora seja capaz de reduzir consideravelmente o consumo, eleva em demasia a probabilidade de perder novas oportunidades de contato.

Capítulo 5

Conclusões e Trabalhos Futuros

Este capítulo apresenta as principais considerações e conclusões deste trabalho. Também aborda tópicos e sugestões relevantes para pesquisas futuras.

5.1 Conclusões

Atualmente observamos os avanços em diversos tipos de redes em diferentes cenários. Redes de sensores tem sido implementadas para obtenção de dados biológicos de diversos animais, redes subaquáticas tem sido utilizadas para monitorar aspectos como temperatura e pH relativos aos mares e rios e as redes veiculares estão cada vez mais próximas de uma implementação definitiva.

Portanto as Redes Tolerantes a Atrasos e Interrupções já são uma realidade e as aplicações dessas redes devem continuar crescendo devido às vantagens que apresentam. Contudo, ainda são vários os desafios encontrados.

O problema de conservação de energia é um dos principais problemas. Desta forma, entender como ocorre o consumo em tais redes é de fundamental importância.

Em [42] foi verificado que o mecanismo de descoberta de dispositivos vizinhos, feito através da transmissão de mensagens de descoberta (*probes*), é um dos principais processos consumidores de energia. Portanto modelar esse processo de forma correta é essencial para entender a relação custo/benefício entre o consumo de energia e as oportunidades de contato.

O principal objetivo deste trabalho foi de analisar o processo de descoberta de dispositivos vizinhos e avaliá-lo com relação ao consumo, buscando uma forma de minimizá-lo, sem contudo, degradar o desempenho dos dispositivos constituintes da rede.

Para alcançar tal objetivo, foi definida uma nova métrica, a *Probabilidade de Perder uma Oportunidade de Contato*, calculada através de um modelo analítico proposto. De forma a validar este modelo e testar sua acurácia, diversas simulações foram realizadas. Estas demonstraram que o modelo é capaz de representar bem um cenário DTN quando o intervalo de contatos é distribuído exponencialmente ou segundo distribuições de Pareto.

Ao avaliar o modelo, verificamos a existência de uma relação custo/benefício entre consumo de energia e probabilidade de perder novas oportunidades de contatos. Ou seja, uma melhora em um dos objetivos leva a uma degradação do outro. Posteriormente tal relação foi demonstrada analiticamente.

Verificamos também que a probabilidade de perder contatos apresenta um 'joelho' acentuado quando a duração do intervalo de contato é distribuída exponencialmente e o número de contatos em um intervalo de *probes* é pequeno. Tal característica nos permitiu reduzir o consumo de energia sem contudo aumentar demasiadamente a probabilidade de perder contatos.

Esta característica não foi encontrada quando os intervalos de contato eram distribuídos segundo Pareto. Contudo, ainda sim verificamos que é possível reduzir o consumo de energia sem aumentar linearmente a probabilidade de perder novos

contatos.

Definimos o problema de minimizar o consumo e ao mesmo tempo minimizar a probabilidade de perda como um problema de otimização multiobjetivo. Para resolvê-lo, propomos um algoritmo baseado na técnica de algoritmos evolutivos.

O algoritmo proposto, *Knee*, se adapta ao ambiente do dispositívo, identificando a distribuição dos intervalos de contato deste nó e por fim, busca pelo ponto mais próximo ao 'joelho' da curva.

Este ponto possui a melhor relação custo/benefício entre reduzir consumo e aumentar probabilidade de perdas, isto porque, nesta região, uma melhora mesmo que pequena em qualquer um dos objetivos implica em uma degradação considerável no outro objetivo. Portanto, a adaptabilidade do algoritmo está relacionada aos contatos realizados entre os dispositivos e como o algoritmo utiliza esta informação para calcular o a potência de envio de mensagens de descoberta que possui melhor relação custo/benefício.

O algoritmo *Knee* foi implementado em um simulador e comparados aos algoritmos adaptativos *AIMD* e *MIAD*. Nas simulações realizadas, verificamos que dentre os algoritmos avaliados, o algoritmo proposto é o que melhor se adapta a variação do tamanho do intervalo de contato quando este é distribuído exponencialmente, gerando o menor acréscimo da probabilidade por redução consumo.

Esta redução chegou a até aproximadamente 50% consumo para um acréscimo de aproximadamente 18% da probabilidade de perder novas oportunidades de contato. Tal redução, para um ambiente como o descrito em [6], em que o processo de descoberta de dispositivos vizinhos é responsável por $36,76\%$ do consumo total, implica em uma sobre vida de aproximadamente $22,51\%$ do tempo de vida total.

Quando os intervalos de contato são distribuídos segundo Pareto, o algoritmo proposto não apresenta boa performance, elevando consideravelmente a probabilidade de perder novas oportunidades de contato. Neste caso, o algoritmo que apresentou melhor performance foi o *MIAD*.

5.2 Considerações e Trabalhos Futuros

Embora diversas hipóteses adotadas em sua formulação não estejam presentes no mundo real, o modelo proposto neste trabalho representa de forma aproximada a dinâmica de uma Rede Tolerante a Atrasos e Interrupções. Uma continuação do trabalho aqui proposto seria introduzir no modelo hipóteses mais realistas.

Neste trabalho supomos que a progapação do sinal de transmisssão correspondia ao modelo de propagação de Friss. Porém, este modelo nem sempre é capaz de representar um cenário real, como observado em [71], sendo principalmente utilizado para fins acadêmicos. Um modelo mais realista, muito utilizado em ambientes in-

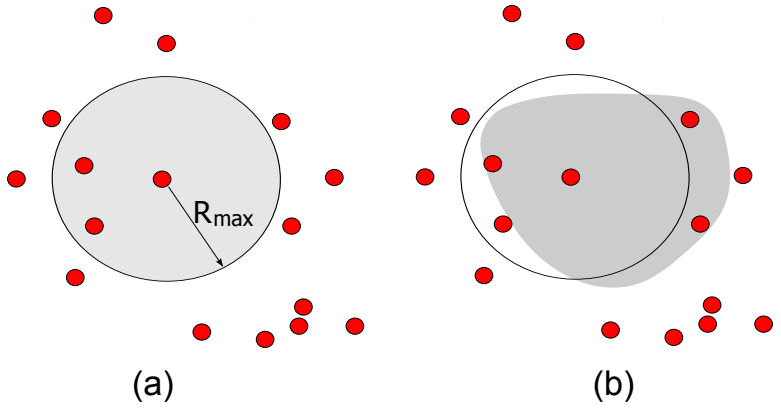

Figura 5.1: *(a) Modelo de Friss (b)Modelo 'Log Distance Path Loss'*

ternos(*indoor*) pode ser encontrado em [72, 73]. Neste modelo,*Log Distance Path Loss*, a atenuação do sinal não é devida apenas ao decaimento do espaço-livre, mas também, ao desvanecimento lento(*slow fading*) ou sombreamento(*shadowing*).

Como trabalho futuro, sugerimos a incorporação deste fenômeno de sombreamento ao modelo de propagação. Isto pode ser feito, ao considerar a atenuação como uma variável composta por duas componentes: (i)uma determinística, $L_d(d)$, calculada pela atenuação de espaço-livre e dada pelo modelo de Friis, e (ii) uma estocástica X_σ, obtida a partir de uma variável aleatória de distribuição log-normal (média zero e desvio padrão σ) quando medida em Watts(W) ou distribuição normal quando medida em decibéis (dB).

A adição da componente estocástica no cálculo da atenuação tem por objetivo, representar o realismo ao incluir sombras e deformações na área de transmissão A_i do dispositivo i, como pode ser visto na Figura 5.1.

Outro ponto a ser observado é que também supomos neste trabalho que os intervalos entre mensagens de descoberta já haviam sido otimizados segundo alguma técnica, como por exemplo, a proposta em [42]. Com esta suposição, foi possível focar a atenção apenas na redução da intesidade dos sinais de transmissão de *probes*. Entretando, o modelo proposto permite tratar os intervalos entre estas mensagens como variáveis do problema, permitindo portanto, incluí-los no problema de otimização multiobjetivo. Assim, ao reformular o problema proposto na Equação 4.8, poderíamos obter como soluções tanto o intervalo entre *probes* otimizado para o cenário quanto a potência ideal para transmitir estas mensagens de descoberta.

Em relação ao algoritmo proposto é importante observar que durante as avaliações, supomos que este possuía informações sobre a distribuição dos intervalos de contato. Desta forma, o algoritmo utilizava a Equação 4.1 para intervalos exponenciais e a Equação 4.5 para intervalos distribuídos segundo Pareto. Uma proposta interessante, seria modificar o algoritmo de forma que este seja capaz de inferir a distribuição dos intervalos de contato durante sua execução.

Ainda em relação ao algoritmo, não foi realizado nenhum estudo sobre capacidade de processamento necessária, capacidade de memória e consumo de energia. Tais aspectos são relevantes, pois como visto na Seção 2.2, tais recursos são escassos em cenários DTN.

Outra proposta seria implementar o algoritmo para redução de energia *Knee* em dispositivos reais de comunicação como o Tmote [67]. Esse dispositivo possui um subsistema de comunicação baseado no CC2420, o que permite variar a intensidade do sinal de transmissão. Também é capaz de executar programas escritos e compilados na linguagem *nesC*, sobre o sistema *TinyOS*, permitindo a implementação do algoritmo nessa linguagem.

Referências Bibliográficas

[1] "ISC Domain Survey: Number of Internet Hosts". . Disponível em: <https://www.isc.org/solutions/survey>.

[2] LIU, T., SADLER, C., ZHANG, P., etal. "Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet". In: *Proceedings of the Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Boston, MA, June 2004. USENIX Association. Disponível em: <http://portal.acm.org/citation.cfm?id=990064.990095>.

[3] " CC1000 Datasheet - Single Chip Very Low Power RF Transceiver". . Disponível em: <http://focus.ti.com/docs/prod/folders/print/cc1000.html>.

[4] ALEXANDRE M. DA SILVA, BRUNO A. A. NUNES, L. F. M. D. M. "Uma Avaliação dos Efeitos das Regras de Borda e dos Modelos de Mobilidade no Comportamento dos Nós em Redes Ad Hoc". In: *VI Workshop de Comunicação Sem Fio e Computação Móvel - WCSF*, Maio 2004.

[5] BRANKE, J., DEB, K., DIEROLF, H., etal. "Finding knees in multi-objective optimization". In: *In the Eighth Conference on Parallel Problem Solving from Nature (PPSN VIII). Lecture Notes in Computer Science*, pp. 722–731. Springer-Verlag, October 2004.

[6] CANO, J.-C., CANO, J.-M., GONZALEZ, E., etal. "Power Characterization of a Bluetooth-based Wireless Node for Ubiquitous Computing", *Wireless and Mobile Communications, International Conference on*, v. 0, pp. 13, December 2006. doi: http://doi.ieeecomputersociety.org/10.1109/ICWMC.2006.75.

[7] CARLOS A. V. CAMPOS, RAFAEL L. BEZERRA, L. F. M. D. M. "Uma Proposta de Técnica para o Ajuste de Modelos de Mobilidade em Redes Ad Hoc e Questionamentos Sobre a Adequação dos Parâmetros Envolvi-

dos com Base em Dados Reais". In: *27 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC - Refice/PE*, Maio 2009.

[8] CARLOS A. V. CAMPOS, RAFAEL L. BEZERRA, T. S. A. L. F. M. D. M. "An Analysis of Human Mobility using Real Traces". In: *EEE Wireless Communications and Networking Conference - WCNC*, Agosto 2009.

[9] " CC2420 Datasheet - 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver". . Disponível em: <http://focus.ti.com/docs/prod/folders/print/cc2420.html>.

[10] BUSH, R., MEYER, D. "RFC3439 - Some Internet Architectural Guidelines and Philosophy". , 2002. Disponível em: <ftp://ftp.internic.net/rfc/rfc3439.txt>.

[11] TOH, C.-K. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Upper Saddle River, NJ, USA, PTR Prentice-Hall, November 2002. ISBN: 0-13-007817-4.

[12] AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., etal. "A survey on sensor networks", *Communications Magazine, IEEE*, v. 40, n. 8, pp. 102–114, July 2002. doi: 10.1109/MCOM.2002.1024422. Disponível em: <http://dx.doi.org/10.1109/MCOM.2002.1024422>.

[13] AKYILDIZ, I. F., POMPILI, D., MELODIA, T. "Underwater acoustic sensor networks: research challenges", *Ad Hoc Networks*, v. 3, n. 3, pp. 257–279, May 2005. doi: 10.1016/j.adhoc.2005.01.004. Disponível em: <http://dx.doi.org/10.1016/j.adhoc.2005.01.004>.

[14] LUO, J., PIERRE HUBAUX, J. *A survey of inter-vehicle communication*. Relatório técnico, School of Computer and Communication Sciences, EPFL, November 2004.

[15] CERF, V., BURLEIGH, S., HOOKE, A., etal. "Delay-Tolerant Networking Architecture". . RFC 4838 (Informational), April 2007. Disponível em: <http://www.ietf.org/rfc/rfc4838.txt>.

[16] BURLEIGH, S., HOOKE, A., TORGERSON, L., etal. "Delay-tolerant networking: an approach to interplanetary Internet", *IEEE Communications Magazine*, v. 41, n. 6, pp. 128–136, July 2003. doi: 10.1109/MCOM.2003.1204759. Disponível em: <http://dx.doi.org/10.1109/MCOM.2003.1204759>.

[17] FALL, K. "A delay-tolerant network architecture for challenged internets". In: *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 27–34, New York, NY, USA, August 2003. ACM. ISBN: 1-58113-735-4. doi: http://doi.acm.org/10.1145/863955.863960.

[18] "Delay Tolerant Networking Research Group". . Disponível em: <http://www.dtnrg.org/wiki>.

