Fernanda Paixão Franciscani

Algoritmos de Configuração e Reconfiguração de Redes *Peer-to-Peer* sobre Redes Móveis Ad hoc

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte
24 de Setembro de 2004

A desk is a dangerous place from which to watch the world.

John le Carre (1931 -)

Resumo

Uma rede *Peer-to-Peer* (P2P) sobre uma rede móvel ad hoc é uma combinação que provê aos usuários acesso a diferentes tipos de informação em qualquer lugar e a qualquer instante. Este trabalho aborda a questão da (re)configuração neste cenário altamente dinâmico. São apresentados três algoritmos projetados para guiar a (re)configuração da rede P2P no cenário ad hoc. Os algoritmos têm como objetivo utilizar os recursos escassos, tal como energia e largura de banda, de maneira eficiente, aumentando o desempenho e o tempo de vida da rede. Como base para comparação, desenvolveu-se também um algoritmo inspirado no Gnutella. Através de simulação, mostrou-se que os algoritmos propostos alcançam seus objetivos, apresentando uma boa relação de custo-benefício.

Abstract

A Peer-to-Peer (P2P) network over a mobile ad hoc network is a combination that provides users with means to access different kinds of information anytime and anywhere. In this work we study the (re)configuration issue in this highly dynamic scenario. We present three algorithms designed to guide the (re)configuration of the P2P network in ad hoc scenario. These algorithms are especially concerned with the constraints of the environment presented: they aim to use the scarce resources of the network, such as energy and bandwidth, in an efficient way, improving performance and network lifetime. A simple Gnutella-like algorithm was also implemented, in order to be used as basis for comparison. All the algorithms were tested through simulations. The results show that the proposed algorithms achieved most of their goals, presenting a good cost-benefit relation.

Agradecimentos

Muitas pessoas participaram desta caminhada. Não tenho a pretensão de citar todos os que merecem meus agradecimentos, mas gostaria de registrar alguns nomes.

Agradeço aos professores e funcionários do DCC, que, pela forma com que me acolheram e confiaram em mim, sempre me incentivaram a seguir o caminho do Mestrado. Em especial ao meu orientador, Prof. Antônio Alfredo Loureiro, pela compreensão e apoio.

Aos colegas, pela companhia, amizade, solidariedade e ajuda. E, claro, ao Rainer: aluno brilhante e pessoa extraordinária, com quem eu tive o privilégio de conviver e trabalhar desde os tempos de caloura na graduação. A ele, e também à Marisa Vasconcelos, meu agradecimento especial pela participação no trabalho que originou esta dissertação.

Aos amigos, que sempre estiveram ao meu lado, dando aquele empurrãozinho nos momentos de desânimo, estresse e cansaço. Dani Cota, muito obrigada!

Aos meus irmãos, que, além de terem sido um belo "cartão de visita" para mim na Computação, sempre me apoiaram nesta escolha.

Aos meus pais, pessoas deslumbrantes que sempre fizeram de tudo para me ajudar a realizar meus sonhos e metas. Faltam palavras para agradecer tanto amor, dedicação, luta, apoio... Obrigada! Vocês estão de parabéns: concluíram seu terceiro mestrado!

Ao Marcelo, meu noivo, por tudo... É impossível descrever ou enumerar o que tenho para agradecer a você.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Computação Móvel tem como um de seus objetivos permitir que as pessoas possam acessar os mais distintos tipos de informação *a qualquer momento* e *em qualquer lugar*. No geral, a arquitetura cliente/servidor não é adequada para satisfazer a essa demanda devido a vários motivos: o servidor pode estar indisponível (pouca tolerância a falhas) ou sobrecarregado (problemas de escalabilidade), ou não haver nenhuma infra-estrutura de acesso ao servidor ou a outras entidades.

Parte dos problemas acima, como escalabilidade e tolerância a falhas, podem ser tratados por algumas aplicações *peer-to-peer* (P2P). As redes P2P são redes virtuais sobre as redes físicas. Elas são formadas por *peers* chamados *servents*—nós que atuam como clientes e servidores. Este papel híbrido dos *peers* permite a troca de dados de uma forma completamente descentralizada—apesar de algumas redes P2P também possuírem uma entidade central—tornando a rede mais escalável, mais tolerante a falhas e, portanto, disponível praticamente *a qualquer momento*. A atual popularidade das redes P2P comprova que elas foram bem sucedidas em atender à demanda *a qualquer momento*.

Em relação à questão do acesso à informação *em qualquer lugar*, pode-se dizer que ela está relacionada à rede propriamente dita, isto é, ela depende da infra-estrutura fornecida. Existem redes, entretanto, que não dependem da infra-estrutura física, como é o caso das redes ad hoc. Nessas redes, a comunicação entre os nós é baseada em cobertura de rádio e pode ser feita diretamente (ponto-a-ponto) ou utilizando outros nós como roteadores. Os nós podem ser estáticos ou móveis e, neste último caso, a rede é chamada MANET (*Mobile Ad-Hoc Network*). É possível, portanto, que um grupo de pessoas utilizando seus telefones celulares, PDAs (*Personal Digital Assistants*) ou *notebooks* formem uma MANET *em qualquer lugar*.

A partir dos pontos acima, é razoável pensar que redes P2P sobre redes ad hoc podem ser uma boa solução para a questão de acessar informação *a qualquer momento* e *em qualquer lugar*. Entretanto, esta combinação leva a um cenário altamente dinâmico: nós entrando e saindo da rede P2P virtual, e nós tornando-se acessíveis e inacessíveis ao alcance de transmissão da rede ad hoc. Conseqüentemente, as referências entre os nós mudam constantemente, demandando uma freqüente reconfiguração. Isto, por sua vez, pode causar um grande impacto nos recursos escassos da rede, tais como energia dos dispositivos e largura de banda. Com o objetivo de controlar e diminuir esse impacto, foram projetados algoritmos para (re)configuração de redes P2P sobre redes ad hoc e analisados seus comportamentos.

Normalmente os estudos de aplicações P2P sobre redes ad hoc (ver capítulo 3) não abordam as questões de (re)configuração. Em muitos destes trabalhos as referências entre os nós são criadas indiscriminadamente através de *broadcasts*, ou dependem de uma entidade central, ou em alguns casos são criadas aleatoriamente. A relevância deste trabalho é, portanto, estudar maneiras adequadas de (re)configurar uma rede P2P sobre redes ad hoc, de forma totalmente autônoma e levando em consideração as sérias limitações que este cenário representa.

1.1 Objetivo

Embora haja alguns estudos em redes P2P sobre ad hoc (veja capítulo 3), geralmente não há a preocupação sobre os problemas que a (re)configuração traz. Muitas vezes, as referências entre os nós simplesmente aparecem ou são criadas indiscriminadamente através de *broadcasts*. Outras vezes, elas se mostram dependentes do auxílio de alguma entidade externa. Assim, este trabalho tem como objetivo a proposição e análise de maneiras adequadas de (re)configuração de uma rede P2P sobre ad hoc que leve em consideração as restrições que esse cenário apresenta. Além disso, a fim de explorar ao máximo a flexibilidade e independência que esta combinação de redes proporciona, objetivamos que nossas propostas permitam que as redes sejam auto-configuráveis, ou seja, sejam independentes de entidades externas.

1.2 Contribuições

A primeira contribuição deste trabalho é o tema abordado. Não são poucos os estudos que tratam de redes P2P e nem em menor quantidade são os que abordam as redes ad hoc. Até mesmo os trabalhos que tratam da combinação de tais redes têm se multiplicado. Porém, as pesquisas sobre desempenho das redes P2P, por exemplo, focam no sucesso das consultas e têm como premissa uma rede já formada. Os trabalhos sobre as redes ad hoc, por sua vez, visam, primordialmente, avaliar a complexa questão da eficiência dos protocolos de roteamento. Assim, nosso trabalho contribui para o entendimento de um campo ainda não muito explorado: a questão da configuração e reconfiguração das redes P2P sobre redes ad hoc.

Além do assunto escolhido, contribuímos também com a elaboração de algoritmos: este trabalho apresenta quatro algoritmos de (re)configuração. O primeiro, o qual chamamos de *Básico*, foi inspirado no Gnutella e é utilizado como base de comparação para os algoritmos propostos. Os outros três, por sua vez, constituem esforços de se otimizar os processos de configuração e reconfiguração das redes P2P sobre redes ad hoc, e representam nossa principal contribuição.

Alguns resultados deste trabalho foram publicados no *International Parallel and Distributed Processing Symposium - IPDPS 2003* [1] e, em breve, serão publicados no *Journal of Parallel and Distributed Computing - Special Issue* [2](aceito para publicação).

1.3 Organização

O restante do trabalho organiza-se da seguinte maneira. No capítulo 2 são apresentados os problemas e conceitos que guiaram o desenvolvimento do trabalho. Nele são descritos brevemente as redes P2P, as redes ad hoc e os protocolos de roteamento utilizados nas simulações. Logo após, é feita a contextualização do papel da (re)configuração nos cenários que combinam tais redes. O capítulo é encerrado com o conceito do efeito *Small World*, que constituiu a diretriz fundamental para o desenvolvimento de um dos algoritmos propostos.

No capítulo 3 são apresentados os trabalhos relacionados. De uma forma geral, os estudos selecionados abordam a avaliação de desempenho em redes P2P, em redes ad hoc e em redes P2P sobre ad hoc.

Os quatro algoritmos de (re)configuração são tratados no capítulo 4, no qual tem-se uma descrição em alto nível de seus comportamentos. O primeiro algoritmo, chamado *Básico*, foi implementado com o objetivo de servir como uma base para comparação na análise de

desempenho dos demais algoritmos propostos, os quais são apresentados separadamente na seção 4.2.

O capítulo 5 registra a parte prática do trabalho, que consiste na codificação e teste, através de simulações, dos algoritmos propostos. Algumas informações, já apresentadas nas descrições dos algoritmos, são recapituladas e detalhadas. Os cenários simulados, juntamente com suas constantes e variáveis, são descritos logo antes dos gráficos, que sintetizam os resultados obtidos e são permeados pela análise do desempenho dos algoritmos propostos.

Finalmente, o capítulo 6 apresenta as conclusões e alguns trabalhos futuros.

Capítulo 2

Fundamentos

2.1 Redes Peer-to-Peer

As redes *peer-to-peer* (P2P) são redes virtuais construídas na camada de aplicação. Elas possuem mecanismos próprios de roteamento que permitem a dispositivos computacionais compartilhar informação e recursos diretamente, sem servidores dedicados. Assim, vê-se que cada membro dessa rede atua como cliente e servidor, sendo portanto chamado de *servent* [3].

Os sistemas P2P são classificados em três principais categorias: centralizados, descentralizados e híbridos [4, 5].

Em sistemas P2P de topologia centralizada, a coordenação entre os *peers* é feita por um servidor central. Após receber a informação desse servidor, os *peers* passam a se comunicar diretamente. Algumas vantagens deste tipo de sistema são a facilidade de gerência e a segurança. Como desvantagens tem-se a baixa tolerância a falhas, uma vez que alguns dados são mantidos somente pelo servidor central, e limitação da escalabilidade pela capacidade do servidor. Entretanto, a escalabilidade ainda pode ser conseguida graças ao crescente poder de processamento das máquinas, que possibilita ao servidor atender a um grande número de usuários. Alguns exemplos desse tipo de sistema são o sistema SETI@Home [6], que possui um *dispatcher* central de tarefas, e a arquitetura de busca do Napster.

Em sistemas de topologia descentralizada, como por exemplo o Freenet [7] e o Gnutella, todos *peers* possuem funções iguais. A comunicação é feita através de múltiplos *unicasts*, onde as mensagens dos *peers* são repassadas por outros *peers*. Algumas vantagens importantes são a extensibilidade e tolerância a falhas. A escalabilidade nesse tipo de sistema

é difícil de ser medida pois a adição de novos usuários, ao mesmo tempo em que torna a rede mais capacitada, aumenta o custo de se manter a consistência dos dados.