[19] SMALL, T., HAAS, Z. J. "The Shared Wireless Infostation Model: A New Ad Hoc Networking Paradigm (Or Where There is a Whale, There is a Way)". In: *MobiHoc 2003: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pp. 233–244, New York, NY, USA, August 2003. ACM Press. ISBN: 1581136846. doi: 10.1145/778415.778443. Disponível em: <http://dx.doi.org/10.1145/778415.778443>.

[20] PENTLAND, A., FLETCHER, R., HASSON, A. "DakNet: rethinking connectivity in developing nations", *Computer*, v. 37, n. 1, pp. 78–83, 2004. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1260729>.

[21] DEMMER, M., DU, B., BREWER, E. "TierStore: a distributed filesystem for challenged networks in developing regions". In: *FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pp. 1–14, Berkeley, CA, USA, August 2008. USENIX Association.

[22] ZHAO, W., AMMAR, M., ZEGURA, E. "A message ferrying approach for data delivery in sparse mobile ad hoc networks". In: *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pp. 187–198, New York, NY, USA, July 2004. ACM Press. ISBN: 1581138490. doi: 10.1145/989459.989483. Disponível em: <http://dx.doi.org/10.1145/989459.989483>.

[23] JUN, H., AMMAR, M. H., CORNER, M. D., etal. "Hierarchical power management in disruption tolerant networks with traffic-aware optimization". In: *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pp. 245–252, New York, NY, USA, February 2006. ACM. ISBN: 1-59593-572-X. doi: http://doi.acm.org/10.1145/1162654.1162662.

[24] BURGESS, J., GALLAGHER, B., JENSEN, D., etal. "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks". In: *INFOCOM 2006. 25th*

IEEE International Conference on Computer Communications. Proceedings, pp. 1–11, July 2006. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4146881>.

[25] JAIN, S., FALL, K., PATRA, R. "Routing in a delay tolerant network". In: *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, v. 34, pp. 145–158. ACM Press, October 2004. ISBN: 1581138628. doi: 10.1145/1015467.1015484. Disponível em: <http://dx.doi.org/10.1145/1015467.1015484>.

[26] LEBRUN, J., CHUAH, C.-N., GHOSAL, D., etal. "Knowledge-Based Opportunistic Forwarding in Vehicular Wireless Ad Hoc Networks". In: *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, v. 4, pp. 2289–2293, July 2005. doi: 10.1109/VETECS.2005.1543743. Disponível em: <http://dx.doi.org/10.1109/VETECS.2005.1543743>.

[27] SHAH, R. C., ROY, S., JAIN, S., etal. "Data MULEs: modeling a three-tier architecture for sparse sensor networks". In: *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pp. 30–41, June 2003. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1203354>.

[28] ZHAO, W., AMMAR, M., ZEGURA, E. "A message ferrying approach for data delivery in sparse mobile ad hoc networks". In: *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pp. 187–198, New York, NY, USA, June 2004. ACM. ISBN: 1-58113-849-0. doi: http://doi.acm.org/10.1145/989459.989483.

[29] JONES, E. P. C., LI, L., SCHMIDTKE, J. K. "Practical Routing in Delay-Tolerant Networks", *IEEE Transactions on Mobile Computing*, v. 6, n. 8, pp. 943–959, 2007. ISSN: 1536-1233. doi: http://dx.doi.org/10.1109/TMC.2007.1016. Member-Ward,, Paul A. S.

[30] VAHDAT, A., BECKER, D. "Epidemic Routing for Partially Connected Ad Hoc Networks". , September 2000. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.6151>.

[31] SPYROPOULOS, T., PSOUNIS, K., RAGHAVENDRA, C. S. "Spray and wait: an efficient routing scheme for intermittently connected mobile networks". In: *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on*

Delay-tolerant networking, pp. 252–259, New York, NY, USA, 2005. ACM. ISBN: 1-59593-026-4. doi: http://doi.acm.org/10.1145/1080139.1080143.

[32] GROSSGLAUSER, M., TSE, D. N. C. "Mobility increases the capacity of ad hoc wireless networks", *IEEE/ACM Trans. Netw.*, v. 10, n. 4, pp. 477–486, August 2002. ISSN: 1063-6692. doi: http://dx.doi.org/10.1109/TNET.2002.801403.

[33] CHAINTREAU, A., HUI, P., CROWCROFT, J., etal. "Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding", *University of Cambridge Computer Laboratory, Tech. Rep. UCAM-CL-TR-617, Feb*, February 2005.

[34] CHAINTREAU, A., HUI, P., CROWCROFT, J., etal. "Impact of Human Mobility on Opportunistic Forwarding Algorithms", *IEEE Transactions on Mobile Computing*, v. 6, n. 6, pp. 606–620, June 2007. ISSN: 1536-1233. doi: 10.1109/TMC.2007.1060. Disponível em: <`http://dx.doi.org/10.1109/TMC.2007.1060`>.

[35] MUSOLESI, M., MASCOLO, C. "Designing Mobility Models based on Social Network Theory", *ACM SIGMOBILE Mobile Computing and Communication Review*, July 2007.

[36] KARAGIANNIS, T., LE BOUDEC, J.-Y., VOJNOVIC, M. "Power law and exponential decay of inter contact times between mobile devices". In: *The Thirteenth Annual International Conference on Mobile Computing and Networking*, November 2007.

[37] CAI, H., EUN, D. Y. "Crossing over the bounded domain: from exponential to power-law inter-meeting time in MANET". In: *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pp. 159–170, New York, NY, USA, September 2007. ACM. ISBN: 978-1-59593-681-3. doi: http://doi.acm.org/10.1145/1287853.1287873.

[38] CAI, H., EUN, D. Y. "Aging rules: what does the past tell about the future in mobile ad-hoc networks?" In: *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pp. 115–124, New York, NY, USA, December 2009. ACM. ISBN: 978-1-60558-531-4. doi: http://doi.acm.org/10.1145/1530748.1530764.

[39] EFRON, B. "Logistic Regression, Survival Analysis, and the Kaplan-Meier Curve", *Journal of the American Statistical Association*, v. 83, n. 402,

pp. 414–425, 1988. ISSN: 01621459. doi: 10.2307/2288857. Disponível em: <http://dx.doi.org/10.2307/2288857>.

[40] ERRAMILLI, V., CHAINTREAU, A., CROVELLA, M., etal. "Diversity of forwarding paths in pocket switched networks". In: *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 161–174, New York, NY, USA, September 2007. ACM. ISBN: 978-1-59593-908-1. doi: http://doi.acm.org/10.1145/1298306.1298330.

[41] FEENEY, L. M., NILSSON, M. "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment". In: *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, v. 3, pp. 1548–1557 vol.3, August 2001. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=916651>.

[42] WANG, W., SRINIVASAN, V., MOTANI, M. "Adaptive contact probing mechanisms for delay tolerant applications". In: *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pp. 230–241, New York, NY, USA, February 2007. ACM. ISBN: 978-1-59593-681-3. doi: http://doi.acm.org/10.1145/1287853.1287882.

[43] SCHURGERS, C., KULKARNI, G., SRIVASTAVA, M. B. "Energy-aware wireless sensor networks", *IEEE Signal Processing*, v. 19, pp. 40–50, March 2002.

[44] WANG, Y., JAIN, S., MARTONOSI, M., etal. "Erasure-coding based routing for opportunistic networks". In: *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 229–236, New York, NY, USA, September 2005. ACM. ISBN: 1-59593-026-4. doi: http://doi.acm.org/10.1145/1080139.1080140.

[45] ANASTASI, G., CONTI, M., FRANCESCO, M., etal. "Energy conservation in wireless sensor networks: A survey", *Ad Hoc Networks*, v. 7, n. 3, pp. 537–568, May 2009. ISSN: 15708705. doi: 10.1016/j.adhoc.2008.06.003. Disponível em: <http://dx.doi.org/10.1016/j.adhoc.2008.06.003>.

[46] VASSIS, D., KORMENTZAS, G., ROUSKAS, A., etal. "The IEEE 802.11g standard for high data rate WLANs", *Network, IEEE*, v. 19, n. 3, pp. 21–26, December 2005. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1453395>.

[47] BRUNO BOUGARD, FRANCKY CATTHOOR, D. C. D. A. C. W. D. "Energy Efficiency of the IEEE 802.15.4 Standard in Dense Wireless Microsensor Networks: Modeling and Improvement Perspectives". In: *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pp. 196–201, Washington, DC, USA, November 2005. IEEE Computer Society. ISBN: 0-7695-2288-2. doi: http://dx.doi.org/10.1109/DATE.2005.136.

[48] LI, N., HOU, J. C. "FLSS: a fault-tolerant topology control algorithm for wireless networks". In: *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pp. 275–286, New York, NY, USA, November 2004. ACM. ISBN: 1-58113-868-7. doi: http://doi.acm.org/10.1145/1023720.1023747.

[49] CHEN, B., JAMIESON, K., BALAKRISHNAN, H., etal. "Span: An energy-efficient coordination algorithm for topology maintenance in Ad Hoc wireless networks". In: *Mobile Computing and Networking*, pp. 85–96, November 2001. Disponível em: <`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.8811`>.

[50] KLEINROCK, L., SILVESTER, J. "Optimum transmission radii for packet radio networks or why six is a magic number". In: *NTC '78; National Telecommunications Conference, Birmingham, Ala., December 3-6, 1978, Conference Record. Volume 1. (A79-40501 17-32) Piscataway, N.J., Institute of Electrical and Electronics Engineers, Inc., 1978, p. 4.3.1-4.3.5.*, v. 1, 1978.

[51] BLUMENTHAL, J., REICHENBACH, F., TIMMERMANN, D. "Minimal Transmission Power vs. Signal Strength as Distance Estimation for Localization in Wireless Sensor Networks". In: *3rd IEEE International Workshop on Wireless Ad-hoc and Sensor Networks*, pp. 761–766, June 2006. New York, USA.

[52] *The limits of localization using signal strength: a comparative study*, July 2004. doi: 10.1109/SAHCN.2004.1381942. Disponível em: <`http://dx.doi.org/10.1109/SAHCN.2004.1381942`>.

[53] GOMEZ, J., CAMPBELL, A. T. "Variable-Range Transmission Power Control in Wireless Ad Hoc Networks", *IEEE Transactions on Mobile Computing*, v. 6, n. 1, pp. 87–99, September 2007. ISSN: 1536-1233. doi: http://dx.doi.org/10.1109/TMC.2007.17.

[54] GUPTA, P., KUMAR. "Critical power for asymptotic connectivity in wireless networks", *In Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W. H. Fleming*, pp. 547–566, June 1998.

[55] DENG, J., HAN, Y. S., CHEN, P.-N., etal. "Optimal Transmission Range for Wireless Ad Hoc Networks Based on Energy Efficiency", *IEEE Trans. on Communications*, v. 55, n. 9, pp. 1772–1782, September 2007.

[56] GAO, J. L. "Analysis of energy consumption for ad hoc wireless sensor networks using a bit-meter-per-joule metric." *IPN Progress Report*, pp. 42–150, October 2002.

[57] MONTEIRO, A. C., VIANNA, R. G., DE CASTRO DUTRA, R., etal. "Sistema de Medição Precisa do Consumo de Energia em Dispositivos Móveis de Comunicação Sem Fio". In: *VIII Workshop em Sistemas Computacionais de Alto Desempenho*, Anais do VIII Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD), Gramado, RS, Brazil, October 2007. (in Portuguese).

[58] SARKAR, T., JI, Z., KIM, K., etal. "A survey of various propagation models for mobile communication", *Antennas and Propagation Magazine, IEEE*, v. 45, n. 3, pp. 51–82, June 2003. ISSN: 1045-9243. doi: 10.1109/MAP.2003.1232163.

[59] FRIIS, H. T. "A Note on a Simple Transmission Formula", *Proceedings of the IRE*, v. 34, n. 5, pp. 254–256, May 1946. Disponível em: <`http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1697062`>.

[60] SRINIVASAN, V., MOTANI, M., OOI, W. T. "Analysis and implications of student contact patterns derived from campus schedules". In: *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pp. 86–97, New York, NY, USA, August 2006. ACM. ISBN: 1-59593-286-0. doi: http://doi.acm.org/10.1145/1161089.1161100.

[61] KIM, M., KOTZ, D., KIM, S. "Extracting a mobility model from real user traces". In: *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Barcelona, Spain, April 2006. IEEE Computer Society Press. doi: 10.1109/INFOCOM.2006.173. Disponível em: <`http://www.cs.dartmouth.edu/~dfk/papers/kim-mobility.pdf`>.

[62] GROUT, V. "Measuring Network Traffic: The Hidden Distribution". In: *Mathematics Today: Bulletin of the Institute of Mathematics and its Applications*, v. 6, pp. 86–97, December 2001.

[63] GROENEVELT, R., NAIN, P., KOOLE, G. "Message delay in MANET", *SIGMETRICS Perform. Eval. Rev.*, v. 33, n. 1, pp. 412–413, October 2005. ISSN: 0163-5999. doi: http://doi.acm.org/10.1145/1071690.1064280.

[64] SHARMA, G., MAZUMDAR, R., SHROFF, N. B. "Delay and capacity tradeoffs in mobile ad hoc networks: a global perspective", *IEEE/ACM Trans. Netw.*, v. 15, n. 5, pp. 981–992, October 2007. ISSN: 1063-6692. doi: http://dx.doi.org/10.1109/TNET.2007.905154.

[65] "The Network Simulator NS-2". . Disponível em: <`http://www.isi.edu/nsnam/ns/`>.

[66] KERÄNEN, A., OTT, J., KÄRKKÄINEN, T. "The ONE Simulator for DTN Protocol Evaluation". In: *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, October 2009. ICST. ISBN: 978-963-9799-45-5.

[67] "Tmote Sky Low Power Wireless Sensor Module". . http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf.

[68] MOUSAVI, S. M., RABIEE, H. R., MOSHREF, M., etal. "MobiSim: A Framework for Simulation of Mobility Models in Mobile Ad-Hoc Networks". In: *WIMOB '07: Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, p. 82, Washington, DC, USA, June 2007. IEEE Computer Society. ISBN: 0-7695-2889-9.