Nos sistemas de topologia híbrida (centralizado + descentralizado), os *peers* redirecionam suas consultas para os *super-peers*, os quais comunicam entre si de maneira descentralizada. As vantagens desta topologia assemelham-se às da topologia descentralizada com uma melhoria na questão da consistência dos dados, uma vez que parte deles é mantida somente pelos *super-peers*. Alguns exemplos do sistema híbrido são o KazaA [8] e o Morpheus.

A tabela 2.1 (de [5]) resume as vantagens e desvantagens das topologias distribuídas.

	Centralizada	**Decentralizada**	**Híbrida**
Gerenciabilidade	sim	não	não
Extensibilidade	não	sim	sim
Tolerância a Falhas	não	sim	sim
Segurança	sim	não	não
Susceptibilidade a processos judiciais	sim	não	não
Escalabilidade	depende	talvez	aparentemente

Tabela 2.1: Os tipos de topologias e suas características.

Nesse trabalho, foram propostos algoritmos aplicáveis somente às topologias descentralizada e híbrida pois elas foram consideradas mais próximas da realidade. Além disso, a alta dinamicidade do ambiente considerado nas simulações faz com que a extensibilidade, presente somente nestas topologias, se torne uma questão importante. Os algoritmos desenvolvidos inspiraram-se no protocolo descentralizado Gnutella, especialmente no tocante ao mecanismo de consulta.

Gnutella é um protocolo de domínio público utilizado principalmente para compartilhamento, busca e recuperação de arquivos e conteúdo. Para fazer parte dessa rede, os *servents* devem se conectar a vizinhos que já pertençam à rede. Uma vez conectado, o *servent* enviará mensagens através de *broadcasts* para seus vizinhos e servirá de roteador retransmitindo mensagens de outros *p*eers. Os tipos de mensagens trocadas nessa rede são: mensagens para integrar-se à rede (*pings* e *pongs*); mensagens de consulta com informações sobre o conteúdo pesquisado; e os arquivos trocados, que são transferidos diretamente entre os *peers*. Ao buscar arquivos, o *servent* envia uma mensagem de consulta a todos seus vizinhos. Estes vizinhos, além de responderem à consulta, repassam-na a seus respec-

tivos vizinhos. A fim de evitar a propagação indefinida, todas as mensagens possuem um campo TTL (*time-to-live*) inicializado com um valor limite e armazenam o número de *hops* passados [3].

2.2 Redes Móveis Ad Hoc

As redes móveis ad hoc são redes auto configuráveis cujos elementos são, por definição, fonte e sorvedouro de informação e roteador para fluxos de informação de outros usuários. Nas MANETs (*Mobile Ad hoc Networks*) os elementos podem se mover de forma praticamente irrestrita, desde que haja ao menos um outro nó pertencente à rede em seu raio de alcance, caso contrário, ele perde o acesso à esta rede [9].

Os dispositivos podem se comunicar diretamente (ponto-a-ponto) ou, quando o raio de transmissão não é suficiente, utilizando outros nós como roteadores. A fonte de energia limitada dos dispositivos restringe o seu raio de transmissão, uma vez que quanto maior o alcance, maior o consumo de energia. Assim, o raio de transmissão é tipicamente pequeno quando comparado à abrangência da rede.

Todos os nós em uma MANET usam uma mesma faixa de freqüência, que representa o seu meio compartilhado para receber e transmitir dados. A transmissão destes dados é feita através de *broadcasts* físicos, ou seja, as MANETs são altamente sujeitas a congestionamentos e a colisões.

Usualmente, as redes móveis ad hoc são orientadas ao usuário e não a dados. O objetivo destas redes é conectar usuários com seus terminais a outros usuários, ao invés de distribuir dados de forma redundante pela rede. Isto significa que uma rota é definida de uma origem até um destino específico, ambos identificados por seus endereços únicos, com o intuito de possibilitar a comunicação entre estes usuários.

Esse tipo de rede é utilizado principalmente em cenários em que não há nenhuma infraestrutura de rede fixa. Alguns exemplos de utilização são: convenções ou reuniões, onde as pessoas, por comodidade, desejam trocar informações rapidamente [10], e operações de emergência em áreas devastadas por furacões ou por terremotos.

As redes móveis ad hoc apresentam, além das restrições comuns às redes sem-fio, o desafio adicional de lidar com uma topologia muito dinâmica. Quando dois nós se afastam muito e não estão ao alcance de rádio um do outro, a ligação entre eles é desfeita. Neste caso, as tabelas de roteamento, pelo menos as destes dois nós, devem ser atualizadas. Uma vez que os nós também atuam como roteadores, estas novas informações também devem

ser repassadas aos demais nós da rede.

Quanto maior a velocidade dos nós e a sua quantidade, mais freqüentes são as atualizações de topologia e, conseqüentemente, maior a largura de banda consumida pelas mensagens que comunicam as alterações. Assim, a escalabilidade das redes móveis ad hoc depende da utilização de algoritmos de roteamento bastante eficientes. Talvez por isso este seja o tema mais discutido quando se trata das redes móveis ad hoc.

Os algoritmos de roteamento em redes móveis ad hoc, de uma forma geral, podem ser divididos em dois grupos: os proativos, ou orientados a tabela (*table driven*), e os reativos, também conhecidos como sob demanda (*on demand*).

Um nó que executa o algoritmo proativo tem uma visão geral da rede a todo momento, assim como um roteador comum na Internet. Todas as atualizações da topologia são difundidas imediatamente ou em um pequeno intervalo de tempo a todos os nós na rede. Desta forma, o estabelecimento de rotas acontece de forma bastante rápida. O problema deste tipo de algoritmo é que se o número de nós na rede tende a crescer, ele se torna impraticável.

Os nós que executam algoritmos reativos, por sua vez, não enviam quaisquer tipos de atualizações de topologia aos seus vizinhos. Quando desejam descobrir uma rota para outro nó, eles enviam à rede um pedido de rota. A resposta é enviada pelo nó final ou por um nó intermediário que tenha feito um pedido de rota para o mesmo destino anteriormente.

Vários algoritmos de roteamento proativos e reativos já foram propostos para MANETs, cada qual com suas vantagens e desvantagens, dependendo do cenário analisado. Em [11], Oliveira, Siqueira e Loureiro comparam o desempenho de alguns destes algoritmos sob uma aplicação P2P. O estudo constatou que o AODV (*Ad hoc On-Demand Distance Vector Routing*) [12] obteve o melhor desempenho em cenários de alta mobilidade. Diante disso, inicialmente ele foi o único protocolo utilizado em nossas simulações. Porém, como poderá ser visto no capítulo 5, ao se observar o peso do protocolo de roteamento no desempenho geral dos algoritmos, resolveu-se simular alguns cenários também com outro protocolo, no caso, o DSR.

2.2.1 DSR

O protocolo *Dynamic Source Routing* (DSR) [13] é um dos principais algoritmos de roteamento sob demanda. Ele é baseado no conceito de *source routing*, e inclui duas fases principais: a descoberta de rota e a manutenção de rota.

Quando um nó deseja enviar uma mensagem para um destinatário para o qual sua

tabela de roteamento não possui rota, ele realiza a difusão de uma mensagem de *requisição de rota* para seus vizinhos. Esta mensagem contém o endereço da origem, o endereço do destino e um número de seqüência único, para que se detecte *loop*. O nó que recebe esta mensagem verifica sua tabela de roteamento, à procura de uma rota para o destino. Se ele não possuir uma rota, ele adiciona seu próprio endereço ao pacote e o repassa. Caso um nó receba cópias da mesma requisição de rota de nós intermediários distintos, ele repassa somente a primeira requisição e descarta as demais. Uma resposta será enviada de volta caso um nó possua uma rota válida até o destinatário, ou até que chegue ao próprio destinatário. Caso o nó seja um intermediário, ele adiciona à mensagem de resposta a rota que possui até o destinatário. De qualquer forma, a resposta contém a rota completa, da origem até o destino, e segue exatamente este caminho, na direção reversa.

Desta forma, todo pacote de dados que atravessa a rede da origem até o destino, contém o caminho completo, com o endereço de todos os nós intermediários. Assim, não é necessário que os nós intermediários armazenem o caminho, o que lhes propicia economia de memória. Em contrapartida, o tamanho de cada pacote aumenta, pois deve conter a informação completa sobre a rota.

A outra etapa, a manutenção da rota, é feita por pacotes de *erro de rota*, que são enviados de volta sempre que um nó intermediário ou o destinatário não são mais acessíveis. Todo nó, ao repassar um pacote de erro de rota, apaga o *hop* que se desfez, assim como todas as rotas que passam por este *hop*. O nó de origem também apaga a rota inválida e inicia uma nova requisição de rota [9].

2.2.2 AODV

O protocolo AODV (*Ad-hoc On-Demand Distance Vector Routing*) [12] também é um algoritmo de roteamento reativo, mas, diferentemente do DSR, nele cada nó tem somente a informação sobre o próximo passo de uma rota. O funcionamento do AODV, assim como o DSR, baseia-se no uso de mensagens de *route request* (RREQ) e de *route reply* (RREP).

Quando um nó deseja enviar uma mensagem para um destinatário para o qual sua tabela de roteamento não possui rota, ele realiza a difusão de uma mensagem RREQ para seus vizinhos. Esta mensagem contém o endereço da origem, o endereço do destino e um número de seqüência único. Todo nó, ao receber uma RREQ, salva o endereço do nó de quem ele recebeu essa mensagem pela primeira vez antes de repassá-la aos seus vizinhos. Caso um nó receba cópias da mesma RREQ, tais cópias serão descartadas. A RREQ é repassada pela rede até que ela alcance um nó que possua uma rota válida até o

destinatário, ou até que chegue ao próprio destinatário. Neste momento, o nó intermediário –ou o destinatário– cria uma mensagem de resposta –RREP– e a envia ao nó de quem ele recebeu a RREQ pela primeira vez. A RREP é, então, repassada pelos nós, fazendo o caminho reverso da RREQ. Todo nó, ao receber a RREP, salva o endereço do vizinho de quem ele a recebeu. Desta forma, quando a RREP chega até a origem, o caminho até o destinatário está estabelecido.

Ao fim do processo, vários nós terão armazenado informação sobre o RREQ, mas não terão recebido o RREP. Isto significa que tais nós não compõem a rota até o destino. Por isso, a informação sobre a RREQ expira e é apagada do nó.

Quando um nó intermediário percebe uma movimentação de um vizinho a jusante, ele gera uma mensagem RREP informando que o próximo passo é infinito, ou seja, que o seu vizinho está inalcançável. Esta mensagem é repassada até a origem, fazendo com que os nós intermediários e a origem apaguem as rotas existentes que levavam ao elemento que se moveu. A partir deste momento, novas RREQ deverão ser enviadas para o descobrimento de uma nova rota.

Uma grande vantagem deste protocolo, especialmente quando comparado aos protocolos *source routing*, é a reduzida carga na rede, visto que cada pacote só contém a informação sobre o próximo passo da rota. Em contrapartida, tem-se a desvantagem de uma maior demanda de memória, pois cada nó deve manter a informação sobre o próximo passo de todas as rotas que passam por ele [9].

2.3 Redes P2P sobre Redes Móveis Ad hoc e o Papel da (Re)Configuração

Os dispositivos móveis sem-fio possuem, em geral, um alcance de transmissão restrito devido a sua fonte limitada de energia. Desta forma, a procura por dados deve ser realizada preferencialmente em um pequeno raio de alcance. Essa procura pode ser realizada de duas formas distintas. A primeira utiliza uma infra-estrutura fixa que provê acesso contínuo a uma rede de informação como a Internet ou uma *intranet* particular, e que, geralmente, envolve um alto custo. A segunda forma de acesso não necessita de infra-estrutura fixa: um conjunto dos dispositivos móveis formam, por si só, uma rede ad hoc e os dispositivos próximos se tornam importantes fontes de dados um para o outro. Isto se assemelha ao paradigma P2P, no qual um elemento de rede atua, ao mesmo tempo, como cliente, servidor e roteador. Percebe-se, então, que em ambas as redes a colaboração entre os elementos é

fundamental para o bom funcionamento do conjunto.

Apesar de tantas semelhanças, estas redes não são capazes de substituírem uma à outra. As redes P2P se referem à camada de aplicação na pilha de protocolos, enquanto que as MANETs focam na camada de rede e inferiores. Desta forma, podemos dizer que suas afinidades, na verdade, tornam-nas naturalmente complementares.

Uma das grandes vantagens de uma rede P2P sobre uma rede ad hoc é a facilidade de se formar a rede, uma vez que não é necessária infra-estrutura e nem há a dependência de um servidor central. Exemplos de possíveis usos de uma rede P2P sobre MANET incluem compartilhamento de dados sobre tráfego e sobre o tempo através da comunicação entre carros em uma MANET de longo alcance, compartilhamento de músicas e vídeos entre dispositivos móveis em um ambiente público e aplicações que alertam sobre a proximidade de determinadas pessoas baseado no cruzamento de perfis.

Por outro lado, P2P sobre MANETs representam uma combinação muito dinâmica: na rede P2P tem-se a constante entrada/saída de nós, enquanto que, na ad hoc, está presente a mobilidade dos elementos. Uma vez que ambas são redes auto-configuráveis, esta dinamicidade torna fundamental, entre outras coisas, que se dê uma atenção especial às questões relacionadas à (re)configuração, como visto a seguir.

As redes P2P são redes virtuais, construídas na camada de aplicação, que focam no compartilhamento de dados. Por serem orientadas a dados, normalmente seus protocolos são projetados para obterem um melhor desempenho no mecanismo de busca, ou seja, achar as informações mais rapidamente e com uma maior freqüência. A topologia de uma rede P2P, entretanto, pode ser bastante distinta da conformação física, de forma que um protocolo P2P considerado eficiente pode causar um impacto negativo na rede ad hoc.

Geralmente, os protocolos P2P têm em comum um mecanismo que consiste em ter uma lista de vizinhos para os quais as consultas são repassadas. Nas redes fixas, cenário tradicional dos sistemas P2P, o estado de um vizinho normalmente oscilaria entre *disponível* e *indisponível*. Nos cenários móveis, entretanto, um *peer* próximo pode tornar-se distante num curto intervalo de tempo, mantendo seu estado de *disponível*. Neste caso, a distância medida em passos (*hops*) poderia continuar sendo a mesma na rede P2P, mas, na rede ad hoc ela aumentaria, gerando um tráfego mais pesado na camada física. A busca de dados poderia até mesmo continuar com o mesmo desempenho de antes, mas a rede ad hoc seria sobrecarregada e os nós demandariam muito mais de suas fontes de energia. Portanto, é importante que se monitore as referências entre os nós, (re)configurando a rede P2P levando em consideração a topologia da rede ad hoc. Essa questão é abordada em nosso

trabalho, e representa o maior objetivo dos algoritmos propostos.

2.4 O Efeito *Small World*

Em 1967, Stanley Milgram, professor da Universidade de Harvard, nos Estados Unidos, realizou uma experiência que se tornou referência no estudo das redes sociais [14]. Ele enviou 160 cartas a um conjunto de pessoas escolhidas aleatoriamente em Omaha, cidade do estado de Nebraska, situado na região central dos E.U.A. As pessoas deveriam repassar suas cartas para uma determinada pessoa-alvo em Boston, estado de Massachusetts, na costa leste. A única regra consistia em usar como intermediários somente pessoas que fossem seus conhecidos (*on a first-name basis*). A cada passo, a pessoa entregava a carta a um amigo que ela considerasse capaz de fazê-la chegar, diretamente ou através de outro intermediário, às mãos do alvo em Boston.

Ao fim, 42 das 160 cartas originais alcançaram seu destino, com uma média de 5,5 intermediários. Este foi um número pequeno se comparado à população americana da época, 200 milhões de pessoas, e à distância que separa origem e destino, aproximadamente 2.350 quilômetros. O experimento demonstrou o que ficou popularmente conhecido como efeito *Small-world* e *Seis Graus de Separação*.

Estes conceitos podem ser facilmente aplicados no mundo das redes de computadores: representando a rede por um grafo no qual as arestas são as conexões físicas e os vértices são computadores ou roteadores, pode-se imaginar que cada vértice é uma pessoa e que cada conexão é uma relação de amizade que ela possui. Este tipo de grafo é chamado de rede social. Observando desta forma, torna-se natural imaginar que o efeito *Small-world* também possa se manifestar nas redes de computadores. Entretanto, nem todas as redes exibem esta característica, de forma que a questão crucial passa a ser como se atingir ou como detectar este efeito.

Um importante passo para se resolver esta questão foi observar que em qualquer grupo social existem alguns relacionamentos que são responsáveis por conectarem grupos bem distantes entre si, funcionando como "pontes". Mais do que isso, mostrou-se em [15] que até mesmo a existência de um pequeno número de pontes pode reduzir drasticamente o comprimento dos caminhos em um grafo. Este fato foi detectado ao se comparar grafos *regulares* e *aleatórios*, como explicado a seguir.

Um *grafo regular* consiste de um anel de n vértices, cada qual conectado aos seus k vizinhos mais próximos, conforme ilustrado pela figura 2.1. Se observarmos grandes grafos

regulares em que n é muito maior que k —para um k muito maior que 1— o caminho mínimo característico é aproximadamente $n/2k$ [4].

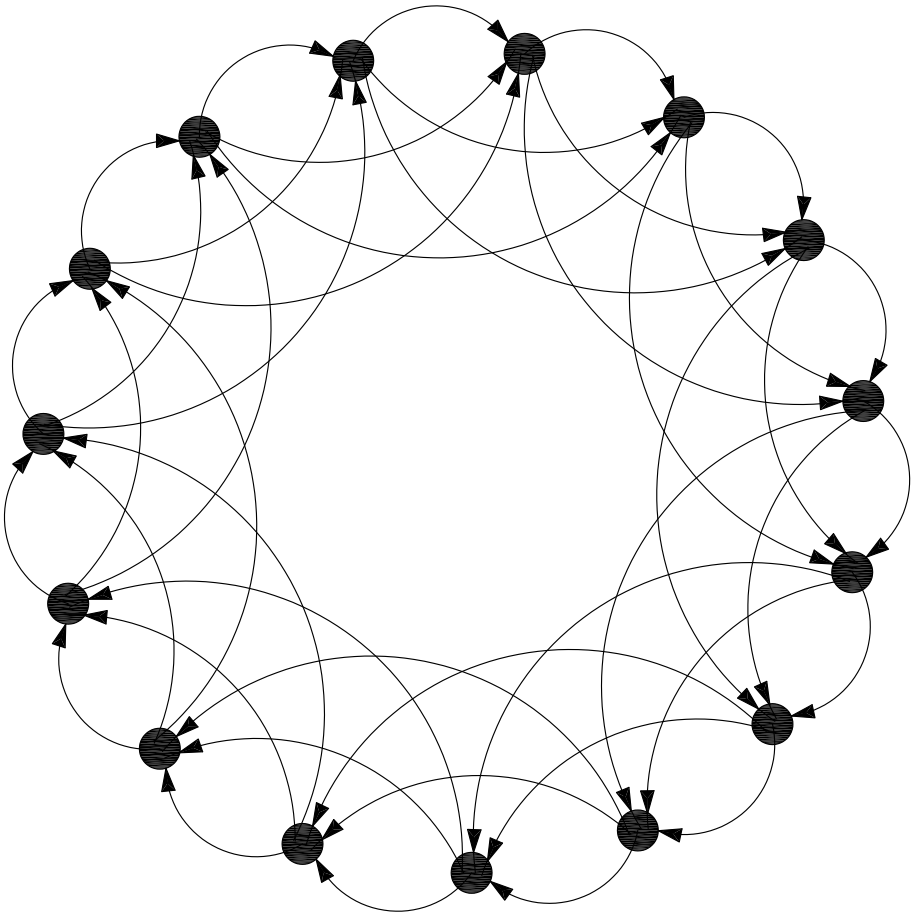

Figura 2.1: Exemplo de grafo regular com k = 3.

Nos *grafos aleatórios* –figura 2.2, entretanto, os vértices são conectados uns aos outros de forma aleatória, sendo k o número médio de arestas por vértice. O caminho mínimo característico em grandes grafos deste tipo é aproximadamente $\log n / \log k$ [4], muito melhor que o caminho característico do grafo regular similar, que é de $n/2k$.

Outra diferença entre estes dois tipos de grafo é o *coeficiente de agrupamento*. O coeficiente de agrupamento pode ser interpretado como sendo a chance de dois vizinhos de um nó em comum serem conectados entre si. Seja con_e o número de conexões existentes entre todos os vizinhos de um nó e con_p, o número de todas as conexões que poderiam existir entre tais vizinhos. O coeficiente de agrupamento é dado por con_e/con_p. Os grafos aleatórios normalmente têm baixos coeficientes de agrupamento, ao contrário dos

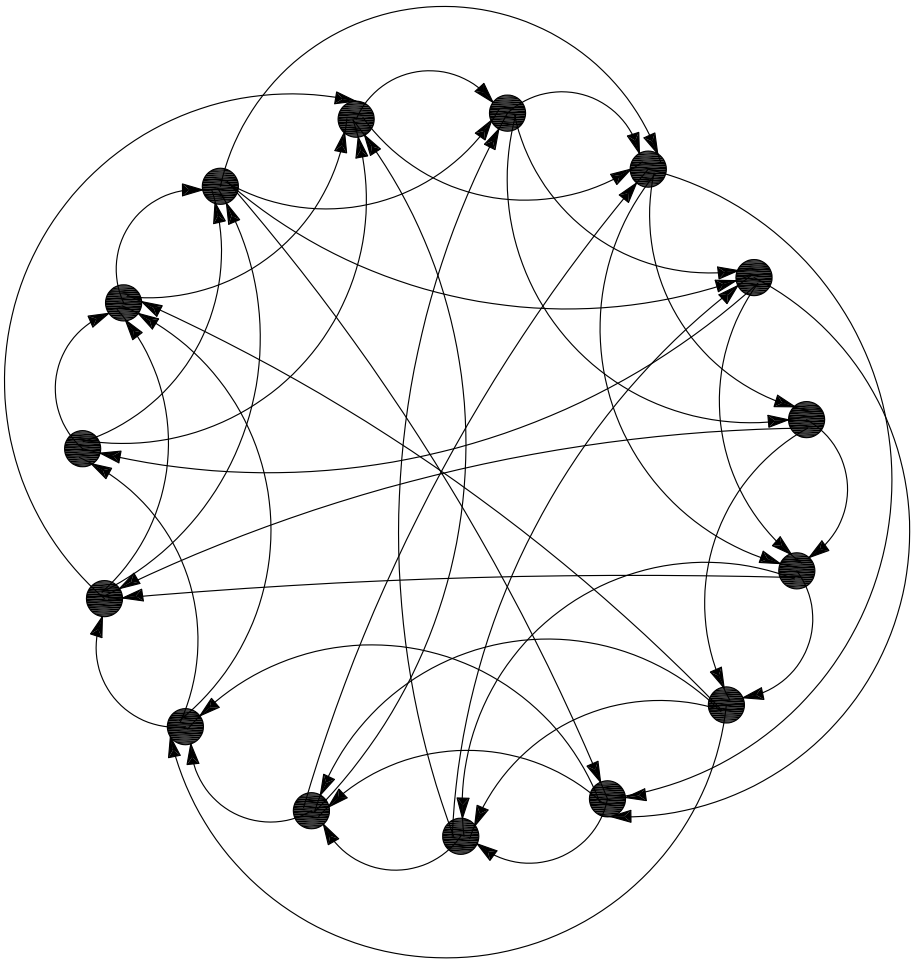

Figura 2.2: Exemplo de grafo *small-world* com k = 3.

grafos regulares, que possuem altos coeficientes de agrupamento.

Com certeza, existem casos intermediários entre os grafos *regulares* e os *aleatórios*. Na verdade, a maioria das redes reais constituem tais casos. Descobriu-se, inclusive, que pequenas modificações nas conexões dos *grafos regulares* são capazes de diminuir bastante o comprimento do caminho mínimo característico, tornando-o semelhante ao dos *grafos aleatórios*. A mudança de algumas arestas que iam até vizinhos para vértices escolhidos aleatoriamente pode representar a criação de *pontes* entre grupos distantes de nós. Estas *pontes* proporcionam a diminuição dos caminhos sem grande alteração do coeficiente de agrupamento [15].