[69] STEUER, R. E. *Multiple Criteria Optimization: Theory, Computation and Application*, v. 3. Radio e Svyaz, Moscow, 504 pp., 1992. (in Russian).

[70] DEB, K., AGRAWAL, S., PRATAP, A., etal. "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II". In: *Parallel Problem Solving from Nature PPSN VI*, pp. 849–858. Springer, October 2000.

[71] TAKAI, M., MARTIN, J., BAGRODIA, R. "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks". , August 2001. Disponível em: <`http://citeseer.ist.psu.edu/takai01effects.html`>.

[72] SEIDEL, S.Y.; RAPPAPORT, T. "914 MHz path loss prediction models for indoor wireless communications in multifloored buildings", *Antennas and Propagation, IEEE Transactions on*, v. 40, n. 5, pp. 254–256, February 1992.

[73] BETTSTETTER, C., HARTMANN, C. "Connectivity of wireless multihop networks in a shadow fading environment". In: *MSWIM '03: Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pp. 28–32, New York, NY, USA, August 2003. ACM. ISBN: 1-58113-766-4. doi: http://doi.acm.org/10.1145/940991.940998.

[74] GAO, Q., BLOW, K. J., HOLDING, D. J., etal. "Radio range adjustment for energy efficient wireless sensor networks", *Ad Hoc Networks*, v. 4, n. 1, pp. 75–82, January 2006.

[75] XI, Y., CHUAH, M., CHANG, K. "Performance Evaluation of a Power Management Scheme for Disruption Tolerant Network." *MONET*, v. 12, n. 5-6, pp. 370–380, November 2007. Disponível em: <`http://dblp.uni-trier.de/db/journals/monet/monet12.html`>.

[76] ESTRIN, D., CULLER, D., PISTER, K., etal. "Connecting the Physical World with Pervasive Networks", *IEEE Pervasive Computing*, v. 1, n. 1, pp. 59–69, May 2002. ISSN: 1536-1268. doi: http://dx.doi.org/10.1109/MPRV.2002.993145.

[77] S. GUO, M.H. FALAKI, E. O. S. U. R. A. S. M. Z. U. I., KESHAV, S. "Design and Implementation of the KioskNet System". In: *International Conference on Information Technologies and Development*, January 2007.

[78] WANG, A. Y., SODINI, C. G., WANG, A. "A Simple Energy Model for Wireless Microsensor Transceivers". In: *Proceedings of IEEE Global Telecommunications Conference*, pp. 3205–3209, September 2003.

[79] CHRISTIAN BETTSTETTER, HANNES HARTENSTEIN, XAVIER PÉREZ-COSTA. "Stochastic properties of the random waypoint mobility model: epoch length, direction distribution and cell-change rate". In: *Proceedings of the 5th ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM), Atlanta, Georgia, USA*, September 2002.

[80] ANTONIOU, T., CHATZIGIANNAKIS, I., MYLONAS, G., etal. "A New Energy Efficient and Fault-tolerant Protocol for Data Propagation in Smart

Dust Networks Using Varying Transmission Range". In: *ANSS '04: Proceedings of the 37th annual symposium on Simulation*, p. 43, Washington, DC, USA, August 2004. IEEE Computer Society. ISBN: 0-7695-2110-X.

[81] JUN, H., AMMAR, M. H., ZEGURA, E. W. "Power management in delay tolerant networks: a framework and knowledge-based mechanisms". In: *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, pp. 418–429, August 2005. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1557095>.

[82] CERF, V., CLARK, D., KAHN, R., etal. "Histories of the Internet". . Disponível em: <http://www.isoc.org/internet/history/brief.shtml>.

[83] POSTEL, J. "RFC 791: Internet Protocol". , set. 1981. Disponível em: <ftp://ftp.internic.net/rfc/rfc760.txt>. Obsoletes RFC0760 [?]. See also STD0005 [?]. Status: STANDARD.

[84] IACONO, A., ROSE, C. "Infostations: A new perspective on wireless data networks". , August 2000. Disponível em: <http://citeseer.ist.psu.edu/iacono00infostations.html>.

[85] CHEN, P., O'DEA, B., CALLAWAY, E. "Energy efficient system design with optimum transmission range for wireless ad hoc networks". In: *Communications, 2002. ICC 2002. IEEE International Conference on*, v. 2, pp. 945–952, July 2002.

Apêndice A

Código do Simulador

O simulador orientado a eventos discretos utilizado, foi implementado em linguagem ANSI C++. Esta linguagem é orientada a objetos. Os objetos modelados foram: lista de eventos, nós computacionais e eventos.

Cada objeto possui um arquivo de cabeçalho onde são feitas as definições da classe e um arquivo em que os métodos da classe são implementados. Além disso, existe um arquivo principal no qual é implementada a função *main*, função principal em C++.

O simulador tem como entradas:

- *-nn NUM*: número de nós simulados
- *-ip PROBE*: intervalo entre probes
- *-pot POT*: potência máxima do sinal transmissão
- *-eft POT*: potência efetiva do sinal transmissão
- *-mov PATH*: arquivo de movimentação dos nos
- *-alg ALGORITMO*: algoritmo para cálculo da potência de transmissão, podendo ser:

  0 - Nenhum - utiliza a potência máxima

  1 - Additive Increase Multiplicative Decrease

  2 - Knee - Exponencial

  3 - Knee - Pareto

  4 - Multiplicative Increase Additive Decrease

Como saídas o simulador pode gerar:

- 0 - Probabilidade de perder oportunidades de contato

- 1 - Distribuição do tempo de contato
- 2 - Função de densidade de probabilidade do tempo de contato
- 3 - Variação dos tempos de contatos

O arquivo de movimentação dos nós deve ser da forma: NODEID, TEMPO, X, Y, como pode ser visto no Arquivo A.1. Neste temos a movimentação de dois dispositivos por uma área de $100x100\ m^2$ durante 15 segundos.

```
NODEID      T          X          Y
  0      1.000     65.000     31.000
  1      1.000     22.000     59.000
  0      2.000     63.000     32.000
  1      2.000     23.000     59.000
  0      3.000     62.000     33.000
  1      3.000     25.000     59.000
  0      4.000     61.000     33.000
  1      4.000     26.000     58.000
  0      5.000     59.000     34.000
  1      5.000     28.000     58.000
  0      6.000     58.000     35.000
  1      6.000     29.000     58.000
  0      7.000     56.000     35.000
  1      8.000     31.000     57.000
  0      9.000     55.000     36.000
  1      9.000     32.000     57.000
  0      10.000    53.000     36.000
  1      10.000    34.000     57.000
  0      11.000    52.000     37.000
  1      11.000    35.000     56.000
  0      12.000    51.000     38.000
  1      12.000    37.000     56.000
  0      13.000    49.000     38.000
  1      13.000    38.000     56.000
  0      14.000    48.000     39.000
  1      14.000    40.000     55.000
  0      15.000    46.000     39.000
  1      15.000    41.000     55.000
  0      16.000    45.000     40.000
  1      16.000    43.000     55.000
```

Arquivos A.1: *Exemplo de trace de movimentação*

O código foi compilado em um sistema Linux Red Hat Release 5 Update 4, com compilador GCC 4.3.3. Para a compilação também foi utilizado o sistema *Makefile*, responsável por armazenar os parâmetros de compilação e gerar os arquivos binários. O arquivo *makefile* utilizado com todos os parâmetros pode ser visualizado em A.2.

Embora tenha sido desenvolvido em um ambiente Linux, por ter sido escrito em linguagem Ansi C++, este código pode ser compilado para outras plataformas como Windows, Unix e MacOS.

Para obtenção dos dados foram realizadas diversas rodadas de simulação, variando-se os arquivos de movimentação, potência efetiva e intervalo entre *probes*. No total foram executadas mais de 220000 simulações em uma máquina com sistema operacional Linux Red Hat Release 5 Update 4 (a mesma utilizada no desenvolvimento), processador Intel Xeon 5450 com 4 núcleos e memória principal de 8GB.

```
RM := rm -rf

# All of the sources participating in the build are defined here
# Add inputs and outputs from these tool invocations to the build
    variables
CPP_SRCS += \
../TComputador.cpp \
../TEvCheckCont.cpp \
../TEvMoving.cpp \
../TEvProbe.cpp \
../TEvento.cpp \
../TFileHandler.cpp \
../TListaEvento.cpp \
../TRandomGen.cpp \
../simvoip.cpp

OBJS += \
./TComputador.o \
./TEvCheckCont.o \
./TEvMoving.o \
./TEvProbe.o \
./TEvento.o \
./TFileHandler.o \
./TListaEvento.o \
./TRandomGen.o \
./simvoip.o

CPP_DEPS += \
./TComputador.d \
./TEvCheckCont.d \
./TEvMoving.d \
./TEvProbe.d \
./TEvento.d \
./TFileHandler.d \
./TListaEvento.d \
```

```
./TRandomGen.d \
./simvoip.d


# Each subdirectory must supply rules for building sources it
    contributes
%.o: ../%.cpp
	@echo 'Building file: $<'
	@echo 'Invoking: GCC C++ Compiler'
	g++ -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"$(@:%.o=%.d)
	    " -MT"$(@:%.o=%.d)" -o"$@" "$<"
	@echo 'Finished building: $<'
	@echo ' '


ifneq ($(MAKECMDGOALS),clean)
ifneq ($(strip $(C++_DEPS)),)
-include $(C++_DEPS)
endif
ifneq ($(strip $(C_DEPS)),)
-include $(C_DEPS)
endif
ifneq ($(strip $(CC_DEPS)),)
-include $(CC_DEPS)
endif
ifneq ($(strip $(CPP_DEPS)),)
-include $(CPP_DEPS)
endif
ifneq ($(strip $(CXX_DEPS)),)
-include $(CXX_DEPS)
endif
ifneq ($(strip $(C_UPPER_DEPS)),)
-include $(C_UPPER_DEPS)
endif
endif

-include ../makefile.defs

# Add inputs and outputs from these tool invocations to the build
    variables

# All Target
all: simulador

# Tool invocations
simulador: $(OBJS) $(USER_OBJS)
	@echo 'Building target: $@'
```

```
79    @echo 'Invoking: GCC C++ Linker'
80    g++  -o"simulador" $(OBJS) $(USER_OBJS) $(LIBS)
81    @echo 'Finished building target: $@'
82    @echo ' '
83
84  # Other Targets
85  clean:
86    -$(RM) $(OBJS)$(C++_DEPS)$(C_DEPS)$(CC_DEPS)$(CPP_DEPS)$(
          EXECUTABLES)$(CXX_DEPS)$(C_UPPER_DEPS) simulador
87    -@echo ' '
88
89  .PHONY: all clean dependents
90  .SECONDARY:
```

Arquivos A.2: *Arquivo contento os parâmetros utilizados na compilação do simulador.*

Como exemplo de utilização do simulador, considere o cenário com parâmetros retirados da Tabela A.1. Considere também o Arquivo A.1 como *trace* de movimentação utilizado no exemplo.

Parâmetros da Simulação			
Característica do Cenário		**Características do Rádio e Sinal**	
Tempo de Simulação:	10 s	**Potência Máxima de Tx.:**	0,1 mW
Modelo de Mobilidade:	RandomWaypoint	**Sensitividade da antena:**	-85 dBm
Velocidade Máxima:	1,58 m/s	**Frequência:**	2,4 GHz
Velocidade Mínima:	1,52 m/s	**Comprimento de Onda:**	0,125 m
Tempo de Pausa:	0 s	**Alcance máximo:**	57,43 m
Dimensões da região:	100x100 m²	**Algoritmo:**	0
Número de Dispositivos:	2	**Tipo Saída:**	0

Tabela A.1: *Parâmetros utilizados no exemplo de execução da simulação.*

Para o caso de uma execução do simulador em que o intervalo entre *probes* seja de 1 segundo e a potência efetiva seja dada por zero, executamos o seguinte comando A.3:

```
1  ./simulador -nn 2 -pot 0.1 -eft 0.0 -mov mov.conf -ip 1.0 -alg 0 -
       out 0
```

Arquivos A.3: *Exemplo de execução do simulador.*

Neste caso, como a potência efetiva utilizada é zero, a probabilidade de perder oportunidades de contato será 1.

A listagem a seguir apresenta todos os arquivos-fonte que compõem o simulador de eventos. Estes também estão disponíveis em mídia digital que acompanha a dissertação.

O Arquivo A.4, define constantes utilizadas no programa.

```
/*****************************************************************
* constantes.h                                                   *
* Define constantes utilizadas no programa.                      *
*                                                                *
* Desenvolvido por Tiago Souza Azevedo                           *
* Laboratorio de Redes de Alta Velocidade                        *
* Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ   *
*                                                                *
* Contato: tiago@ravel.ufrj.br                                   *
*****************************************************************/

#ifndef CONSTANTES_H_
#define CONSTANTES_H_

#define PI         3.1415  //Numero Pi
#define LAMBDA       0.125 //Comprimento de onda
#define RSENSITIVITY  0.000000003 //Sensitividade do radio
#define NUMARGS      15     //Numero de argumentos
#define IDFIELD      0    //Posicao do campo Id
#define TFIELD       1    //Posicao do campo Time
#define XFIELD       2    //Posicao do campo X
#define YFIELD       3    //Posicao do campo Y
#define AMOSTRAS     100    //Qtd amostras fronteira de Pareto
#define STEP       0.01   //Saltos entre amostra de pontos
#define MINEFT       0.2
#define MAXEFT       1.0
#define EPS        0.0001


#endif /*CONSTANTES_H_*/
```

Arquivos A.4: *Arquivo-fonte*:constantes.h

O Arquivo A.5, define variáveis globais utilizadas no simulador.

```
1  /*****************************************************************
2  * globais.h                                                     *
3  * Define variaveis e funcoes globais utilizadas no programa.    *
4  *                                                               *
5  * Desenvolvido por Tiago Souza Azevedo                          *
6  * Laboratorio de Redes de Alta Velocidade                       *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ  *
8  *                                                               *
9  * Contato: tiago@ravel.ufrj.br                                  *
10 *****************************************************************/
11
12 #ifndef GLOBAIS_H_
13 #define GLOBAIS_H_
14
15 #include <vector>
16 #include <string>
17 #include <iostream>
18 #include <cstdio>
19 #include <cstdlib>
20 #include <cmath>
21
22 using namespace std;
23
24 #include "TRandomGen.h"
25 #include "TListaEvento.h"
26 #include "TComputador.h"
27 #include "TFileHandler.h"
28
29 //-----------------------------------------
30 // Variaveis Globais de Controle do Cenario
31 //-----------------------------------------
32 extern TRandomGen randomGen;
33 extern TListaEvento *listaEventos;
34 extern TFileHandler *fileMov;
35 extern vector<TComputador *> simNodes;
36 extern double tempoAtualSim;
37 extern int rodadaAtual;
38 extern unsigned int numNodes;
39 extern float potEfet;
40 extern vector<unsigned int> tempoContato;
41 extern unsigned int rangeTempoContato;
42 extern int algId;
43 extern int outType;
44
```

```
45 //--------------------------------------------
46 // Funcoes Globais
47 //--------------------------------------------
48 extern float calcDistancia(float posX1, float posY1, float posX2,
       float posY2);
49 extern float calDist(float x, float y);
50 extern float funcProb(float x, float m, float T);
51
52 #endif /*GLOBAIS_H_*/
```

Arquivos A.5: *Arquivo-fonte*:globais.h

O Arquivo A.6, implementa a função principal do simulador. Neste são executadas as rodadas da simulação e posteriormente o cálculo das estatísticas.

```
1  /*****************************************************************
2  * simulador.cpp                                                 *
3  * Execução principal do programa.                               *
4  *                                                               *
5  * Desenvolvido por Tiago Souza Azevedo                          *
6  * Laboratorio de Redes de Alta Velocidade                       *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ  *
8  *                                                               *
9  * Contato: tiago@ravel.ufrj.br                                  *
10 *****************************************************************/
11
12
13 #include <cstdio>
14 #include <cstdlib>
15 #include <string>
16 #include <cstring>
17 #include <iostream>
18 #include <iomanip>
19 #include <sstream>
20 #include <vector>
21 #include <cmath>
22 #include <time.h>
23
24 using namespace std;
25
26 #include "TRandomGen.h"
27 #include "TListaEvento.h"
28 #include "TComputador.h"
29 #include "TEvento.h"
30 #include "TEvProbe.h"
31 #include "TFileHandler.h"
32 #include "TEvMoving.h"
33 #include "TEvCheckCont.h"
34 #include "constantes.h"
35
36
37 //*********************************************
38 // Variaveis Globais de Controle do Cenario
39 // ----------------------------------------
40 // Algumas sao usadas pelos outros modulos do
41 // programa (eventos, etc). Estas
42 // sao definidas como extern no arquivo globais.h
43 //*********************************************
```

```
44
45 // Gerador de numeros aleatorios e amostras de
46 // distribuicoes
47 TRandomGen randomGen(234346553);
48
49 // Lista de eventos
50 TListaEvento *listaEventos;
51
52 // File Handler
53 TFileHandler *fileMov;
54
55
56 // Tempo atual da simulacao (em segundos)
57 double tempoAtualSim;
58
59 // Numero de nos
60 unsigned int numNodes;
61 vector<TComputador *>  simNodes;
62
63 //Intervalo entre probes em segundos
64 float intProbes;
65
66 //Caminho do arquivo de movimentacao
67 string movPath;
68
69 //Potencia maxima do sinal de transmissao medida em mW
70 float potMax;
71
72 // Potencia efetiva do sinal de transmissao
73 // medida em porcentagem da potencia total
74 float potEfet;
75
76 // Tipo de saida
77 // 0 - Probabilidade de perder op. de contato
78 // 1 - Distribuicao do tempo de contato
79 // 2 - Funcao de densidade de probabilidade do tempo de contato
80 // 3 - Variacao do tamanho medio do tempo de contato
81 int outType;
82
83 // Tempo de contato
84 vector<unsigned int> tempoContato;
85 unsigned int rangeTempoContato;
86
87 //Prob. Media de perder uma oportunidade de contato
88 float probMediaPC;
89
90 //Algoritmo de escolha do sinal
```

```
91  int algId;
92
93  bool fim = false;
94  bool resetado = false;
95
96  //Consumo medio
97  double consumoMedio;
98  double consumoVar;
99
100
101 //*************************************
102 // Help do programa
103 //*************************************
104
105 #define USAGE " \
106 \n\
107 simvoip [OPCOES]\n\
108 \n\
109 OPCOES:\n\
110  -nn NUM_R\n\
111     numero de nos NUM_N\n\
112     default: 100\n\
113     \n\
114  -ip PROBE\n\
115     intervalo entre probes PROBE\n\
116     default: 5s\n\
117     \n\
118  -pot POT\n\
119     potencia max sinal tx\n\
120     default: 0.1 mW\n\
121     \n\
122  -eft POT\n\
123     potencia efetiva sinal tx\n\
124     default: 0.8 max\n\
125     \n\
126  -mov PATH_R\n\
127     arquivo de movimentacao dos nos PATH\n\
128     default: ./mov.conf\n\
129     \n\
130  -alg ALGORITMO\n\
131       0 - Nenhum\n\
132       1 - Additive Increase Multiplicative Decrease\n\
133       2 - Knee - Exponencial\n\
134       3 - Knee - Pareto\n\
135       4 - Multiplicative Increase Additive Decrease\n\
136     default: 0\n\
137     \n\
```

```
138  -out OUTPUT\n\
139        0 - Probabilidade de perder op. de contato\n\
140        1 - Distribuicao do tempo de contato\n\
141        2 - Funcao de densidade de prob. do tempo de contato\n\
142        3 - Variacao dos tempos de contatos\n\
143      default: 0\n\
144      \n\
145 \n"
146
147
148 void finalizaSimulador();
149 void finalizaDados();
150 bool setup(int argc, char **argv);
151 float calcDistancia(float posX1, float posY1, float posX2, float
       posY2);
152 void resetEstatisticas();
153
154 //Funcoes
155 float calDist(float ax, float ay, float bx, float by)
156 {
157     return sqrt((ax*bx)-(ay*by));
158 }
159
160 float funcProb(float x, float m, float T)
161 {
162     float y = m*T;
163     return ((exp(-y)-1+y)/y)+((2*(cosh(y)-1)/y)*(exp(-y)*(1-x))
           /(1-(exp(-y)*(1-x))));
164 }
165
166
167 //***************************************
168 // Funcao Main
169 //***************************************
170 int main(int argc, char **argv)
171 {
172     //-----------------------------------
173     // INICIALIZANDO O SIMULADOR
174     // -------------------------
175     //  Preparando os valores iniciais para
176     // as variaveis de cenario e controle
177     //-----------------------------------
178     if(!setup(argc, argv))
179     {
180         printf(USAGE);
181         return 1;
182     }
```

```
183
184
185     //Inicializando os nos
186     for(int i=0; i<numNodes;i++)
187     {
188         float X,Y;
189         int ID;
190
191         if(!fileMov->readNextPos())
192         {
193             cout << "Erro ao ler arquivo na inicializacao dos nos!"
                        << endl;
194             finalizaSimulador();
195             return 1;
196         }
197
198         X = atof(fileMov->dataFields[XFIELD].c_str());
199         Y = atof(fileMov->dataFields[YFIELD].c_str());
200         ID = atoi(fileMov->dataFields[IDFIELD].c_str());
201
202         TComputador *novo = new TComputador(ID, intProbes, potMax,
                potEfet, X, Y);
203         simNodes.push_back(novo);
204
205     }
206
207     //Preciso inserir na lista de eventos, o evento do primeiro
            probe de cada
208     // no. Esses instantes serao gerados aleatoriamente(distrib.
            uniforme) no intervalo
209     // de 1 a intProbes segundos;
210     for(int i=0; i<numNodes;i++)
211     {
212         TEvProbe *ev;
213         double tempo;
214         tempo = 2 + randomGen.Random()*intProbes;
215
216         ev = new TEvProbe(tempo,simNodes[i]);
217         if(!listaEventos->insereEvento(ev))
218         {
219             cout << "Erro ao inserir os primeiros eventos de probe
                        na lista!" << endl;
220             finalizaSimulador();
221             return 1;
222         }
223
224     }
```

```
225
226     //Preciso inserir na lista de eventos,
227     // o evento do primeiro movimento
228     TEvMoving *ev;
229
230     //Obtendo dados sobre movimento do arquivo
231     if(!fileMov->readNextPos())
232     {
233         cout << "Erro ao ler arquivo na inicializacao do evento do
                primeiro movimento!" << endl;
234         finalizaSimulador();
235         return 1;
236     }
237
238     ev = new TEvMoving(atof(fileMov->dataFields[TFIELD].c_str()));
239     if(!listaEventos->insereEvento(ev))
240     {
241         cout << "Erro ao inserir os primeiros eventos de movimento
                na lista!" << endl;
242         finalizaSimulador();
243         return 1;
244     }
245
246     //--------------------------------------------
247     // RODADAS DE SIMULAcaO
248     //--------------------------------------------
249     // Comeco aqui as rodadas de simulacao
250     // condicao de parada e lista de eventos vazia
251     while(1)
252     {
253         // Variavel usada como handler para os eventos da lista de
                eventos
254         // Considero o proximo evento para execucao
255         TEvento *evento = listaEventos->proximoEvento();
256
257         if(evento) // != NULL
258         {
259             // Atualizo o tempo da simulacao para o tempo deste
                    evento
260             tempoAtualSim = evento->tempo;
261
262             // Faco o evento ocorrer
263             if( ! evento->ocorre() )
264             {
265                 // Houve algum erro na ocorrencia do evento
266                 cout << "**ERRO NA OCORRENCIA DO EVENTO!**" << endl
                        ;
```

```
267                 cout << evento->toString();
268                 delete evento;
269                 break;
270             }
271             // Ocorreu normalmente. Desaloco a
272             // memoria ocupada por este evento,
273             // e continuo a simulacao
274             delete evento;
275         }
276         else
277         {
278             // Lista vazia (ou algum erro). Paro o simulador
279             //cout << "**LISTA VAZIA**" << endl;
280             break;
281         }
282     }
283
284     //-----------------------------------------------
285     // FIM DO SIMULADOR
286     // ----------------
287     //  Todas as variaveis alocadas sao liberadas
288     //-----------------------------------------------
289     finalizaDados();
290
291     //Calcula o numero total de contatos fisicos
292     float somaTotal = 0.0;
293     float mediaContato = 0.0;
294     float somaParcial = 0.0;
295
296     for(int i=0; i<tempoContato.size(); i++)
297         somaTotal += tempoContato[i];
298
299     switch(outType)
300     {
301         case 0:
302         {
303             // Calcula a probabilidade media de perder uma
                    oportunidade de contato
304             // ao nao realizar um probe(contato logico) em um
                    contato fisico
305             for(int i=0; i<numNodes; i++)
306             {
307                 int counter = 0;
308                 for(int j=0; j< numNodes; j++)
309                 {
310                     if((i != j) && (simNodes[i]->contFisico[j] !=
                            0))
```

```
311                     {
312                         simNodes[i]->probPerdaM += (simNodes[i]->
                               contLogico[j]/simNodes[i]->contFisico[j
                               ]);
313                         counter++;
314                     }
315                 }
316                 simNodes[i]->probPerdaM = 1-(simNodes[i]->
                       probPerdaM/counter);
317             }
318 
319             for(int i=0; i<numNodes; i++)
320                 probMediaPC += simNodes[i]->probPerdaM;
321 
322             probMediaPC = probMediaPC/numNodes;
323 
324             float probVarPC = 0.0;
325             float intConf = 0.0;
326 
327             for(int i=0; i<numNodes; i++)
328                 probVarPC += (simNodes[i]->probPerdaM - probMediaPC
                       ) * (simNodes[i]->probPerdaM - probMediaPC);
329 
330             probVarPC = probVarPC / numNodes;
331 
332             //Calculo o consumo medio e variancia
333             consumoMedio = 0.0;
334             for(int i=0; i<numNodes; i++)
335                 consumoMedio += simNodes[i]->consumo;
336 
337             consumoMedio = consumoMedio/numNodes;
338 
339             consumoVar = 0.0;
340             for(int i=0; i<numNodes; i++)
341                 consumoVar += (simNodes[i]->consumo - consumoMedio)
                       * (simNodes[i]->consumo - consumoMedio);
342 
343             consumoVar = consumoVar / numNodes;
344             cout << probMediaPC <<  "\t" << probVarPC << "\t" <<
                   consumoMedio << "\t" << sqrt(consumoVar) << endl;
345         }
346             break;
347 
348         case 1:
349         {
350             //Calcula a CDF
351             for(int i=0; i<tempoContato.size(); i++)
```

```
352             {
353                 mediaContato += i*(tempoContato[i]/somaTotal);
354                 somaParcial += tempoContato[i];
355                 cout << i << "\t" << somaParcial/somaTotal << endl;
356             }
357 
358             cout << "#"<< mediaContato << endl;
359             cout << "#"<< 1/mediaContato << endl;
360         }
361             break;
362         case 2:
363         {
364             //Calcula a PDF
365             for(int i=0; i<tempoContato.size(); i++)
366                 cout <<  i << " \t " << tempoContato[i]/somaTotal
                        << endl;
367 
368             break;
369         }
370     }
371     finalizaSimulador();
372 
373     return 0;
374 }
375 
376 
377 //***********************************
378 // Funcoes auxiliares
379 //***********************************
380 
381 //-----------------------------------
382 // Verifica os parametros do programa
383 // e inicializa todas as variaveis do
384 // simulador
385 //-----------------------------------
386 bool setup(int argc, char **argv)
387 {
388     short int argVistos = 0;
389     if(argc > NUMARGS) return false;
390 
391     //---------------------------------------
392     // Carrega defaults para os parametros
393     //---------------------------------------
394     intProbes = 5;
395     numNodes = 100;
396     potMax = 0.1;  // em mW
397     potEfet = 0.8;  // em porcentagem da potMax
```

```
398     movPath = "./mov.conf";
399     outType = 0;
400     algId = 0;
401     //----------------------------------------
402     // Checa os parametros da linha de comando
403     //----------------------------------------
404     for(int i=0; i < argc; i++)
405     {
406         if(strcmp(argv[i],"-alg") == 0)
407         {
408             algId = atoi(argv[i+1]);
409             i++;
410             argVistos+=2;
411             continue;
412         }
413         if(strcmp(argv[i],"-ip") == 0)
414         {
415             intProbes = atof(argv[i+1]);
416             i++;
417             argVistos+=2;
418             continue;
419         }
420         if(strcmp(argv[i],"-pot") == 0)
421         {
422             potMax = atof(argv[i+1]);
423             i++;
424             argVistos+=2;
425             continue;
426         }
427         if(strcmp(argv[i],"-eft") == 0)
428         {
429             potEfet = atof(argv[i+1]);
430             i++;
431             argVistos+=2;
432             continue;
433         }
434         if(strcmp(argv[i],"-nn") == 0)
435         {
436             numNodes = atoi(argv[i+1]);
437             i++;
438             argVistos+=2;
439             continue;
440         }
441         if(strcmp(argv[i],"-mov") == 0)
442         {
443             movPath = argv[i+1];
444             i++;
```

```
445             argVistos+=2;
446             continue;
447         }
448         if(strcmp(argv[i],"-out") == 0)
449         {
450             outType = atoi(argv[i+1]);
451             i++;
452             argVistos+=2;
453             continue;
454         }
455     }
456 
457     if(((argVistos+1) != argc) && (argc != 1)) return false;
458 
459     //-------------------------------------------------------
460     // Configurando o cenario de acordo com os parametros
461     //-------------------------------------------------------
462 
463     // Inicializando a lista de eventos
464     listaEventos = new TListaEvento();
465 
466     // Zerando o contador do tempo total de simulacao
467     tempoAtualSim = 0.0; // segundos
468 
469     //Inicializa fileHandler
470     fileMov = new TFileHandler(movPath);
471 
472     //Inicializa vetor tempoContato
473     tempoContato.assign(1,0.0);
474     rangeTempoContato = 1;
475 
476     //Inicializa prob
477     probMediaPC = 0.0;
478 
479     return true;
480 }
481 //--------------------------------
482 // Reseta as estatisticas de todos
483 // os dispositivos
484 //--------------------------------
485 void resetEstatisticas()
486 {
487     for(int i=0; i<numNodes; i++)
488     {
489         simNodes[i]->consumo = 0.0;
490         simNodes[i]->numProbeTx = 0;
491         for(int k=0; k<numNodes; k++)
```

```
492         {
493             if(simNodes[i]->statusContFisico[k])
494             {
495                 simNodes[i]->contFisico[k] = 1.0;
496                 if(simNodes[i]->statusContLogico[k])
497                     simNodes[i]->contLogico[k] = 1.0;
498                 simNodes[i]->inicioContFisico[k] = 0.0;
499 
500             }
501             simNodes[i]->inicioContFisico[k] = 0.0;
502             simNodes[i]->contFisico[k] = 0.0;
503             simNodes[i]->contLogico[k] = 0.0;
504             simNodes[i]->statusContFisico[k]=false;
505             simNodes[i]->statusContLogico[k]=false;
506         }
507 
508     }
509 }
510 //---------------------------------
511 // Desaloca qualquer espaco de
512 // memoria alocado no simulador
513 //---------------------------------
514 void finalizaDados()
515 {
516 
517     for(int i=0; i< numNodes; i++)
518     {
519         for(int k = 0; k< numNodes; k++)
520         {
521             if(i < k) continue;
522             if(simNodes[i]->statusContFisico[k])
523             //se estavam em contato
524             {
525                 float intContatoFisico;
526 
527                 intContatoFisico = (int)(tempoAtualSim - simNodes[i
                        ]->inicioContFisico[k]);
528                 if((int)intContatoFisico > rangeTempoContato)
529                 {
530                     tempoContato.resize((int)intContatoFisico
                            +1,0.0);
531                     rangeTempoContato = (int)(intContatoFisico+1);
532                 }
533                 tempoContato[(int)intContatoFisico]++;
534             }
535             simNodes[i]->statusContFisico[k] = false;
536         }
```

```
537     }
538 }
539
540 //------------------------------
541 // Desaloca qualquer espaco de
542 // memoria alocado no simulador
543 //------------------------------
544 void finalizaSimulador()
545 {
546
547     if(listaEventos != NULL) delete listaEventos;
548
549     if(fileMov != NULL) delete fileMov;
550
551     for(int i=0; i<simNodes.size(); i++)
552         delete simNodes.at(i);
553
554 }
555
556 //---------------------------------
557 // Gera uma saida em modo texto com
558 // todas as informacoes sobre o
559 // estado atual do cenario
560 //---------------------------------
561 void imprimeCenario()
562 {
563     cout << "----------------------------------------" << endl;
564     cout << "TEMPO SIMULACAO: " << tempoAtualSim << " milisegundos"
        << endl;
565     cout << "----------------------------------------" << endl;
566     cout << listaEventos->toString() << endl;
567 }
568 //---------------------------------
569 // Calcula a distancia Euclidiana
570 // entre dois dispositivos
571 //---------------------------------
572 float calcDistancia(float posX1, float posY1, float posX2, float
    posY2)
573 {
574     return sqrt(pow((posX1 - posX2),2) + pow((posY1 - posY2),2));
575 }
```