Os grafos que possuem altos coeficientes de agrupamento e, ao mesmo tempo, caminhos mínimos característicos curtos são chamados de grafos *small-world* –figura 2.3. O nosso

algoritmo de (re)configuração chamado *Aleatório*, mostrado na seção 4.2.2, resultou de alterações de outro algoritmo, denominado *Regular*, que objetivaram a construção das redes *peer-to-peer* como grafos *small-world*.

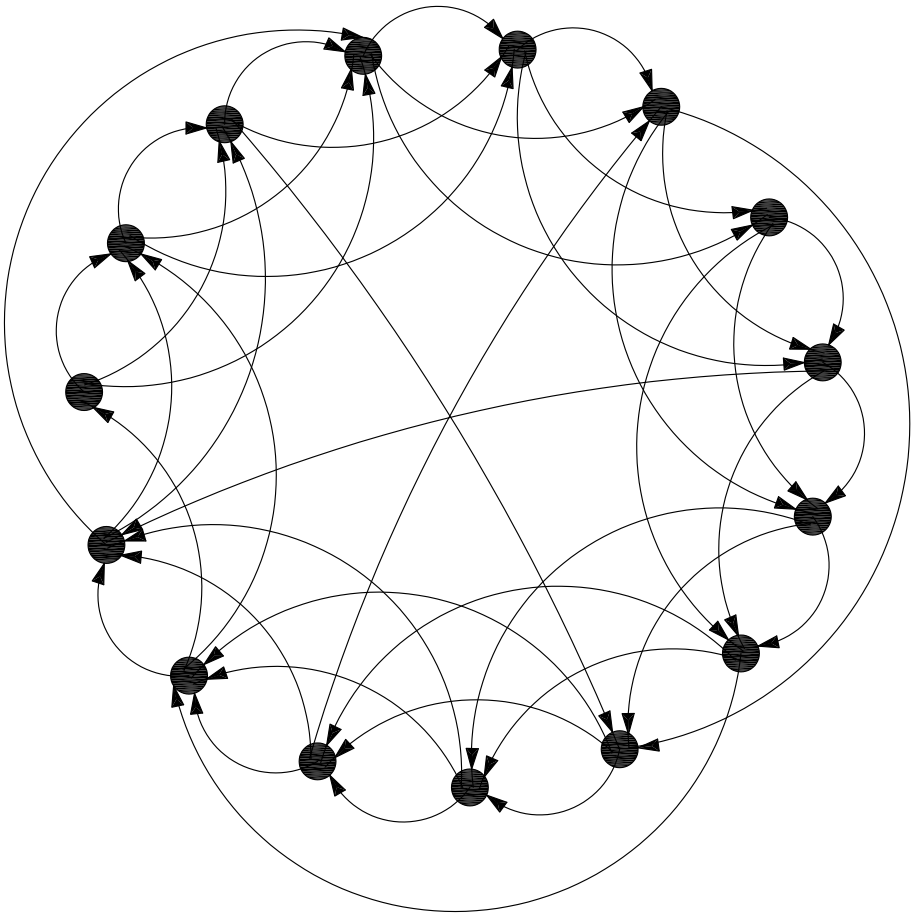

Figura 2.3: Exemplo de grafo *small-world* com k = 3.

Capítulo 3

Trabalhos Relacionados

Esta dissertação trata de uma rica combinação de fatores: propor algoritmos de *configuração* e *reconfiguração* de *redes P2P* sobre *redes móveis ad hoc* e estudar o seu *desempenho*. Em nossas pesquisas, não foram encontrados trabalhos que tratem destes mesmos dois temas. Deparamo-nos, entretanto, com diversos estudos sobre desempenho em redes P2P, em redes ad hoc e em redes P2P sobre ad hoc. Todos foram importantes, pois tratam de variáveis como algoritmo de roteamento e mecanismo de busca que, apesar de não representarem o nosso foco, compõem nosso universo de estudo e impactam a avaliação do desempenho de nosso sistema.

Em [4], a avaliação do desempenho de redes P2P é feita utilizando-se de dois estudos de caso: Gnutella e Freenet. Ambas são redes consolidadas e bastante difundidas que se baseiam em uma infra-estrutura fixa, a Internet. Antes da apresentação dos experimentos e dos resultados, abordam-se os tipos de redes P2P e o significado da análise de desempenho em tais redes. Ao contextualizar as métricas utilizadas, é realizada, ainda, uma interessante apresentação sobre o efeito *Small-World*, incluindo uma breve passagem pela teoria de grafos.

A avaliação das redes observou aspectos como tolerância a falhas, escalabilidade, eficiência (ter sucesso em mais buscas), rapidez (ter sucesso com menos passos) e pior caso.

Nas simulações do Freenet, observou-se que, na média, as consultas são rápidas, mas no pior caso elas demoram um tempo desproporcionalmente longo. Com relação ao caminho médio percorrido por uma consulta, constatou-se que ele cresce logaritmicamente com relação ao tamanho da rede, ou seja, a Freenet é uma rede bastante escalável.

Na avaliação da tolerância a falhas, foram observados dois possíveis cenários: falhas naturais e ataques. Uma vez que a quantidade de ligações não é uniforme entre os nós, um

cenário de ataque pode ser perigoso. Aqueles nós com mais ligações representam os alvos preferenciais e a saída de vários destes nós de uma vez pode comprometer o funcionamento geral da rede. As falhas naturais, entretanto, têm maior possibilidade de atingirem nós com poucas ligações, pois eles existem em maior quantidade. Assim, a rede não sofre grande impacto.

O desempenho do Gnutella possui outras características. As buscas são satisfeitas bem rapidamente, tanto no caso médio quanto no pior caso. Aliás, ambos são próximos do ótimo caso. Para alcançar isto, entretanto, a rede tem que contactar muitos nós ou seja, tem um custo bem maior que a Freenet.

As diferenças continuam quando nos referimos à tolerância a falhas. Comparado à Freenet, o Gnutella tolera melhor os ataques, mas é mais impactado pelas falhas naturais.

Por fim, devido ao seu mecanismo de busca baseado no uso de *broadcasts*, o Gnutella apresenta um consumo de largura de banda por consulta linearmente proporcional ao tamanho da rede. Isso significa que ele apresenta um sério problema de escalabilidade.

Em [16], também é feita a comparação entre sistemas P2P. Este trabalho, entretanto, não se refere às redes tradicionais, já que estuda sistemas P2P sobre redes móveis ad hoc (MANETs). Os autores propõem cinco métodos de roteamento que combinam protocolos de roteamento das MANETs com protocolos já existentes de busca em redes P2P.

O trabalho identifica alguns problemas comuns às redes *peer-to-peer* e ad hoc, como a grande variação da topologia, o problema de se achar dados/rotas e os problemas de escalabilidade que surgem com o uso de inundação ou *broadcasts*. São apontadas, também, algumas diferenças: a conotação "virtual" dos *broadcasts* da rede P2P –em contraposição aos *broadcasts* físicos das MANETs; as restrições de recursos nas MANETs, o que não representa uma grande preocupação dos sistemas P2P sobre a Internet; e o fato das redes P2P referenciarem a camada de aplicação, enquanto as MANETs focam na camada de rede e inferiores.

Os métodos propostos são identificados da seguinte forma: *Broadcast* sobre *broadcast*; *Broadcast*; DHT (*Distributed Hash Table*) sobre *broadcast*; DHT sobre DHT; e DHT. O primeiro deles é fácil de ser implementado, porém sobre do problema da escalabilidade por causa dos *broadcasts* duplos. Sua complexidade foi apresentada como $O(n^2)$, sendo n o número de *peers*. O segundo método também é facilmente implementado e mantido e, apesar de também ser pouco escalável, possui complexidade de $O(n)$.

Os outros três métodos, por utilizarem protocolos de busca baseados em DHT, já são mais difíceis de implementar e de manter, sendo o quarto método, DHT sobre DHT,

considerando o mais trabalhoso de todos.

A combinação de DHT sobre *broadcast* mostra-se vantajosa sobre as anteriores somente no tocante à complexidade do roteamento, que é $O(nlogn)$. O quarto método é mais escalável –$O((logn)^2)$, mas ainda fica atrás da última abordagem, que tem complexidade $O(logn)$, e é mais simples de ser mantida e implementada. O estudo conclui que as abordagens que misturam as camadas de aplicação e de rede (segundo e último métodos) apresentam boa melhora no desempenho. Além disso, observa-se que, de uma forma geral, o método *Broadcast* seria uma boa escolha para redes pequenas, enquanto que o DHT mostra-se como uma opção adequada para grandes redes.

Em [9], é feita uma comparação entre o roteamento em redes ad hoc e em redes P2P. Além de apresentar uma taxonomia de ambos os tipos de redes, propõe-se o seu uso conjunto, objetivando um efeito sinérgico.

Roteamento também é a principal questão abordada em [11], que avalia o desempenho de três algoritmos de roteamento em uma rede ad hoc sob uma aplicação P2P. Os protocolos DSDV (*Destination-Sequenced Distance-Vector Routing*), DSR (*Dynamic Source Routing Protocol*) e AODV (*Ad Hoc On Demand Distance Vector*) foram testados através de simulação. A avaliação do desempenho utilizou como métricas o número de pacotes em cada camada de protocolo, o tempo médio gasto em uma consulta, o número médio de passos até a encontrar a resposta e a energia gasta por cada *peer*.

Nos trabalhos [17, 18] o enfoque retorna à própria aplicação P2P. Neles é apresentada uma aplicação denominada 7DS *(Seven Degrees of Separation)*: uma rede para disseminação de dados em redes ad hoc. Além da implementação, estes estudos apresentam resultados de simulações que enfocam a conservação da energia, controle da cobertura de rádio e estratégias de cooperação entre os nós para a disseminação dos dados.

Em [19], é apresentada uma plataforma para desenvolvimento de aplicações Peer-to-Peer em redes ad hoc de pequeno alcance, mais especificamente em PANs (*Personal Area Networks*).

Todos os trabalhos acima estudam importantes questões relacionadas às redes P2P e a maioria deles lida com redes P2P sobre redes ad hoc. Entretanto, apesar de avaliarem o desempenho nestes cenários, os referidos estudos não abordam a questão da (re)configuração. Nosso trabalho foca exatamente neste problema, analisando seu impacto no desempenho das redes P2P sobre ad hoc.

Capítulo 4

Algoritmos de (Re)Configuração

Neste trabalho, foram desenvolvidos quatro algoritmos de (re)configuração, chamados *Básico*, *Regular*, *Aleatório* e *Híbrido*.

O algoritmo *Básico* foi implementado com o objetivo de servir como uma base para comparação na análise de desempenho dos demais algoritmos propostos. Ele é um protocolo simples, descentralizado, inspirado no *Gnutella* e que apresenta algumas inadequações ao cenário móvel sem fio. Sua utilização nos proporciona a avaliação do impacto das tentativas de otimização usadas nos novos algoritmos propostos. Desta forma, pode-se analisar o custo/benefício entre a complexidade inserida e o ganho de desempenho.

Os demais algoritmos foram propostos no intuito de otimizar a (re)configuração da rede, tendo um enfoque especial de sanar alguns problemas detectados no algoritmo *Básico*. Dois deles, o *Regular* e o *Aleatório*, também são descentralizados, possuindo diversas semelhanças com o *Básico*. Por fim, o algoritmo *Híbrido*, como o próprio nome indica, segue um paradigma um pouco diferenciado, o da topologia híbrida. Como visto na seção 2.1, neste caso não há uma descentralização total, havendo a criação de grupos nos quais a comunicação ocorre de forma centralizada. A comunicação entre os grupos, entretanto, dá-se de forma descentralizada, daí o comportamento híbrido.

Na descrição de todos estes algoritmos, será dito, por exemplo, que os nós estão *conectados*, tentando se *conectar* ou mantendo uma *conexão*. É importante destacar, entretanto, que estamos lidando com redes sem fio —ambiente no qual conexões reais entre os nós como, por exemplo, conexões TCP são ineficientes e devem ser evitadas. Deve-se, então, sempre se ter em mente que as chamadas *conexões* na verdade são referências, ou seja, elas representam o conhecimento do endereço de alguns nós especiais, seus *vizinhos*, que eles consideram correntemente alcançáveis. Desta forma, uma conexão simétrica é aquela na

qual um nó A mantém uma referência para o nó B enquanto o nó B também referencia A. Conexões assimétricas também existem e são usadas no algoritmo *Básico*.

4.1 Algoritmo Básico

O algoritmo *Básico* destina-se a representar um algoritmo de (re)configuração simples e, portanto, servir como uma base para comparação e avaliação dos demais algoritmos propostos. Ele foi inspirado no protocolo *Gnutella*. Sua principal característica — simplicidade — proporciona uma fácil implementação e manutenção, mas ignora parcialmente a natureza dinâmica da rede.

Algoritmo Básico: Estabelecimento de Conexões
enquanto o nó pertence à rede P2P
  se número de conexões < MAXNCON **então**
    tente estabelecer ***novas*** conexões a nós num raio de
      NPASSOS até o limite de MAXNCON conexões;
    aguarde TEMPO antes de nova tentativa;
  fim se
fim enquanto

Figura 4.1: Algoritmo Básico: Estabelecimento de Conexões.

Este algoritmo, mostrado nas figuras 4.1 e 4.2, utiliza três constantes denominadas MAXNCON, NPASSOS e TEMPO. A primeira representa o número máximo de conexões por nó. A segunda é o número de passos (*hops*) que uma mensagem pode percorrer; e a terceira representa o intervalo de tempo entre duas tentativas de se estabelecer conexões.

O algoritmo funciona da seguinte forma. Um nó, quando inicia sua participação na rede *peer-to-peer*, faz o *broadcast* de uma mensagem para descobrir outros nós da vizinhança os quais estejam distantes até NPASSOS. Todo nó que escuta esta mensagem, responde-a. Assim que a resposta chega, o nó estabelece uma conexão com o vizinho que a enviou até o limite de MAXNCON conexões. Caso o número de respostas seja menor que MAXNCON, e sempre que o número de conexões for menor que MAXNCON, o nó continua tentando estabelecer o restante das conexões. Entre as tentativas o nó aguarda durante um intervalo de tempo —TEMPO— para evitar uma sobrecarga de tráfego na rede. Uma vez que uma

Algoritmo Básico: **Manutenção de Conexões Estabelecidas**

enquanto esta conexão existe
  envie um *ping* ao nó conectado;
  espere algum tempo pelo *pong*;
  se o *pong* foi recebido **então**
    espere algum tempo antes de enviar o próximo *ping*;
  senão encerre esta conexão;
  fim se
fim enquanto

Figura 4.2: Algoritmo Básico: Manutenção de Conexões.

referência[1] é criada, sua validade é freqüentemente verificada através do envio de *pings*. O recebimento de um *pong* mostra que a conexão ainda existe enquanto que sua ausência significa que o vizinho não é mais alcançável e, portanto, a conexão foi desfeita.

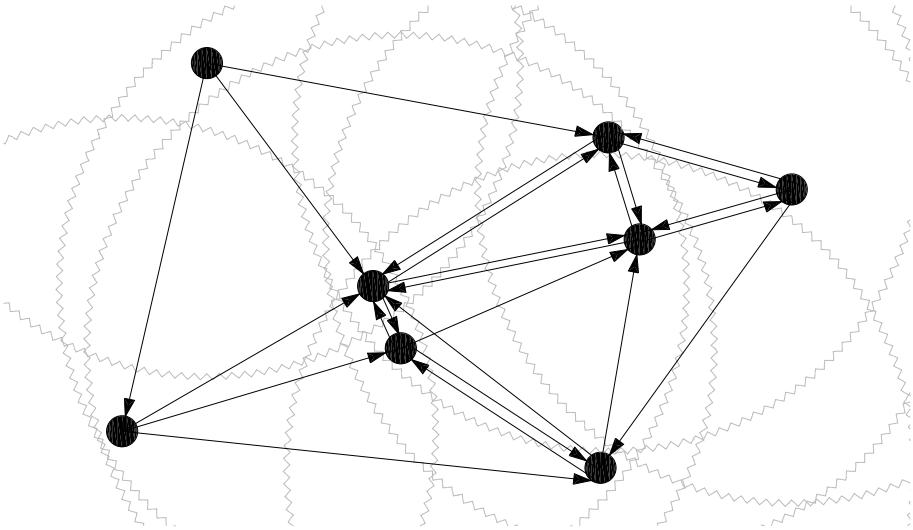

Figura 4.3: Exemplo de configuração seguindo o algoritmo *Básico*.

Na figura 4.3, é ilustrada uma possível configuração de rede P2P seguindo o algoritmo *Básico* em que o número máximo de conexões, MAXNCON, é igual a 3. Uma vez que as conexões estabelecidas através deste algoritmo não são necessariamente simétricas, elas

[1]Como mencionado, as chamadas *conexões* na verdade são referências.

foram representadas por setas que ligam um nó àquele outro ao qual ele se considera conectado. Essa assimetria garante que um *peer* cujo raio de transmissão alcance algum nó possuirá pelo menos uma conexão. Entretanto, a assimetria também pode gerar situações em que alguns *peers* são referenciados por muito mais que MAXNCON nós, ao mesmo tempo que outros elementos não considerados como vizinhos por ninguém. Assim, a carga de mensagens da rede P2P pode acabar se distribuindo de forma heterogênea, sobrecarregando alguns elementos. Tendo em vista a escassez de recursos característica dos dispositivos móveis, tal sobrecarga tem o potencial de resultar na menor sobrevida de alguns elementos, o que pode interferir no desempenho global da rede.

4.2 Algoritmos Otimizados

Antes de apresentar os algoritmos *Regular* e *Aleatório*, serão listadas suas variáveis e constantes, muitas presentes em ambos os algoritmos. Existem três variáveis: *npassos*, *passosaleat* e *tempo*. A primeira representa o número de passos que uma mensagem de *broadcast* que está à procura de uma conexão *regular* pode percorrer. Ela é inicializada com o valor NPASSOS_INICIAL, que é maior que 1, e tem como valor máximo MAXNPASSOS. A segunda variável, *passosaleat*, tem um significado semelhante mas aplica-se somente às conexões *aleatórias*. A terceira representa o intervalo de tempo que um nó espera entre duas tentativas de estabelecer uma conexão. Ela é inicializada com TIMER_INICIAL e pode crescer até MAXTIMER. Por fim, restam duas constantes ainda não explicadas: MAXNCON, que é o número máximo de conexões por nó, e MAXDIST, que é a distância máxima permitida para que dois nós ainda sejam considerados conectados (medida em número de passos na rede ad hoc).

4.2.1 Algoritmo Regular

Inicialmente, tem-se a rede ad hoc sobre a qual alguns de (ou todos os) seus nós desejam construir a rede P2P e, para tal, segue o algoritmo apresentado na figura 4.4. Como pode ser visto, cada um dos nós faz o *broadcast* de uma mensagem comunicando que ele deseja estabelecer conexões. As mensagens possuem um número específico de passos *npassos* que elas vão percorrer. Ao receber esta mensagem, um nó que deseje se conectar começa um *three-way handshake* com o remetente, tentando estabelecer uma conexão **simétrica**. Caso, naquele raio de *npassos*, só se consiga conectar **simetricamente** a menos de MAXNCON vizinhos, o nó fará um novo *broadcast* com um número maior de

passos — $npassos + 2$. Antes do novo *broadcast*, entretanto, ele aguarda por um intervalo de tempo *tempo*. Assim como no algoritmo *Básico*, este intervalo é uma tentativa de se evitar a sobrecarga de tráfego. Este mecanismo é repetido até que o máximo de MAXNCON conexões sejam estabelecidas ou até que o máximo de MAXNPASSOS seja atingido, o que ocorrer antes. Quando atribui-se o valor 0 a *npassos*, significa que o nó tentou todos os valores possíveis para *npassos* sem conseguir se conectar a MAXNCON vizinhos. Neste caso, o valor do intervalo de tempo é dobrado antes da próxima rodada de tentativas, na qual *npassos* recomeçará com o valor de NPASSOS_INICIAL. O *tempo* tem como limite superior MAXTIMER, e como limite inferior, TIMER_C_INICIAL.

Algoritmo Regular: Estabelecimento de Conexões

enquanto o nó pertence à rede P2P
  se número de conexões $<$ MAXNCON **então**
    se $npassos \neq 0$ **então**
      tente estabelecer conexões ***novas*** e ***simétricas*** a nós num raio de
       npassos até o limite de MAXNCON conexões
      aguarde *tempo* antes da próxima tentativa;
    senão
      $tempo = \min(2 \times tempo, \text{MAXTEMPO})$;
    fim se
    $npassos = (npassos + 2) \bmod (\text{MAXNPASSOS} + 2)$;
  fim se
fim enquanto

Figura 4.4: Algoritmo Regular: Estabelecimento de Conexões.

Uma vez que uma conexão é estabelecida com sucesso, o nó inicia sua manutenção como é apresentado no algoritmo da figura 4.5. A conexão é freqüentemente verificada usando-se *pings*. O recebimento de um *pong* indica que o vizinho ainda está alcançável, mas isto não é suficiente para que a conexão seja mantida. Para permanecer conectado a um nó, a distância entre eles deve ser menor que MAXNPASSOS passos. Caso a distância seja maior que este valor, a conexão é encerrada assim como na ausência de um *pong*. Uma vez que confiamos na simetria das conexões, não é necessário que uma conexão seja testada por ambos os vértices. Assim, o nó que iniciou o processo de estabelecimento da conexão ficará responsável pelo envio dos *pings*. Ao outro nó cabe responder com *pongs*

Algoritmos Regular e Aleatório: Manutenção de Conexões Estabelecidas

```
enquanto esta conexão existe
  se é o nó que iniciou o 3-way handshake então
    envie um ping ao nó conectado;
    aguarde algum tempo pelo pong;
    se o pong foi recebido então
      se o nó está mais próximo que MAXDIST então
        aguarde algum tempo antes de enviar o próximo ping;
      senão encerre esta conexão;
      fim se
    senão encerre esta conexão;
    fim se
  senão
    aguarde algum tempo pelo ping;
    se o ping foi recebido então
      envie um pong;
    senão encerre esta conexão;
    fim se
  fim se
fim enquanto
```

Figura 4.5: Algoritmo Regular: Manutenção de Conexões.

e usar um mecanismo de temporização para controlar o recebimento dos *pings*. Caso se passe um longo intervalo de tempo sem recebê-los, a conexão é encerrada.

Na figura 4.6, é ilustrada uma possível configuração de rede P2P seguindo o algoritmo *Regular* em que o número máximo de conexões, MAXNCON, é igual a 3. É importante observar que as conexões são simétricas, como demonstram as setas bidirecionais que as representam. A simetria faz com que haja um equilíbrio entre as referências: um nó que possui n vizinhos em sua lista, também é vizinho de n nós. Assim, espera-se que a carga de mensagens da rede P2P seja melhor distribuída entre os elementos da rede. A simetria, entretanto, também pode gerar um problema: um nó pode não conseguir se conectar aos outros caso eles já tenham MAXNCON conexões. Na verdade, esta situação só se torna realmente crítica em cenários estáticos, em que as conexões se preservam durante um

longo intervalo de tempo. Em situações com uma certa dinamicidade, as conexões estão constantemente se desfazendo e se refazendo, dando a oportunidade de um nó "isolado" se integrar ao grupo.