Arquivos A.6: *Arquivo-fonte*:simulador.cpp

Os Arquivos A.7 e A.8 definem o objeto Lista de Eventos. Nestes são implementados todos os atributos necessários a manutenção da lista de eventos, bem como os métodos que atuam sobre esses atributos.

```
1  /*****************************************************************
2  * TListaEvento.h                                                *
3  * Define a classe TListaEvento.                                 *
4  *                                                               *
5  * Desenvolvido por Tiago Souza Azevedo                          *
6  * Laboratorio de Redes de Alta Velocidade                       *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ  *
8  *                                                               *
9  * Contato: tiago@ravel.ufrj.br                                  *
10 *****************************************************************/
11
12 #ifndef TLISTA_EVENTO_H_
13 #define TLISTA_EVENTO_H_
14
15 #include "TEvento.h"
16 #include "TComputador.h"
17 #include <string>
18 using namespace std;
19
20 //-------------------------------------------------
21 // Estrutura que define os nos da Lista de
22 // Eventos
23 //-------------------------------------------------
24 typedef struct TBox
25 {
26     TEvento *event_ptr;
27     TBox *next_ptr;
28 };
29
30
31
32 //-------------------------------------------------
33 // Classe Lista De Evento
34 //
35 // Representa uma lista encadeada e ordenada de
36 // estruturas TBox, cada uma contendo um ponteiro
37 // para um evento e um ponteiro para a proxima
38 // estrutura TBox. Sera ordena pela propriedade
39 // tempo da classe TEvento.
40 //-------------------------------------------------
41 class TListaEvento
42 {
```

```
43 private:
44     // Sentinela, apontara sempre para o primeiro no da lista
45     TBox *first_ptr;
46     // Quantidade de nos na lista
47     unsigned int count_ev;
48 
49 public:
50     TListaEvento();
51     virtual ~TListaEvento();
52 
53     // Delete todos os eventos de uma classe definida
54     // Retorna o numero de eventos deletados
55     //int deletaEvento(string tipo, TTipoPc tipoPc);
56     // Insere um evento na lista de eventos
57     bool insereEvento(TEvento *evento_ptr);
58     // Retira o primeiro evento da lista retornando como valor da
          funcao
59     TEvento *proximoEvento();
60     // Retorna uma string com todos os eventos da lista
61     string toString();
62 };
63 
64 #endif /*TLISTA_EVENTO_H_*/
```

Arquivos A.7: *Arquivo-fonte*:TListaEvento.h

```
1  /*************************************************************
2  * TListaEvento.cpp                                          *
3  * Implementa a classe TListaEvento.                         *
4  *                                                           *
5  * Desenvolvido por Tiago Souza Azevedo                      *
6  * Laboratorio de Redes de Alta Velocidade                   *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ *
8  *                                                           *
9  * Contato: tiago@ravel.ufrj.br                              *
10 *************************************************************/
11 
12 #include "TListaEvento.h"
13 #include <sstream>
14 #include <iostream>
15 
16 using namespace std;
17 
18 //----------------------------------------
19 TListaEvento::TListaEvento()
20 {
21     first_ptr = NULL;
22     count_ev = 0;
```

```
23 }
24 //----------------------------------------
25 TListaEvento::~TListaEvento()
26 {
27     TBox *atual_ptr,
28          *anterior_ptr;
29     TEvento *evento_atual;
30
31     // Desaloco todos os nos e eventos dentro deles
32     if(count_ev > 0 && first_ptr != NULL)
33     {
34         atual_ptr = first_ptr->next_ptr;
35         anterior_ptr = first_ptr;
36         while(atual_ptr != NULL)
37         {
38             // Deleto o no e seu evento anteior ao atual
39             evento_atual = anterior_ptr->event_ptr;
40             delete evento_atual;
41             delete anterior_ptr;
42
43             // Vou para frente
44             anterior_ptr = atual_ptr;
45             atual_ptr = atual_ptr->next_ptr;
46         }
47         // Neste ponto, sobra o ultimo no
48         evento_atual = anterior_ptr->event_ptr;
49         delete evento_atual;
50         delete anterior_ptr;
51     }
52 }
53 //---------------------------------------
54 bool TListaEvento::insereEvento(TEvento *evento_ptr)
55 {
56
57     // Ponteiro temporario para criacao de um novo box
58     TBox       *box_ptr = new TBox;
59     // Ponteiro que encontrara a posicao na lista em
60     // que sera inserido o novo
61     TBox       *aux_ptr;
62
63     // Inicializo os boxes
64     box_ptr->event_ptr = evento_ptr;
65     box_ptr->next_ptr = NULL;
66
67
68     if(count_ev > 0) // se a lista ja tem eventos
69     {
```

```
70          // se e menor que o primeiro
71          if(evento_ptr->tempo < first_ptr->event_ptr->tempo)
72          {
73              // Insiro na primeira posicao
74              box_ptr->next_ptr = first_ptr;
75              first_ptr = box_ptr;
76          }
77          else
78          {
79              // procuro a posicao ate que alcance o ultimo evento
80              aux_ptr = first_ptr;
81              while((aux_ptr->next_ptr != NULL) && (aux_ptr->next_ptr
                    ->event_ptr->tempo <= evento_ptr->tempo))
82              {
83                  aux_ptr = aux_ptr->next_ptr;
84              }
85
86              // neste ponto sei que o evento a ser inserido sera o
                    proximo
87              box_ptr->next_ptr = aux_ptr->next_ptr;
88              aux_ptr->next_ptr = box_ptr;
89          }
90      }
91      else
92      {
93          // Lista vazia. e o primeiro no
94          first_ptr = box_ptr;
95      }
96      count_ev++;
97
98      return true;
99  }
100 //-------------------------------------
101 TEvento *TListaEvento::proximoEvento()
102 {
103     TEvento *evento_ptr;
104     TBox *box_anterior_ptr;
105
106     evento_ptr = NULL;
107     if(count_ev > 0)
108     {
109         evento_ptr = first_ptr->event_ptr;
110         box_anterior_ptr = first_ptr;
111         first_ptr = first_ptr->next_ptr;
112
113         // Deleto SOMENTE o no que ja passou
114         // pois vou retornar este evento para o simulador
```

```
115         delete box_anterior_ptr;
116         count_ev--;
117     }
118 
119     return evento_ptr;
120 }
121 //---------------------------------------
122 string TListaEvento::toString()
123 {
124     ostringstream strOut(ostringstream::out);
125     TBox *atual_ptr;
126 
127     strOut << "[LISTA] Num Eventos: " << count_ev << endl;
128     atual_ptr = first_ptr;
129     while(atual_ptr != NULL)
130     {
131         strOut << "---" << endl;
132         strOut << atual_ptr->event_ptr->toString();
133         atual_ptr = atual_ptr->next_ptr;
134     }
135 
136     return strOut.str();
137 }
138 //---------------------------------------
```

Arquivos A.8: *Arquivo-fonte*:TListaEvento.cpp

Os Arquivos A.9 e A.10 definem o objeto responsável por realizar a entrada de dados de movimentação dos dispositivos.

```
/*****************************************************************
* TEvFileHandler.h                                               *
* Define a classe TEvFileHandler responsavel pela manipulacao    *
* de arquivos de entrada.                                        *
*                                                                *
* Desenvolvido por Tiago Souza Azevedo                           *
* Laboratorio de Redes de Alta Velocidade                        *
* Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ   *
*                                                                *
* Contato: tiago@ravel.ufrj.br                                   *
*****************************************************************/

#ifndef TFILEHANDLER_H_
#define TFILEHANDLER_H_

#include <string>
#include <vector>
#include <fstream>

using namespace std;

class TFileHandler {

public:

    fstream fHandler;
    string filePath;

    vector<string> dataFields;


    TFileHandler(string path);
    virtual ~TFileHandler();
    bool readNextPos();

};

#endif /* TFILEHANDLER_H_ */
```