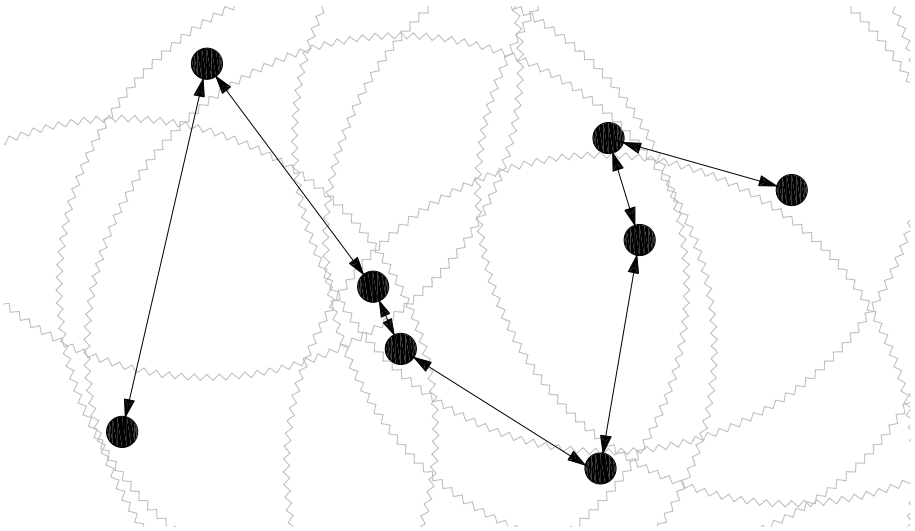

Figura 4.6: Exemplo de configuração seguindo o algoritmo *Regular*.

O algoritmo Regular possui quatro aprimoramentos quando comparado ao algoritmo *Básico*. O primeiro é que o número de passos que as mensagens à procura de conexões percorrem é aumentado gradualmente. Uma vez que este tipo de mensagem é enviada por *broadcast*, controlar o número de passos significa menos tráfego na rede. Como estamos lidando com redes sem fio, as quais têm restrições quanto à largura de banda e consumo de energia, esta ação pode ter um impacto positivo razoável.

A natureza dinâmica de nossa rede, juntamente com a preocupação acerca do tráfego, inspiraram o segundo aprimoramento, relativo ao *tempo*. Como visto, o intervalo de tempo entre dois *broadcasts* não tem um valor fixo. Ao contrário, ele dobra toda vez que termina um ciclo de tentativas de se estabelecer conexões, diminuindo o tráfego geral. Além disso, considerando-se que esteja difícil de se conectar a outros nós, enquanto o nó aguarda por este intervalo mais longo, a rede pode assumir uma configuração mais favorável, tornando mais fácil estabelecer as conexões desejadas. Um detalhe não apresentado no pseudo-código é que, sempre que uma conexão é feita, o valor de *tempo* retorna ao seu valor inicial. Isto é feito porque esta nova conexão pode significar uma melhor configuração da rede.

O terceiro aprimoramento é, ao testar se uma conexão está ativa, verificar também a distância entre os nós. Uma vez que a rede ad hoc pode ter grande mobilidade, limitar

a distância entre os *peers* é muito importante para favorecer as conexões locais. Estar conectado a vizinhos próximos diminui o número global de *pings* e de *pongs* que trafegam na rede. Com este mesmo intuito de diminuir a quantidade de *pings* e *pongs* fez-se o quarto e último aprimoramento, que é a verificação da conexão por apenas um dos seus vértices, cuja viabilidade se deve à simetria das conexões.

4.2.2 Algoritmo Aleatório

Adotando o algoritmo *Regular*, cada nó se conectaria preferencialmente aos seus vizinhos mais próximos. Desta forma, em uma rede P2P densa, as conexões seriam estabelecidas dentro de um pequeno número de passos. Isto provavelmente levaria a uma rede com características semelhantes àquelas de um *grafo regular*, especialmente no tocante a caminhos mínimos longos. Com o propósito de evitar isto e de se ganhar características de *small-world*, nosso algoritmo *Regular* sofreu uma pequena modificação, tornando-se o algoritmo *Aleatório*.

O estabelecimento das primeiras $\mathsf{MAXNCON} - 1$ conexões segue os mesmos passos mencionados no algoritmo *Regular*. Por esta razão, elas serão chamadas de conexões *regulares*. A manutenção de todas as conexões existentes, sejam elas *regulares* ou não, também é idêntica ao algoritmo anterior —ela segue o mesmo esquema mostrado na figura 4.5. A diferença entre os dois algoritmos reside na última conexão, como pode ser visto no algoritmo da figura 4.7.

Como visto, pequenas mudanças nas conexões podem transformar um *grafo regular* em um *grafo small-world*. Para promover esta mudança, o nó não tenta estabelecer sua última conexão a *npassos* de distância. Ao invés disto, ele escolhe um número aleatório —*passosaleat*— entre *npassos* e $2 \times \mathsf{MAXNPASSOS}$. Ele então faz o *broadcast* de uma mensagem à procura de conexões para todos os nós em um raio de *passosaleat*. O nó aguarda a chegada de respostas por um certo tempo, analisa as respostas recebidas, e continua o *three-way handshake* somente com o vizinho mais distante. Uma conexão estabelecida desta forma é chamada de conexão *aleatória* e, sempre que ela for desfeita, deverá ser substituída por outra conexão *aleatória*.

O efeito final esperado é que algumas das conexões totais liguem *peers* distantes e, desta forma, atuem como *pontes*, tornando o caminho mínimo mais curto ao mesmo tempo que mantêm o coeficiente de agrupamento alto. Com isto, esperamos alcançar o efeito *small-world*.

Algoritmo Aleatório: Estabelecimento de Conexões

enquanto o nó pertence à rede P2P
  se número de conexões $<$ MAXNCON **então**
    se $npassos \neq 0$ **então**
      tente estabelecer conexões ***novas*** e ***simétricas*** com nós num raio de *npassos* até o limite de MAXNCON $-$ 1 conexões ***regulares***;
      espere *tempo* antes da próxima tentativa;
    senão
      $tempo = \min(2 \times tempo, \text{MAXTEMPO})$;
    fim se
    se uma conexão ***aleatória*** é necessária **então**
      atribua a *passosaleat* um valor aleatório escolhido entre *npassos* e $2 \times$ MAXNPASSOS;
      tente estabelecer ***uma*** conexão ***aleatória nova*** e ***simétrica*** ao nó mais distante possível num raio de *passosaleat*;
    fim se
    $npassos = (npassos + 2) \bmod (\text{MAXNPASSOS} + 2)$;
  fim se
fim enquanto

Figura 4.7: Algoritmo Aleatório.

4.2.3 Algoritmo Híbrido

Uma rede ad hoc pode ser formada por tipos diferentes de dispositivos. Nesse caso, os esforços da rede devem se concentrar naqueles dispositivos que possuem maior poder computacional, visando prolongar o tempo de vida da rede como um todo. Ao contrário dos algoritmos anteriores, o algoritmo *Híbrido* foi projetado para ser utilizado naquelas que chamamos de redes heterogêneas, ou seja, redes onde os nós são diferenciados por algum *qualificador*. Esse qualificador pode representar qualquer característica do poder computacional do dispositivo como, por exemplo, o nível de energia.

O princípio do algoritmo é formar sub-redes com um *mestre* (intermediador) e um número limitado de *escravos*. Cada escravo se comunica apenas com seu respectivo mestre; mestres, porém, podem comunicar entre si, formando a rede híbrida. Para obter essa

configuração, o algoritmo *Híbrido* (descrito na figura 4.8) define diversos estados em que um *peer* pode estar em um determinado instante do tempo: MESTRE, ESCRAVO, RESERVADO e INICIAL. Cada *peer* inicia sua execução no estado INICIAL e, posteriormente, migra para o estado MESTRE ou para ESCRAVO. O estado RESERVADO é utilizado apenas na transição de estado INICIAL para ESCRAVO.

O algoritmo opera da seguinte maneira: inicialmente, cada *peer* tenta contactar outros que estejam a um certo limite (NPASSOS_INICIAL) de distância. Se não houver respostas, o limite é dobrado e tenta-se um novo contato. Eventualmente, se o limite alcançar MAXNPASSOS e nenhuma conexão tiver sido estabelecida, então o *peer* se intitula um mestre e utiliza o algoritmo *Regular* para contactar outros mestres.

O tipo de mensagem utilizada nesta primeira fase (descoberta) se denomina *captura* e contém uma importante informação – o *qualificador* do remetente. Se algum *peer* no estado INICIAL e com menor *qualificador* receber essa mensagem, ele tentará, através de um *3-way-handshake*, tornar-se um escravo do remetente da mensagem. Se o *qualificador* do receptor for maior e seu estado for tanto MESTRE quanto INICIAL, então ele enviará uma mensagem de captura de volta para o remetente. Esse passo garante que um novo *peer* sempre obtenha uma resposta ao entrar na rede, seja descobrindo os mestres já presentes ou outros nós em estado de inicialização. Os *peers* nos estados ESCRAVO e RESERVADO não respondem a mensagens de captura, pois se comunicam apenas com seus mestres ou candidatos a mestre, respectivamente.

A segunda fase (manutenção) é descrita na figura 4.9, e assemelha-se àquela do algoritmo *Regular*: periodicamente, o nó envia uma mensagem *ping* aos seus vizinhos e, se a resposta não é recebida em um certo intervalo de tempo ou se o vizinho está muito distante, o nó desfaz aquela conexão. Se o nó é um ESCRAVO, ele retorna seu *estado* para INICIAL. Se o nó for um MESTRE que ficou sem escravos, ele também retorna seu *estado* para INICIAL. Em ambos os casos, após retornarem para INICIAL, os *peers* tentam se conectar a outros nós.

A reconfiguração da rede híbrida também ocorre em outra situação. Alguns nós tornam-se mestres mesmo sem ter escravos; quando passam um certo período de tempo ainda sem conseguir obter nenhum escravo, eles retornam ao estado INICIAL. Desta forma, permite-se que tais nós, potencialmente, sejam escravos de outros mestres.

Algoritmo Híbrido: Estabelecimento de Conexões

```
enquanto este nó pertence à rede P2P
  escolha estado
   caso INICIAL:
     se npassos ≠ 0 então
       tente encontrar um mestre e tornar-se seu escravo ou
       tente encontrar escravos e tornar-se mestre deles;
     senão /* completou um ciclo */
       mude seu estado para MESTRE;
     fim se
     npassos = (npassos + 2) mod (MAXNPASSOS + 2);
   quebra;

   caso MESTRE:
     use o algoritmo Regular para contactar outros mestres;
     se nescravos < MAXNESCRAVOS então
       aceite qualquer conexão de escravo que chegar;
     fim se
     se o peer está sem escravos durante MAXTEMPOMESTRE então
       mude seu estado para INICIAL;
     fim se
   quebra;

   caso ESCRAVO:
     mantenha a conexão com o mestre;
   quebra;
  fim escolha
fim enquanto
```

Figura 4.8: Algoritmo Híbrido: Estabelecimento de Conexões.

Algoritmo Híbrido: Manutenção de Conexões Estabelecidas

```
enquanto esta conexão existe
  envie um ping para o nó conectado e espere algum tempo por um pong;
  se um pong foi recebido então
    se dist_nos < MAXDIST então
      espere algum tempo para enviar o próximo ping;
    senão
      feche esta conexão;
      não responda à mensagem ping oriunda do nó previamente conectado;
      se estado = ESCRAVO então
        mude seu estado para INICIAL;
      fim se
      se estado = MESTRE então
        se nescravos = 0 então
          mude seu estado para INICIAL;
        fim se
      fim se
  senão /* pong não foi recebido*/
    feche esta conexão;
    não responda ao ping oriundo do nó previamente conectado;
    se estado = ESCRAVO então
      mude seu estado para INICIAL;
    fim se
    se estado = MESTRE então
      se nescravos = 0 então
        mude seu estado para INICIAL;
      fim se
    fim se
  fim se
fim enquanto
```

Figura 4.9: Algoritmo Híbrido: Manutenção de Conexões.

Capítulo 5

Resultados

Os algoritmos descritos no capítulo anterior foram implementados na linguagem C++ e testados através de simulação utilizando-se o *Network Simulator ns-2* [20], que provê grande suporte à construção de modelos de redes ad hoc. A máquina utilizada para os testes possui processador de 2.4 GHz e 2.0 GB de memória. O tempo de simulação é de 2.400s e intervalo de confiança mínimo de 95%.