Arquivos A.9: *Arquivo-fonte*:TFileHandler.h

```
/*****************************************************************
* TEvFileHandler.cpp                                             *
* Implementa a classe TEvFileHandler.                            *
```

```
4  *                                                                    *
5  * Desenvolvido por Tiago Souza Azevedo                               *
6  * Laboratorio de Redes de Alta Velocidade                            *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ       *
8  *                                                                    *
9  * Contato: tiago@ravel.ufrj.br                                       *
10 **********************************************************************/
11
12 #include "TFileHandler.h"
13 #include "globais.h"
14
15 using namespace std;
16 //---------------------------------------
17 TFileHandler::TFileHandler(string path) {
18
19     //Seto a variavel
20     filePath = path;
21     dataFields.resize(4);
22
23     //abro o arquivo
24     fHandler.open(filePath.c_str(), fstream::in );
25
26     if(!fHandler.is_open())
27     {
28         cout << "Erro ao abrir o arquivo: " << filePath.c_str() <<
               endl;
29         exit(1);
30     }
31
32 }
33 //--------------------------------
34 bool TFileHandler::readNextPos(){
35
36     char buff[256];
37     string line;
38
39     // Se houver algum problema na leitura do arquivo
40     // retorno false e o simulador para
41     if(fHandler.fail()) return false;
42
43     // Se tiver alcancado o fim do arquivo, retorno
44     // true sem tentar ler do arquivo novamente
45     // a partir daqui, nenhum evento de movimento sera adicionado
          novamente
46     if(fHandler.eof()) return true;
47
48     // Se nenhuma das situacoes anteriores for encontrada, prossigo
```

```
        normalmente
49      //obtenho a linha
50      fHandler.getline(buff,256);
51
52      //coloco em uma var string para faciliar manipulacao
53      line = buff;
54      size_t found;
55      size_t pos = 0;
56
57      for (int i=0; i<4;i++)
58      {
59          found = line.find("\t",pos);
60          dataFields.at(i) = line.substr(pos,(found - pos));
61          pos = found + 1;
62      }
63
64      return true;
65  }
66  //-------------------------------
67  TFileHandler::~TFileHandler() {
68      fHandler.close();
69  }
70  //-------------------------------
```

Arquivos A.10: *Arquivo-fonte*:TFileHandler.cpp

Os Arquivos A.11 e A.12 implementam a geração de números aleatórios.

```
/*************************************************************
* TRandomGen.cpp                                             *
* Define a classe TRandomGen responsavel pela geracao        *
* de numero aleatorios.                                      *
*                                                            *
* Desenvolvido por Tiago Souza Azevedo                       *
* Laboratorio de Redes de Alta Velocidade                    *
* Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ *
*                                                            *
* Contato: tiago@ravel.ufrj.br                               *
*************************************************************/

#ifndef TRANDOMGEN_H_
#define TRANDOMGEN_H_

// Define integer types with known size: int32_t, uint32_t,
// int64_t, uint64_t. If this doesn't work then insert
// compiler-specific definitions here:
#if defined(__GNUC__)
  // Compilers supporting C99 or C++0x have inttypes.h
// defining these integer types
  #include <inttypes.h>
  #define INT64_SUPPORTED // Remove this if the compiler
// doesn't support 64-bit integers
#elif defined(_WIN16) || defined(__MSDOS__) || defined(_MSDOS)
   // 16 bit systems use long int for 32 bit integer
  typedef   signed long int int32_t;
  typedef unsigned long int uint32_t;
#elif defined(_MSC_VER)
  // Microsoft have their own definition
  typedef   signed __int32  int32_t;
  typedef unsigned __int32 uint32_t;
  typedef   signed __int64  int64_t;
  typedef unsigned __int64 uint64_t;
  #define INT64_SUPPORTED // Remove this if the compiler doesn't
  //support 64-bit integers
#else
  // This works with most compilers
  typedef signed int          int32_t;
  typedef unsigned int       uint32_t;
  typedef long long           int64_t;
  typedef unsigned long long uint64_t;
  #define INT64_SUPPORTED // Remove this if the compiler doesn't
     support 64-bit integers
```

```
44 #endif
45
46
47 /*****************************************************
48 System-specific user interface functions
49 *****************************************************/
50
51 void EndOfProgram(void); // System-specific exit code (userintf.cpp
       )
52
53 void FatalError(const char *ErrorText);// System-specific error
54
55 #if defined(__cplusplus)// class definitions only in C++
56 /*****************************************************
57 Define random number generator classes
58 *****************************************************/
59
60 class TRandomGen {                // Encapsulate random number
       generator
61 // Choose which version of Mersenne Twister you want:
62 #if 0
63 // Define constants for type MT11213A:
64 #define MERS_N   351
65 #define MERS_M   175
66 #define MERS_R   19
67 #define MERS_U   11
68 #define MERS_S   7
69 #define MERS_T   15
70 #define MERS_L   17
71 #define MERS_A   0xE4BD75F5
72 #define MERS_B   0x655E5280
73 #define MERS_C   0xFFD58000
74 #else
75 // or constants for type MT19937:
76 #define MERS_N   624
77 #define MERS_M   397
78 #define MERS_R   31
79 #define MERS_U   11
80 #define MERS_S   7
81 #define MERS_T   15
82 #define MERS_L   18
83 #define MERS_A   0x9908B0DF
84 #define MERS_B   0x9D2C5680
85 #define MERS_C   0xEFC60000
86 #endif
87
88 public:
```

```
89      TRandomGen(int seed) {          // Constructor
90        RandomInit(seed); LastInterval = 0;}
91     void RandomInit(int seed);          // Re-seed
92     void RandomInitByArray(int const seeds[], int NumSeeds); // Seed
           by more than 32 bits
93     int IRandom (int min, int max);     // Output random integer
94     int IRandomX(int min, int max);     // Output random integer,
          exact
95     double Random();                    // Output random float
96     uint32_t BRandom();                 // Output random bits
97     double amostraExponecial(double lambda);
98 private:
99     void Init0(int seed);               // Basic initialization
          procedure
100    uint32_t mt[MERS_N];                // State vector
101    int mti;                            // Index into mt
102    uint32_t LastInterval;              // Last interval length for
          IRandomX
103    uint32_t RLimit;                    // Rejection limit used by
          IRandomX
104 };
105
106
107
108 #endif // __cplusplus
109
110 #endif /*TRANDOMGEN_H_*/
```

Arquivos A.11: *Arquivo-fonte*:TRandomGen.h

```
1  /*************************************************************
2  * TRandomGen.cpp                                            *
3  * Implementa a classe TRandomGen.                           *
4  *                                                           *
5  * Desenvolvido por Tiago Souza Azevedo                      *
6  * Laboratorio de Redes de Alta Velocidade                   *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ *
8  *                                                           *
9  * Contato: tiago@ravel.ufrj.br                              *
10 *************************************************************/
11
12
13 #include "TRandomGen.h"
14 #include <cmath>
15
16 void TRandomGen::Init0(int seed) {
17    // Seed generator
18    const uint32_t factor = 1812433253UL;
```

```
19     mt[0]= seed;
20     for (mti=1; mti < MERS_N; mti++) {
21        mt[mti] = (factor * (mt[mti-1] ^ (mt[mti-1] >> 30)) + mti);
22     }
23  }
24  //-----------------------------------------
25  void TRandomGen::RandomInit(int seed) {
26     // Initialize and seed
27     Init0(seed);
28
29     // Randomize some more
30     for (int i = 0; i < 37; i++) BRandom();
31  }
32  //-----------------------------------------
33  void TRandomGen::RandomInitByArray(int const seeds[], int NumSeeds)
        {
34     // Seed by more than 32 bits
35     int i, j, k;
36
37     // Initialize
38     Init0(19650218);
39
40     if (NumSeeds <= 0) return;
41
42     // Randomize mt[] using whole seeds[] array
43     i = 1;  j = 0;
44     k = (MERS_N > NumSeeds ? MERS_N : NumSeeds);
45     for (; k; k--) {
46        mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >> 30)) * 1664525UL)) +
              (uint32_t)seeds[j] + j;
47        i++; j++;
48        if (i >= MERS_N) {mt[0] = mt[MERS_N-1]; i=1;}
49        if (j >= NumSeeds) j=0;}
50     for (k = MERS_N-1; k; k--) {
51        mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >> 30)) * 1566083941UL)
             ) - i;
52        if (++i >= MERS_N) {mt[0] = mt[MERS_N-1]; i=1;}}
53     mt[0] = 0x80000000UL;  // MSB is 1; assuring non-zero initial
          array
54
55     // Randomize some more
56     mti = 0;
57     for (int i = 0; i <= MERS_N; i++) BRandom();
58  }
59  //-----------------------------------------
60  uint32_t TRandomGen::BRandom() {
61     // Generate 32 random bits
```

```
62    uint32_t y;
63
64    if (mti >= MERS_N) {
65       // Generate MERS_N words at one time
66       const uint32_t LOWER_MASK = (1LU << MERS_R) - 1;
67       // Lower MERS_R bits
68       const uint32_t UPPER_MASK = 0xFFFFFFFF << MERS_R;
69       // Upper (32 - MERS_R) bits
70       static const uint32_t mag01[2] = {0, MERS_A};
71
72       int kk;
73       for (kk=0; kk < MERS_N-MERS_M; kk++) {
74          y = (mt[kk] & UPPER_MASK) | (mt[kk+1] & LOWER_MASK);
75          mt[kk] = mt[kk+MERS_M] ^ (y >> 1) ^ mag01[y & 1];}
76
77       for (; kk < MERS_N-1; kk++) {
78          y = (mt[kk] & UPPER_MASK) | (mt[kk+1] & LOWER_MASK);
79          mt[kk] = mt[kk+(MERS_M-MERS_N)] ^ (y >> 1) ^ mag01[y &
               1];}
80
81       y = (mt[MERS_N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
82       mt[MERS_N-1] = mt[MERS_M-1] ^ (y >> 1) ^ mag01[y & 1];
83       mti = 0;
84    }
85    y = mt[mti++];
86
87    // Tempering (May be omitted):
88    y ^=  y >> MERS_U;
89    y ^= (y << MERS_S) & MERS_B;
90    y ^= (y << MERS_T) & MERS_C;
91    y ^=  y >> MERS_L;
92
93    return y;
94 }
95 //---------------------------------------------------------------------
96 double TRandomGen::Random() {
97    // Output random float number in the interval 0 <= x < 1
98    // Multiply by 2^(-32)
99    return (double)BRandom() * (1./(65536.*65536.));
100 }
101 //---------------------------------------------------------------------
102 int TRandomGen::IRandom(int min, int max) {
103    // Output random integer in the interval min <= x <= max
104    // Relative error on frequencies < 2^-32
105    if (max <= min) {
106       if (max == min) return min; else return 0x80000000;
107    }
```

```
108     // Multiply interval with random and truncate
109     int r = int((double)(uint32_t)(max - min + 1) * Random() + min);
110     if (r > max) r = max;
111     return r;
112 }
113 //-------------------------------------------------------------------
114 int TRandomGen::IRandomX(int min, int max) {
115     // Output random integer in the interval min <= x <= max
116     // Each output value has exactly the same probability.
117     // This is obtained by rejecting certain bit values so that the
118      //number of possible bit values is divisible by the interval
119      // length
120     if (max <= min) {
121        if (max == min) return min; else return 0x80000000;
122     }
123 #ifdef  INT64_SUPPORTED
124     // 64 bit integers available. Use multiply and shift method
125     uint32_t interval;                    // Length of interval
126     uint64_t longran;                     // Random bits * interval
127     uint32_t iran;                        // Longran / 2^32
128     uint32_t remainder;                   // Longran % 2^32
129 
130     interval = uint32_t(max - min + 1);
131     if (interval != LastInterval) {
132        // Interval length has changed. Must calculate rejection
           limit
133        // Reject when remainder >= 2^32 / interval * interval
134        // RLimit will be 0 if interval is a power of 2. No rejection
            then
135        RLimit = uint32_t(((uint64_t)1 << 32) / interval) * interval
           - 1;
136        LastInterval = interval;
137     }
138     do { // Rejection loop
139        longran  = (uint64_t)BRandom() * interval;
140        iran = (uint32_t)(longran >> 32);
141        remainder = (uint32_t)longran;
142     } while (remainder > RLimit);
143     // Convert back to signed and return result
144     return (int32_t)iran + min;
145 
146 #else
147     // 64 bit integers not available. Use modulo method
148     uint32_t interval;                    // Length of interval
149     uint32_t bran;                        // Random bits
150     uint32_t iran;                        // bran / interval
151     uint32_t remainder;                   // bran % interval
```

```
152
153     interval = uint32_t(max - min + 1);
154     if (interval != LastInterval) {
155        // Interval length has changed. Must calculate rejection
              limit
156        // Reject when iran = 2^32 / interval
157        // We can't make 2^32 so we use 2^32-1 and correct afterwards
158        RLimit = (uint32_t)0xFFFFFFFF / interval;
159        if ((uint32_t)0xFFFFFFFF % interval == interval - 1) RLimit
              ++;
160     }
161     do { // Rejection loop
162        bran = BRandom();
163        iran = bran / interval;
164        remainder = bran % interval;
165     } while (iran >= RLimit);
166     // Convert back to signed and return result
167     return (int32_t)remainder + min;
168
169 #endif
170 }
171 //-----------------------------------------------
172 double TRandomGen::amostraExponecial(double lambda)
173 {
174     double u0;
175
176     u0 = Random();
177     return -1.0 * (log(u0) / lambda);
178 }
179 //-----------------------------------------------
```

Arquivos A.12: *Arquivo-fonte*:TRandomGen.cpp

Os Arquivos A.13 e A.14 implementam a classe abstrata de eventos. A partir dessa classe, serão derivadas novas classes para cada tipo de evento.

```
/*****************************************************************
* TEvento.cpp                                                    *
* Modela a classe TEvento. Classe abstrata que define todos     *
* os eventos                                                     *
*                                                                *
* Desenvolvido por Tiago Souza Azevedo                           *
* Laboratorio de Redes de Alta Velocidade                        *
* Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ   *
*                                                                *
* Contato: tiago@ravel.ufrj.br                                   *
*****************************************************************/


#ifndef TEVENTO_H_
#define TEVENTO_H_

#include <string>
using namespace std;

#include "TComputador.h"

/*****************************************
 * Classe Evento (Abstrata)
 *
 * Classe pai para os eventos especificos
 * Todos os eventos herdam desta classe
 *****************************************/
class TEvento
{
protected:
    string versaoStr;
    virtual void converteStr();

public:
    // Tempo de ocorrencia do evento. Em segundos
    double tempo;
    // Tipo do evento. E o nome da classe
    string tipo;
    // Local de ocorrencia do evento.
    TComputador *pc;

    TEvento(double tempoOcorrencia, TComputador *comp);
    TEvento(double tempoOcorrencia);
```

```
44      virtual ~TEvento();
45
46      // Metodo tratador do evento. Nesta classe base ele e virtual
            puro
47      // e portanto, deve ser implementado nas classes derivadas
48      virtual bool ocorre() = 0;
49      // Retorna uma string com informacoes sobre o evento
50      virtual string toString();
51  };
52
53  #endif /*TEVENTO_H_*/
```