5.1 Modelo

O modelo proposto neste trabalho não visa nenhuma aplicação P2P específica; ele constitui um algoritmo genérico de intercâmbio de dados em redes P2P auto-configuráveis. Além deste esclarecimento, é importante reafirmar que o termo *conexão* na verdade significa *referência*. Em outras palavras, cada nó mantém uma lista de endereços de outros nós vizinhos, aos quais ele se considera "conectado". Portanto uma conexão simétrica é uma em que um nó A referencia B e B mantém A em sua lista de referências. Este tipo de conexão é utilizada nos algoritmos otimizados *Regular, Aleatório e Híbrido*. Conexões assimétricas também existem e são utilizadas no algoritmo *Básico*. Conexões são criadas pela aplicação P2P com o objetivo de criar caminhos para busca e, em alguns casos, troca de informação.

Para os algoritmos descentralizados, foi considerada uma rede ad hoc homogênea, isto é, composta por nós que possuem as mesmas funcionalidades, sendo capazes de agir identicamente como *servents*. Para o algoritmo Híbrido, foi considerada uma rede heterogênea, ou seja, uma rede na qual cada nó é capaz de agir tanto como mestre quanto como escravo – seu papel em uma conexão é determinado por um *qualificador*, que pode ser qualquer característica, e, inclusive, variar ao longo do tempo. Em nossas simulações, o qualificador se

relaciona com o nível de energia inicial do nó, e pode possuir cinco valores distintos. Neste caso, pode-se dizer que os nós podem apresentar as mesmas funcionalidades, porém sob diferentes condições ou períodos de tempo. Os qualificadores foram considerados estáticos em nossas simulações, porém os algoritmos projetados não impedem o uso de qualificadores dinâmicos (por exemplo, nível de energia). Em ambas as redes, cada nó possui um identificador único que o endereça.

5.2 Cenários

Em todas as simulações adotou-se um número de 150 usuários, dos quais 75% pertencem à rede P2P e os demais 25% atuam somente na rede ad hoc. A área de simulação corresponde a um quadrado de 100m de lado e o alcance de cada nó foi estipulado em 10m. O tempo de simulação é de 2400s. Na tabela 5.1 são apresentados estes e outros parâmetros comuns a todos os cenários experimentados.

Foram testadas situações estáticas e dinâmicas, ou seja, em algumas simulações os nós se movem e, em outras, não. Em ambos os cenários, inicialmente os nós são posicionados aleatoriamente sobre a área seguindo uma distribuição uniforme. Nos cenários dinâmicos, após o posicionamento, utiliza-se o modelo de mobilidade *Random Waypoint Mobility Model* [21]. Este modelo inclui tempos de pausa entre mudanças de sentido e velocidade, ou seja, o nó intercala períodos de movimentação e de repouso. Adotou-se 1.0 m/s como velocidade máxima e 900 s como tempo máximo de pausa.

Além dos cenários estáticos e dinâmicos, variou-se também o número de conexões entre os nós e a energia inicial dos dispositivos. Com relação ao nível de energia, foram feitos alguns testes em que se observou o consumo total de energia dos nós. A partir disso, pode-se escolher entre níveis de energia que proporcionariam ou não a sobrevivência de todos os nós durante toda a simulação. Assim, no cenário sem morte de nós, a energia inicial é bem superior que aquela do cenário com morte de nós. Esta diferença deve ser lembrada durante a análise dos gráficos apresentados na seção 5.3, pois implicam na utilização de diferentes escalas no eixo que representa o nível de energia.

Como foi dito anteriormente, na seção 5.1, os nós das simulações do algoritmo Híbrido são divididos em classes. Cada classe é caracterizada por um *qualificador*, o qual se relaciona com a energia inicial atribuída ao dispositivo. Nos demais algoritmos, todos os nós começam a simulação com a mesma energia. Nas simulações realizadas, foram definidos cinco grupos de nós no algoritmo Híbrido, a energia inicial que cada grupo recebeu foi cal-

Parâmetros	Valor
área de simulação	100 m x 100 m
alcance de transmissão	10 m
número de arquivos distintos	20
freqüência do arquivo mais popular	40%
TTL para consultas	6 P2P hops
NPASSOS (Algoritmo *Básico*)	6 ad hoc hops
NPASSOS_INICIAL	2 ad hoc hops
MAXNPASSOS	6 ad hoc hops
MAXDIST	6 ad hoc hops
MAXNESCRAVOS	3
energia inicial média (cenário **sem** morte de nós)	100 Joules
energia inicial média (cenário **com** morte de nós)	15 Joules

Tabela 5.1: Parâmetros comuns utilizados e seus valores.

culada de forma que a energia inicial média fosse igual a dos demais algoritmos, permitindo uma comparação mais fidedigna.

Durante a análise preliminar dos resultados obtidos, verificou-se a importante participação do consumo de energia com roteamento de mensagens em relação ao gasto total de energia. Diante disto, resolveu-se variar, também, o algoritmo de roteamento adotado.

A figura 5.1 ilustra a combinação de parâmetros que caracterizaram os cenários simulados, os quais foram executados repetidamente para cada um dos quatro algoritmos implementados, garantindo um intervalo de confiança maior que 95%.

5.2.1 Modelo de Consultas

O processo de consulta foi inspirado no Gnutella. No caso dos algoritmos descentralizados, um nó envia sua consulta para todos os outros nós presentes em sua lista de vizinhos. Já no caso do algoritmo híbrido, há duas situações: caso o nó seja um escravo, a consulta é enviada para seu respectivo mestre; caso contrário, é enviada para seus vizinhos e escravos. Ao receber uma consulta, o nó a processa, decrementa o número de passos para aquela consulta, e a envia para seus vizinhos seguindo o modelo acima, caso o número de passos

Figura 5.1: Cenários simulados.

ainda permita.

A fim de limitar o número de mensagens que trafegam na rede, foram estipuladas as seguintes regras:

- Um nó não envia uma consulta para o nó de origem daquela consulta.
- Um nó não envia a consulta para o nó que a enviou no passo anterior.
- Um nó só responde a uma consulta uma única vez.

Caso o nó tenha o arquivo requisitado, a resposta é enviada diretamente para o nó de origem. Após enviar uma consulta, o nó espera durante um período de 20s para a chegada de respostas. Em seguida, espera-se um tempo aleatório entre 10 e 20s para o envio de uma nova consulta.

A distribuição de consultas por arquivo segue a lei de Zipf [22], assim como a distribuição de arquivos por nó. A freqüência máxima de presença de um arquivo –$MAXFREQ$– foi determinada em 40%, o que significa dizer que o arquivo mais popular estará presente em 40% dos nós e representará 40% de todas requisições realizadas. O segundo arquivo mais popular terá freqüência de $\frac{40}{2} = 20\%$, o terceiro, $\frac{40}{3}$ e assim por diante.

5.2.2 Métricas

Durante as simulações, foram coletados vários tipos de dados, como energia consumida, número e tipo de mensagens trocadas, posicionamento do nó na área, consultas realizadas, respostas obtidas, número de vizinhos, etc. A diversidade dos dados, somada à existência de vários cenários e ao fato de estarem sendo analisados **quatro** algoritmos, fez com que fossem selecionadas métricas capazes de sintetizar o comportamento geral dos algoritmos. A escolha visou demonstrar especialmente os aspectos de conservação de energia, eficiência na busca e distribuição de carga na rede, como pode ser visto na descrição de cada métrica:

- Índice de sucesso –*hits*– das consultas: representa a porcentagem média de sucesso das consultas realizadas pela aplicação P2P. Esta métrica permite avaliar o custo dos aprimoramentos feitos nos algoritmos otimizados, observando o impacto causado por eles no desempenho do protocolo P2P.

- Energia média da rede: representa a média da energia residual de todos os nós da rede em um dado instante, permitindo que se acompanhe a evolução do gasto de energia à medida em que a rede evolui. Com esta métrica, podemos analisar o benefício alcançado pelos algoritmos otimizados com relação à sobrevida da rede. Durante as simulações, foram coletados dados sobre a energia dos nós a cada 50s, obtendo-se 48 "fotografias" da energia da rede.

- Composição média do gasto de energia: divide a energia total despendida em três grupos (*Relacionadas a Roteamento*, *Aplicação P2P* e *Algoritmo de (re)configuração*), considerando-se com qual o tipo de mensagem recebida ou enviada ocorreu o gasto de energia. A categorização das mensagens será melhor explicada a seguir.

As simulações dos algoritmos envolvem o tráfego de diversos tipos de mensagens: consultas P2P[1], mensagens dos algoritmos de (re)configuração (à procura de conexões, *pings*, *pongs*, *3-way-handshake* etc), mensagens da camada de rede (*route request* (RREQ), *route reply* (RREP)) e das camadas inferiores. As mensagens da camada de rede e inferiores foram reunidas em um só grupo por todas servirem ao roteamento. As mensagens dos algoritmos de (re)configuração foram divididas em dois grupos: aquelas que eram originadas ou destinadas ao próprio nó que a enviou ou recebeu; e aquelas que somente eram repassadas pelo nó, ou seja, quando ele desempenhava seu papel de roteador da rede ad hoc. As mensagens da rede P2P também foram separadas em dois conjuntos seguindo este mesmo

[1] A transferência dos arquivos não foi considerada.

critério. Assim, o gasto de energia do nó atuando como roteador se uniu ao gasto com mensagens de roteamento, formando o conjunto de mensagens chamado de *Relacionadas a Roteamento*.

5.3 Resultados

Esta seção, dividida em duas partes, descreve os resultados obtidos com as simulações dos cenários apresentados. A primeira subseção apresenta os resultados dos experimentos que utilizaram o AODV como protocolo de roteamento. Na segunda subseção, tem-se os testes nos quais se passou a utilizar o algoritmo DSR para roteamento.

5.3.1 Resultados AODV

Nesta seção, serão observados os dados coletados nas simulações que utilizaram o AODV como protocolo de roteamento. O primeiro aspecto analisado será a energia média da rede ao longo da simulação.

A figura 5.2 ilustra o resultado do cenário estático, com número máximo de conexões igual a 3 e com morte de nós. Percebe-se um comportamento bastante semelhante entre os algoritmos Regular e Aleatório, com uma pequena diferença em favor do primeiro. O algoritmo Básico teve o maior consumo de energia, aproximadamente 9% a mais que o Aleatório. Além de apresentar a maior conservação de energia ao longo do tempo, percebe-se que o algoritmo Híbrido possui uma curva sutilmente diferente das demais, demonstrando um comportamento mais linear na segunda metade do intervalo de tempo. Em todos os casos, constata-se que a taxa do gasto de energia é maior nos primeiros instantes de vida da rede, caracterizando o momento em que ela se forma.