Arquivos A.13: *Arquivo-fonte*:TEvento.h

```
1   /*************************************************************
2   * TEvento.cpp                                                *
3   * Implementa a classe TEvento.                               *
4   *                                                            *
5   * Desenvolvido por Tiago Souza Azevedo                       *
6   * Laboratorio de Redes de Alta Velocidade                    *
7   * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ  *
8   *                                                            *
9   * Contato: tiago@ravel.ufrj.br                               *
10  *************************************************************/
11
12  #include "TEvento.h"
13  #include <sstream>
14  #include <iostream>
15  #include <cstring>
16  #include "globais.h"
17
18  //-----------------------------------------------------------
19  TEvento::TEvento(double tempoOcorrencia, TComputador *comp)
20  {
21      tipo = "TEvento";
22      tempo = tempoOcorrencia;
23      pc = comp;
24      versaoStr = "";
25  }
26  //-----------------------------------------------------------
27  TEvento::TEvento(double tempoOcorrencia)
28  {
29      tipo = "TEvento";
30      tempo = tempoOcorrencia;
31      pc = NULL;
32      versaoStr = "";
33  }
34  //-----------------------------------------------------------
```

```
35 TEvento::~TEvento()
36 {
37 }
38 //---------------------------------------------------------------
39 void TEvento::converteStr()
40 {
41     ostringstream strOut(ostringstream::out);
42 
43     strOut << "[" << tipo << "] ";
44     if(pc != NULL) strOut << "PC" << pc->id << endl;
45     strOut << "tempo: " << tempo << " milisegundos" << endl;
46 
47     // Lembrar sempre de concatenar
48     versaoStr = strOut.str();
49 }
50 //---------------------------------------------------------------
51 string TEvento::toString()
52 {
53     converteStr();
54     return versaoStr;
55 }
56 //---------------------------------------------------------------
```

Arquivos A.14: *Arquivo-fonte*:TEvento.cpp

Os Arquivos A.15 e A.16 implementam a classe de dispositivos. Nestes estão implementados os atributos e métodos necessários para representar um nó na simulação.

```
1  /*****************************************************************
2  * TComputador.h                                                 *
3  * Define a classe TComputador. Modela os dispositivos estudados.*
4  *                                                               *
5  * Desenvolvido por Tiago Souza Azevedo                          *
6  * Laboratorio de Redes de Alta Velocidade                       *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ  *
8  *                                                               *
9  * Contato: tiago@ravel.ufrj.br                                  *
10 *****************************************************************/
11
12 #ifndef TCOMPUTADOR_H_
13 #define TCOMPUTADOR_H_
14
15 #include <cstring>
16 #include <string>
17 #include <vector>
18 using namespace std;
19
20 class TComputador
21 {
22 public:
23
24     // Identifica qual e este pc
25     int id;
26
27     // Intervalo de tempo entre probes
28     float intProbe;
29
30     //Potencia maxima do sinal transmissao
31     float potMaxSinalTx;
32
33     //Potencia efetiva do sinal transmissao
34     float potEfetSinalTx;
35
36     //Numero de probes transmitidos
37     double numProbeTx;
38
39     //Raio maximo, obtido atraves da utilizacao da
40     // potencia maxima do sinal de tx e da equacao de Friss
41     double raioMax;
42
```

```
43      //Raio Efetivo
44      double raioEfet;
45
46      //Consumo
47      double consumo;
48
49      //Posicao no cenario
50      float posX;
51      float posY;
52
53      // Armazena informacoes sobre contato
54      // fisico para cada dispositivo
55      vector<float> contFisico;
56
57      // Armazena informacoes sobre contato
58      // logico para cada dispositivo
59      vector<float> contLogico;
60
61      // Armazena informacoes sobre status
62      // do contato fisico para cada dispositivo
63      vector<bool> statusContFisico;
64
65      // Armazena informacoes sobre status do
66      // contato fisico para cada dispositivo
67      vector<bool> statusContLogico;
68
69      // Armazena informacoes sobre o instante
70      // de inicio do contato
71      vector<float> inicioContFisico;
72
73      // Armazena informacoes sobre intervalo de contato
74      vector<float> intContFisico;
75      vector<float> inEmpContFisico;
76
77      float mediaIntContFisico;
78
79      // Armazena a probabilidade de perder uma
80      // oportunidade de contato para cada par
81      vector<float> probPerdaPorNo;
82
83      //Probabilidade de perder media total do no
84      float probPerdaM;
85
86      //Ultimo contato foi descoberto
87      bool lastCont;
88
89      //Acrescimo a ser usado no AIMD
```

```
90      float potStep;
91
92      //Guarda a media do intervalo de contato
93      float mediaContFis;
94
95      TComputador(int idp, float intProbep, float potMaxSinalTxp,
            float potEfetSinalTxp, float posXp, float posYp);
96      TComputador();
97      virtual ~TComputador();
98
99      //Algoritmos de ajuste da Potencia do Sinal de Tx
100     void ExecutaAlgoritmo();
101     string toString();
102 };
103
104 #endif /*TCOMPUTADOR_H_*/
```

Arquivos A.15: *Arquivo-fonte*:TComputador.h

```
1  /*************************************************************
2  * TComputador.cpp                                           *
3  * Implementa a classe TComputador.                          *
4  *                                                           *
5  * Desenvolvido por Tiago Souza Azevedo                      *
6  * Laboratorio de Redes de Alta Velocidade                   *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ  *
8  *                                                           *
9  * Contato: tiago@ravel.ufrj.br                              *
10 *************************************************************/
11
12 #include "TComputador.h"
13 #include "globais.h"
14 #include "constantes.h"
15 #include <sstream>
16 #include <cmath>
17
18 using namespace std;
19
20 //-----------------------------------------------
21 TComputador::TComputador(int idp, float intProbep, float
       potMaxSinalTxp, float potEfetSinalTxp, float posXp, float posYp)
22 {
23     // Inicializando os dados do dispositivo
24     id = idp;
25     intProbe = intProbep;
26     potMaxSinalTx = potMaxSinalTxp;
27     mediaContFis = 0.0;
28
```

```
29
30      potEfetSinalTx = potEfetSinalTxp;
31
32      posX = posXp;
33      posY = posYp;
34      potStep = 0.1;
35      lastCont = false;
36
37      //Calculando o Rmax e Refet(em metros) do dispositivo
38      raioMax = (LAMBDA/(4*PI))*sqrt(potMaxSinalTxp/RSENSITIVITY);
39      raioEfet = (LAMBDA/(4*PI))*sqrt((potEfetSinalTxp *
            potMaxSinalTxp)/RSENSITIVITY);
40
41      // Zerando contatores de contato
42      contFisico.assign(numNodes,0);
43      contLogico.assign(numNodes,0);
44
45      statusContFisico.assign(numNodes,false);
46      statusContLogico.assign(numNodes,false);
47      inicioContFisico.assign(numNodes,0.0);
48      probPerdaPorNo.assign(numNodes,0.0);
49
50      probPerdaM = 0.0;
51
52      // Zerando contadores de consumo
53      numProbeTx = 0;
54      consumo = 0;
55
56      //Serve para amostrar a duracao de um contato
57      intContFisico.assign(numNodes,0.0);
58      inEmpContFisico.assign(numNodes,0.0);
59      mediaIntContFisico = 0.0;
60
61
62  }
63  //-------------------------------------------------
64  TComputador::TComputador()
65  {
66
67  }
68
69  //-------------------------------------------------
70  TComputador::~TComputador()
71  {
72  }
73  //-------------------------------------------------
74  string TComputador::toString()
```

```
75 {
76     ostringstream strOut(ostringstream::out);
77
78     strOut << "[Node " << id << "]" << endl;
79     strOut << "posX: " << posX << endl;
80     strOut << "posY: " << posY << endl;
81     strOut << "potMaxSinalTx: " << potMaxSinalTx << endl;
82     strOut << "potEfetSinalTx: " << potEfetSinalTx << endl;
83     strOut << "raioMax: " << raioMax << endl;
84     strOut << "raioEfet: " << raioEfet << endl;
85     strOut << "intProbe: " << intProbe << endl;
86     strOut << "numProbeTx: " << numProbeTx << endl;
87     strOut << "consumo: " << consumo << endl;
88     strOut << "End: " << this << endl;
89     strOut << "-----------------------------------" << endl;
90
91     return strOut.str();
92 }
93 //---------------------------------------------------
94 void TComputador::ExecutaAlgoritmo()
95 {
96     float media = 0.0;
97     int count = 0.0;
98     vector<float> angle;
99     float maiorAngle = 0.0;
100    float pot = 0.0;
101    float alpha = 0.5;
102
103    float pointA, pointB, pointC, distA, distB, distC;
104
105    switch(algId)
106    {
107        case 1: // AIMD
108
109            if(this->lastCont)
110            {
111                this->potEfetSinalTx = this->potEfetSinalTx / 2;
112                if(this->potEfetSinalTx < MINEFT) this->
                       potEfetSinalTx = MINEFT;
113            }
114            else
115            {
116                this->potEfetSinalTx = this->potEfetSinalTx +
                       potStep;
117                if(this->potEfetSinalTx > MAXEFT) this->
                       potEfetSinalTx = MAXEFT;
118            }
```

```
119             break;
120 
121         case 2: // Knee - Tenta encontrar a potencia efetiva
122                 // mais proxima do joelho da curva de probabilidade
123 
124             //Descubro o tamanho medio do intervalo de contato
125             for(int i = 0; i < numNodes; i++)
126             {
127                 if(this->intContFisico[i] != 0)
128                 {
129                     media += this->intContFisico[i];
130                     count++;
131                 }
132 
133             }
134             if(count !=0) media = media / count;
135             else break;
136 
137             this->mediaContFis = (this->mediaContFis * alpha) + (
                    media * (1-alpha));
138             if((this->id == 2) && (outType == 3)) cout <<
                    tempoAtualSim << " \t " << this->mediaContFis <<
                    endl;
139 
140 
141             //Calculo o angulo de cada ponto da curva e encontro o
                    maior
142             angle.assign(AMOSTRAS,0.0);
143 
144             for(int i=1 ; i<AMOSTRAS; i++)
145             {
146                 //gero os pontos da fronteira de pareto
147                 if(media == 0) media = 1;
148                 pointA = funcProb(i*STEP,(1/this->mediaContFis),
                        intProbe);
149                 pointB = funcProb((i-1)*STEP,(1/this->mediaContFis)
                        , intProbe);
150                 pointC = funcProb((i+1)*STEP,(1/this->mediaContFis)
                        , intProbe);
151 
152                 //calculo o angulo entre estes pontos
153                 distA = calcDistancia((i-1)*STEP, pointB, (i+1)*
                        STEP,pointC);
154                 distB = calcDistancia(i*STEP,pointA, (i+1)*STEP,
                        pointC);
155                 distC = calcDistancia(i*STEP,pointA, (i-1)*STEP,
                        pointB);
```

```
156
157                 angle[i] = 360 - (acos((pow(distC,2.0) + pow(distB
                        ,2.0) - pow(distA,2.0))/(2*distB*distC)) *
                        (180.0/PI));
158                 if(angle[i] > maiorAngle)
159                 {
160                     maiorAngle = angle[i];
161                     pot = i*STEP;
162                 }
163             }
164
165             this->potEfetSinalTx = pot;
166             if(this->potEfetSinalTx < MINEFT) this->potEfetSinalTx
                    = MINEFT;
167
168             break;
169
170         case 4: // MIAD
171
172             if(this->lastCont)
173             {
174                 this->potEfetSinalTx = this->potEfetSinalTx -
                        potStep;
175                 if (this->potEfetSinalTx < MINEFT) this->
                        potEfetSinalTx = MINEFT;
176             }
177             else
178             {
179                 this->potEfetSinalTx = this->potEfetSinalTx * 2.0;
180                 if(this->potEfetSinalTx > MAXEFT) this->
                        potEfetSinalTx = MAXEFT;
181             }
182             break;
183
184         case 3: // Knee Pareto
185         {
186             float n = 1.0;
187             float k = 1.7;
188             float p = 0.1;
189             float m = 27.0;
190             float t = 0.0;
191             float a, b, c, aa, bb, cc;
192             double res = 0.0;
193             double res1=0.0;
194
195             a = (1-n)*pow((m/(n-1)),k);
196             b = (n+1)*pow((m/(n+1)),k);
```

```
197             c = (2*n)*pow((m/n),k);
198
199             aa = (pow(m,k)*pow(n,-k)*pow((n-1),-k)*(n*pow((n-1),k)+
                    pow(n,k)*(1-n)))/t*(1-k);
200             bb = (pow(m,k)*pow(n,-k)*pow((n+1),-k)*(n*pow((n+1),k)-
                    pow(n,k)*(n+1)))/t*(k-1);
201             cc = 1-sqrt(p);
202
203             res1 = ((a - b + c)+(k/t)*(aa-bb)) * cc;
204             while(sqrt(pow(res1,2.0)) > EPS)
205             {
206                 res+= res1;
207                 n++;
208
209                 a = (1-n)*pow((m/(n-1)),k);
210                 b = (n+1)*pow((m/(n+1)),k);
211                 c = (2*n)*pow((m/n),k);
212
213                 aa = (pow(m,k)*pow(n,-k)*pow((n-1),-k)*(n*pow((n-1)
                        ,k)+pow(n,k)*(1-n)))/t*(1-k);
214                 bb = (pow(m,k)*pow(n,-k)*pow((n+1),-k)*(n*pow((n+1)
                        ,k)-pow(n,k)*(n+1)))/t*(k-1);
215                 cc = 1-sqrt(p);
216
217                 res1 = ((a - b + c)+(k/t)*(aa-bb)) * cc;
218             }
219             this->potEfetSinalTx = res;
220             if(this->potEfetSinalTx < MINEFT) this->potEfetSinalTx
                    = MINEFT;
221
222             break;
223         }
224     }
225
226     //Atualizo a intensidade do sinal de transmissao
227     raioEfet = (LAMBDA/(4*PI))*sqrt((this->potEfetSinalTx * this->
            potMaxSinalTx)/RSENSITIVITY);
228
229
230 }
```

Arquivos A.16: *Arquivo-fonte*:TComputador.cpp

Os Arquivos A.17 e A.18 implementam o evento responsável pela movimentação dos dispositivos. Essa movimentação foi gerada pelo gerador de mobilidade *Mobisim*[68] e passada como parâmetro de entrada para o simulador.

```
1  /*****************************************************************
2  * TEvMoving.cpp                                                 *
3  * Define a classe TEvMoving responsavel pela movimentacao       *
4  * dos dispositivos             .                                *
5  *                                                               *
6  * Desenvolvido por Tiago Souza Azevedo                          *
7  * Laboratorio de Redes de Alta Velocidade                       *
8  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ  *
9  *                                                               *
10 * Contato: tiago@ravel.ufrj.br                                 *
11 *****************************************************************/
12
13 #ifndef TEVMOVING_H_
14 #define TEVMOVING_H_
15
16 #include "TEvento.h"
17
18 class TEvMoving: public TEvento {
19 public:
20     TEvMoving(double tempoOcorrencia);
21     virtual ~TEvMoving();
22     bool ocorre();
23 };
24
25 #endif /* TEVMOVING_H_ */
```

Arquivos A.17: *Arquivo-fonte*:TEvMoving.h

```
1  /*****************************************************************
2  * TEvMoving.cpp                                                 *
3  * Implementa a classe TEvMoving.                                *
4  *                                                               *
5  * Desenvolvido por Tiago Souza Azevedo                          *
6  * Laboratorio de Redes de Alta Velocidade                       *
7  * Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ  *
8  *                                                               *
9  * Contato: tiago@ravel.ufrj.br                                  *
10 *****************************************************************/
11
12 #include "TEvMoving.h"
13 #include "TEvCheckCont.h"
14 #include "globais.h"
15 #include "constantes.h"
```

```
16 #include <cmath>
17
18 using namespace std;
19
20 //------------------------------------------------
21 TEvMoving::TEvMoving(double tempoOcorrencia)
22 : TEvento(tempoOcorrencia)
23 {
24     tipo = "TEvMoving";
25 }
26 //------------------------------------------------
27 TEvMoving::~TEvMoving() {
28 }
29 //------------------------------------------------
30 bool TEvMoving::ocorre() {
31
32     // executo o movimento do primeir no, cuja posicao foi lida
        anteriormente
33     simNodes[atoi(fileMov->dataFields[IDFIELD].c_str())]->posX =
        atof(fileMov->dataFields[XFIELD].c_str());
34     simNodes[atoi(fileMov->dataFields[IDFIELD].c_str())]->posY =
        atof(fileMov->dataFields[YFIELD].c_str());
35
36     //Executo o movimento dos demais nos
37     for(int i=0; i<numNodes-1; i++)
38     {
39         if(!fileMov->readNextPos())
40         {
41             cout << "Erro ao acessar arquivo!!!" << endl;
42             return false;
43         }
44
45         if(!fileMov->fHandler.eof())
46         {
47             double time = atof(fileMov->dataFields[TFIELD].c_str())
                ;
48             if(time > tempo)
49             {
50                 cout << "Erro no tempo do evento!" << endl;
51                 return false;
52             }
53             simNodes[atoi(fileMov->dataFields[IDFIELD].c_str())]->
                posX = atof(fileMov->dataFields[XFIELD].c_str());
54             simNodes[atoi(fileMov->dataFields[IDFIELD].c_str())]->
                posY = atof(fileMov->dataFields[YFIELD].c_str());
55         }
56     }
```

```
57
58      // Ao final deste evento, devo iniciar outro que sera
            responsavel por checar
59      // os contatos fisicos entre os nos
60      TEvCheckCont *evchk;
61      evchk = new TEvCheckCont(tempoAtualSim);
62      if(!listaEventos->insereEvento(evchk))
63      {
64          cout << "Erro ao inserir evento na lista!" << endl;
65          if(evchk != NULL) delete evchk;
66          return false;
67      }
68
69      //Tambem devo inserir na lista de eventos, o  proximo evento de
            movimentacao
70      //cout << "Lendo novos dados do arquivo" << endl;
71      if(!fileMov->readNextPos())
72      {
73          cout << "Erro ao acessar arquivo!!!" << endl;
74          return false;
75      }
76
77      if(!fileMov->fHandler.eof())
78      {
79          TEvMoving *ev;
80          double time = atof(fileMov->dataFields[TFIELD].c_str());
81          ev = new TEvMoving(time);
82          if(!listaEventos->insereEvento(ev))
83          {
84              cout << "Erro ao inserir evento na lista!" << endl;
85              if(ev != NULL) delete ev;
86              return false;
87          }
88      }
89
90      return true;
91
92  }
93  //-----------------------------------------------
```

Arquivos A.18: *Arquivo-fonte*:TEvMoving.cpp

Os Arquivos A.19 e A.20 implementam os eventos de *probe*, executados pelos dispositivos.

```
/*****************************************************************
* TEvProbing.h                                                   *
* Define a classe TEvProbe, que modela o envio de probes         *
* por cada dispositivo                                           *
*                                                                *
* Desenvolvido por Tiago Souza Azevedo                           *
* Laboratorio de Redes de Alta Velocidade                        *
* Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ   *
*                                                                *
* Contato: tiago@ravel.ufrj.br                                   *
*****************************************************************/

#ifndef TEVPROBE_H_
#define TEVPROBE_H_

#include "TEvento.h"

class TEvProbe: public TEvento {
public:
    TEvProbe(double tempoOcorrencia, TComputador *comp);
    virtual ~TEvProbe();
    bool ocorre();
};

#endif /* TEVPROBE_H_ */
```

Arquivos A.19: *Arquivo-fonte*:TEvProbe.h

```
/*****************************************************************
* TEvProbe.cpp                                                   *
* Implementa a classe TEvProbing.                                *
*                                                                *
* Desenvolvido por Tiago Souza Azevedo                           *
* Laboratorio de Redes de Alta Velocidade                        *
* Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ   *
*                                                                *
* Contato: tiago@ravel.ufrj.br                                   *
*****************************************************************/
#include "TEvProbe.h"
#include "globais.h"

using namespace std;
//-------------------------------------------------------------
TEvProbe::TEvProbe(double tempoOcorrencia, TComputador *comp)
```

```
17 : TEvento(tempoOcorrencia, comp)
18 {
19     tipo = "TEvProbe";
20     //pc = comp;
21 }
22 //---------------------------------------------------------------
23 TEvProbe::~TEvProbe()
24 {
25 }
26 //---------------------------------------------------------------
27 bool TEvProbe::ocorre() {
28     // Incremento a contagem de probes
29     simNodes[pc->id]->numProbeTx++;
30 
31     //Executo o algoritmo para escolha da potencia de transmissao
32     pc->ExecutaAlgoritmo();
33 
34     //Efetuo o calculo do consumo
35     pc->consumo += (pc->potEfetSinalTx * pc->potMaxSinalTx);
36 
37 
38     // Executo o probe
39     // Verifico a distancia do no que realiza o probe a todos os
       demais nos
40     // Caso a distancia seja menor que raio efet, os dispositivos
       estao em contato logico
41     for(int i=0; i<numNodes; i++)
42     {
43 
44         if(i != pc->id)
45         {
46             float dist = calcDistancia(pc->posX, pc->posY, simNodes
                   [i]->posX, simNodes[i]->posY);
47             //Contato logico
48             if(dist <= pc->raioEfet)
49             {
50 
51                 if(!pc->statusContLogico[i])//se anteriormente era
                       false, e um novo contato
52                 {
53                     pc->statusContLogico[i] = true;
54                     pc->contLogico[i]++;
55                     pc->inEmpContFisico[i] = tempoAtualSim;
56                 }
57             }
58             else
59             {
```

```
60                  //se anteriormente era verdadeiro, o contato acabou
61                  if(pc->statusContLogico[i])
62                  {
63                      //guardo o tempo aproximado do contato
64                      pc->intContFisico[i] = tempoAtualSim - pc->
                            inEmpContFisico[i];
65                      pc->statusContLogico[i] = false;
66                  }
67              }
68          }
69      }
70      pc->lastCont = false;
71      for(int k=0; k<numNodes; k++)
72          if((pc->statusContLogico[k]) && (pc->id != simNodes[k]->id)
                ) pc->lastCont = true;
73
74
75      if(!fileMov->fHandler.eof())
76      {
77          TEvProbe *ev;
78          ev = new TEvProbe(tempoAtualSim + pc->intProbe, pc);
79          if(!listaEventos->insereEvento(ev))
80          {
81              delete ev;
82              return false;
83          }
84      }
85      return true;
86  }
87  //---------------------------------------------------------------
```

Arquivos A.20: *Arquivo-fonte*:TEvProbe.cpp

Os Arquivos A.21 e A.22 implementam o evento responsável por verificar quando os dispositivos estão em contato, contabilizando estes intervalos.

```
/*****************************************************************
* TEvCheckCont.h                                                 *
* Define a classe TEvCheckCont. Modela o evento de checagem      *
* de contatos                                                    *
*                                                                *
* Desenvolvido por Tiago Souza Azevedo                           *
* Laboratorio de Redes de Alta Velocidade                        *
* Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ   *
*                                                                *
* Contato: tiago@ravel.ufrj.br                                   *
*****************************************************************/


#ifndef TEVCHECKCONT_H_
#define TEVCHECKCONT_H_

#include "TEvento.h"

class TEvCheckCont: public TEvento {
public:
    TEvCheckCont(double tempoOcorrencia);
    virtual ~TEvCheckCont();
    bool ocorre();
};

#endif /* TEVCHECKCONT_H_ */
```

Arquivos A.21: *Arquivo-fonte*:TEvCheckCont.h

```
/*****************************************************************
* TEvCheckCont.cpp                                               *
* Implementa a classe TEvCheckCont.                              *
*                                                                *
* Desenvolvido por Tiago Souza Azevedo                           *
* Laboratorio de Redes de Alta Velocidade                        *
* Programa de Engenharia de Sistemas e Computacao - COPPE/UFRJ   *
*                                                                *
* Contato: tiago@ravel.ufrj.br                                   *
*****************************************************************/

#include "TEvCheckCont.h"
#include "globais.h"
#include "constantes.h"

```

```
16 using namespace std;
17
18 //-------------------------------------------------
19 TEvCheckCont::TEvCheckCont(double tempoOcorrencia)
20 : TEvento(tempoOcorrencia)
21 {
22     tipo = "TEvCheckCont";
23 }
24 //-------------------------------------------------
25 TEvCheckCont::~TEvCheckCont()
26 {
27 }
28 //-------------------------------------------------
29 bool TEvCheckCont::ocorre()
30 {
31     for(int j=0; j<numNodes; j++)
32     {
33         for(int i=0; i<numNodes; i++)
34         {
35             // Quando um no i esta em contato com j, entao j tb
                esta em contato com i
36             // Portanto preciso apenas verificar uma parte da
                tabela
37             if(i < j)
38             {
39                 float dist = calcDistancia(simNodes[j]->posX,
                    simNodes[j]->posY, simNodes[i]->posX, simNodes[i
                    ]->posY);
40                 //cout << "dist de(moving) " << i << "  para  "  <<
                     simNodes[j]->id << ": " << dist << endl;
41
42                 //Contato fisico
43                 if(dist < simNodes[j]->raioMax)
44                 {
45                     if(!simNodes[j]->statusContFisico[i])//se
                        anteriormente era false, e um novo contato
46                     {
47                         simNodes[j]->statusContFisico[i] = true;
48                         simNodes[i]->statusContFisico[j] = true;
49                         simNodes[j]->contFisico[i]++;
50                         simNodes[i]->contFisico[j]++;
51                         simNodes[j]->inicioContFisico[i] =
                            tempoAtualSim;
52                         simNodes[i]->inicioContFisico[j] =
                            tempoAtualSim;
53                     }
54                 }
```

```
55                  else
56                  {
57                      if(simNodes[j]->statusContFisico[i])//se
                            estavam em contato
58                      {
59                          int intContatoFisico;
60                          intContatoFisico = (int)(tempoAtualSim -
                                simNodes[j]->inicioContFisico[i]);
61                          //cout << "tempoAtualSim: " <<
                                tempoAtualSim <<  " pc->inicioContFisico
                                [i]: " <<  pc->inicioContFisico[i] <<
                                endl;
62                          //cout << "intContatoFisico: " <<
                                intContatoFisico << endl;
63                          if(intContatoFisico >= rangeTempoContato)
64                          {
65                              tempoContato.resize(intContatoFisico
                                    +1,0.0);
66                              rangeTempoContato = intContatoFisico+1;
67                          }
68                          tempoContato[intContatoFisico]++;
69                          //cout << "tempoContato[" <<
                                intContatoFisico << "]: " <<
                                tempoContato[intContatoFisico] << endl;
70                      }
71
72                      simNodes[j]->statusContFisico[i] = false;
73                      simNodes[i]->statusContFisico[j] = false;
74                  }
75              }
76          }
77      }
78
79      return true;
80  }
81  //-----------------------------------------------
```

Arquivos A.22: *Arquivo-fonte*:TEvCheckCont.cpp