O cenário com os mesmos parâmetros de mobilidade (estático) e número máximo de conexões (três), porém sem morte de nós, tem seus resultados mostrados na figura 5.3. O comportamento relativo dos algoritmos permanece igual ao cenário com morte: o Híbrido possui melhor desempenho, seguido pelo Regular, Aleatório e Básico. Sem a morte de nós, entretanto, a diferença de consumo entre o algoritmo Híbrido e os demais diminuiu. Este comportamento provavelmente indica que a diferenciação entre mestres e escravos propicia economia maior de energia para *estabelecer*, e não para *manter* conexões. Desta forma, em um cenário menos dinâmico (sem morte de nós) a economia de energia é menor. Os algoritmos Regular e Aleatório, por sua vez, passaram a exibir um comportamento mais diferenciado. Em todos os casos, as curvas deste novo cenário se mostram muito

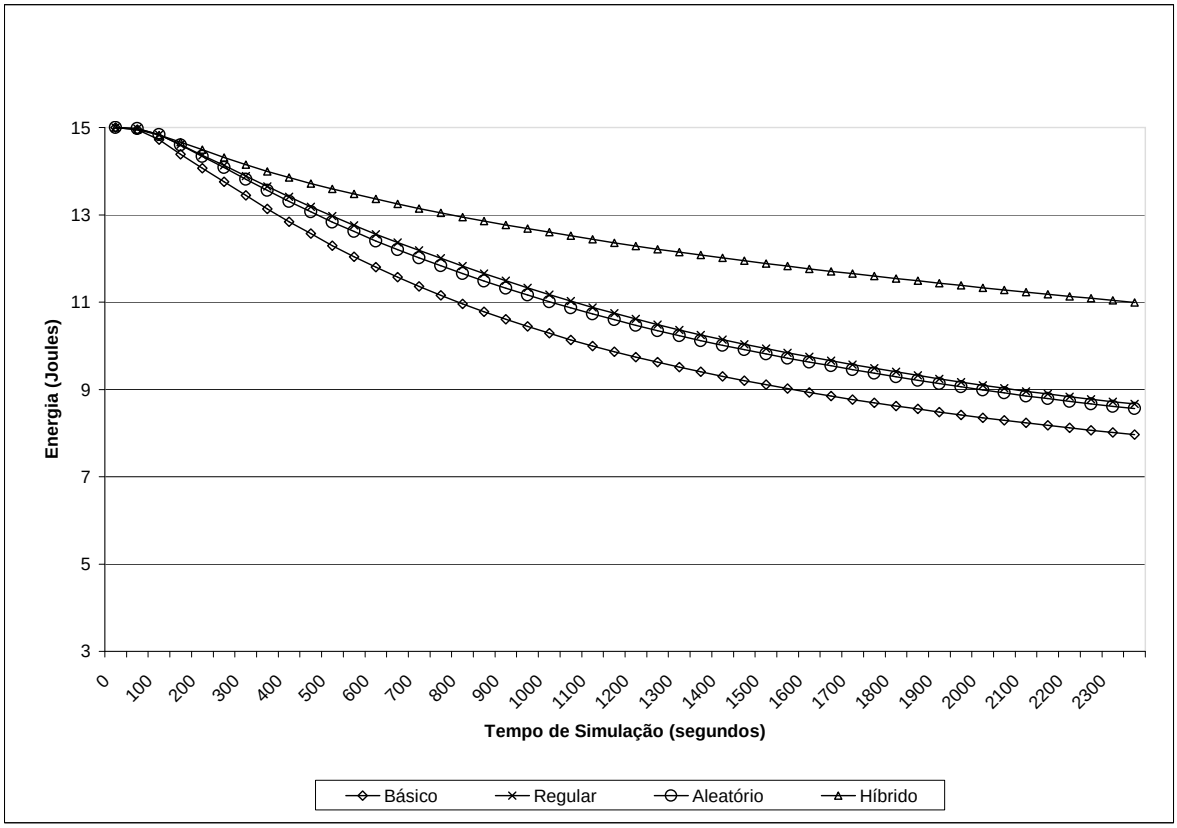

Figura 5.2: Energia média da rede ao longo da simulação (3 conexões, estático, AODV, com morte de nós).

mais lineares que no cenário anterior. Esta linearidade significa um consumo de energia mais homogêneo ao longo do tempo. Estima-se que isto se deve à falta de dinamicidade do cenário, que, além de estático, não apresenta morte de nós. Nesta situação, após estabelecer as conexões possíveis, o nó repete até o fim da simulação o mesmo processo de manutenção das conexões existentes e de procura das conexões faltantes. Ou seja, mantém sua rotina de gasto de energia.

As figuras 5.4 e 5.5 também se referem ao cenário estático, respectivamente com e sem morte de nós, porém com número máximo de conexões igual a 6. Na primeira, é possível observar a grande semelhança com o cenário equivalente com 3 conexões: o algoritmo Híbrido destaca-se positivamente com relação aos demais, o Regular e Aleatório praticamente se sobrepõem e o Básico novamente exibe o pior comportamento. Naturalmente, o cenário com máximo de 6 conexões apresenta um consumo de energia bem superior ao de 3 conexões.

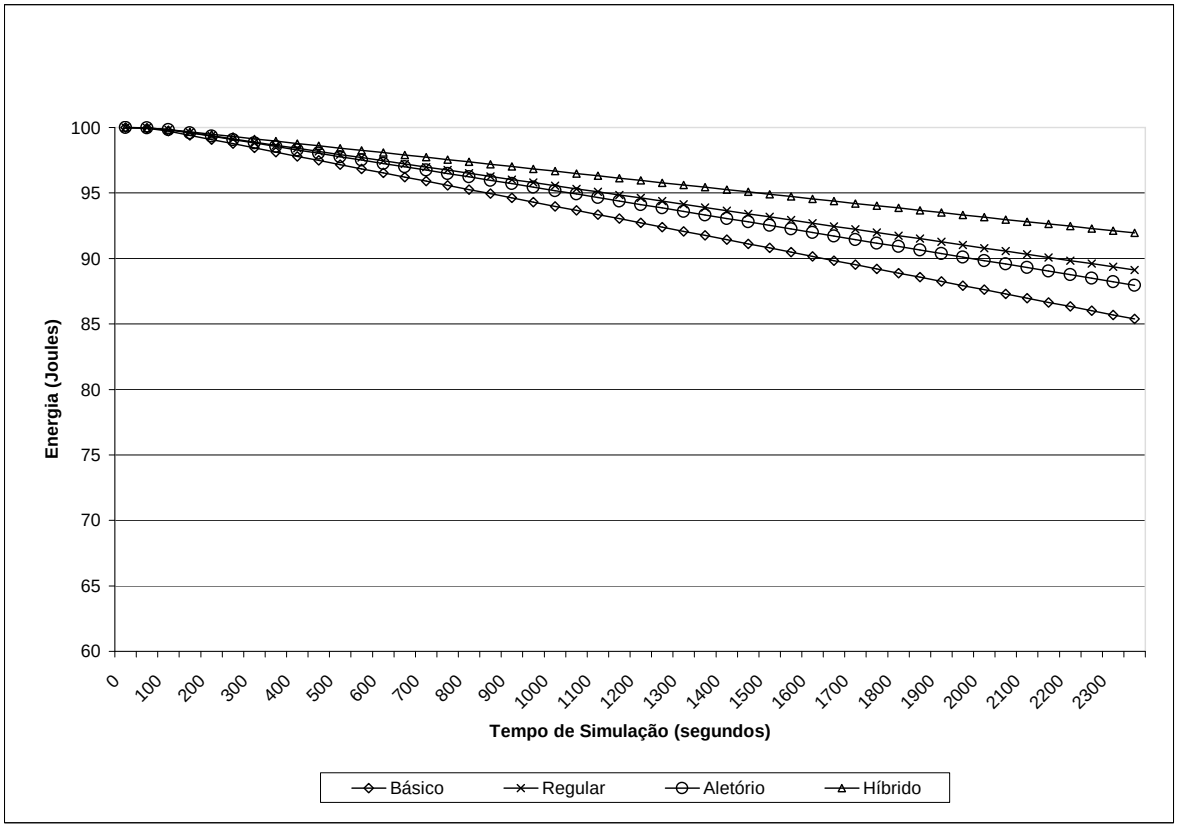

Figura 5.3: Energia média da rede ao longo da simulação (3 conexões, estático, AODV, sem morte de nós).

Novamente, no cenário com morte de nós —mostrado na figura 5.4— constata-se que a taxa do gasto de energia é maior nos primeiros instantes de vida da rede, especialmente no terço inicial das simulações, quando há a predominância do gasto com a formação inicial da rede. A figura 5.5, sem morte de nós, mostra que, apesar do consumo maior, as curvas exibem o mesmo comportamento do cenário com máximo de 3 conexões. A manutenção do comportamento geral é um indicativo da escalabilidade dos algoritmos propostos.

O próximo conjunto de gráficos —figuras 5.6 a 5.9, ilustra a energia média da rede ao longo das simulações em cenários dinâmicos, ou seja, com movimentação dos nós pela área. Esta situação implica em uma maior efemeridade das conexões, pois os elementos estão constantemente entrando e saindo do raio de alcance uns dos outros.

Em todos os cenários dinâmicos, a ordenação dos algoritmos com relação ao seu consumo de energia permanece a mesma do cenário estático: começando pelo que menos consome, tem-se o Híbrido, Regular, Aleatório e Básico.

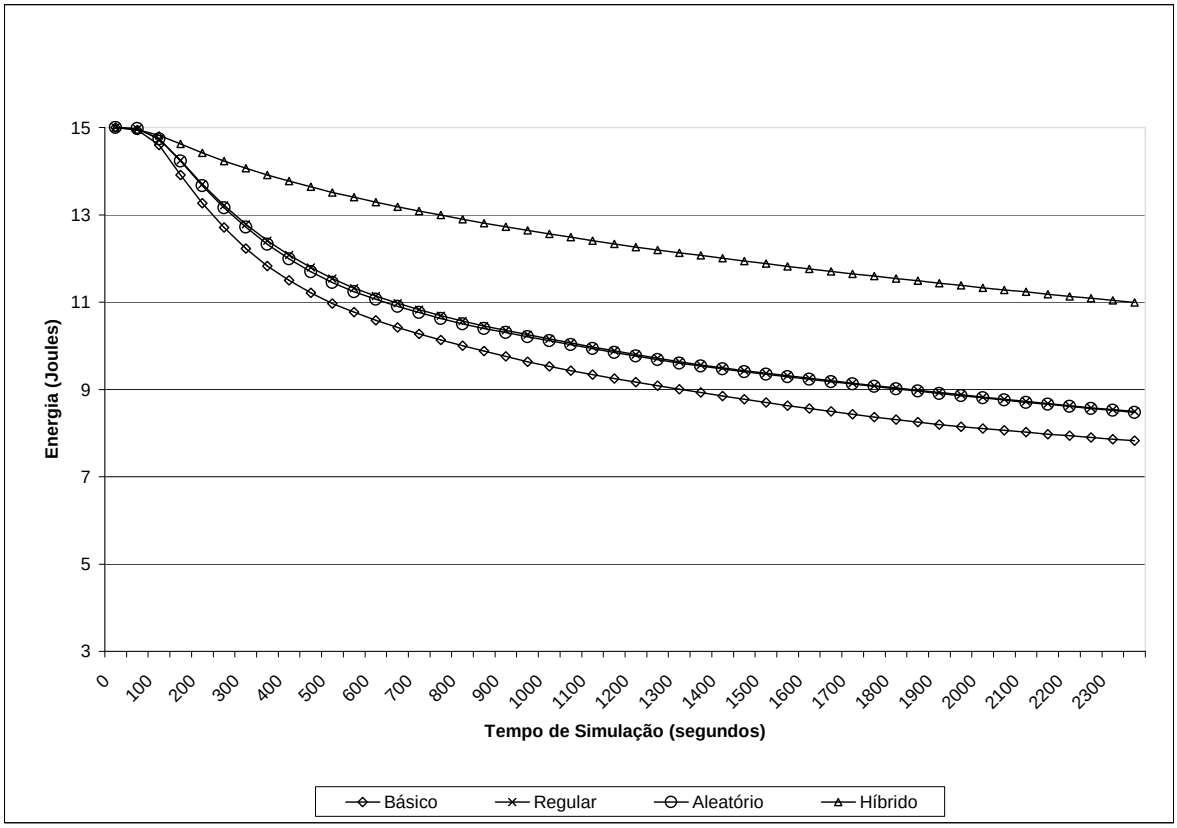

Figura 5.4: Energia média da rede ao longo da simulação (6 conexões, estático, AODV, com morte de nós).

Nos testes com número máximo de 3 conexões, com morte de nós —figura 5.6, a inserção de movimentação fez com que o algoritmo Híbrido apresentasse um desempenho um pouco mais semelhante aos demais, ao passo que causou um maior distanciamento entre as curvas do Regular e do Aleatório. Além disso, não há mais um consumo elevado de energia no início das simulações. Ou seja, não há mais uma fase inicial de formação da rede: a dinamicidade faz com que a rede esteja constantemente se reconfigurando.

Em todos os algoritmos, a energia média final da rede é bem inferior àquela verificada no cenário estático, o que confirma o cenário dinâmico como o pior caso sob o aspecto do consumo de energia.

A figura 5.7, também referente ao cenário dinâmico, exibe uma situação interessante. Ela ilustra os resultados das simulações com número máximo de 3 conexões, porém sem morte de nós, nos quais o comportamento do algoritmo Híbrido fica bastante semelhante ao do Regular e do Aleatório. Na verdade, o comportamento dos algoritmos Regular e

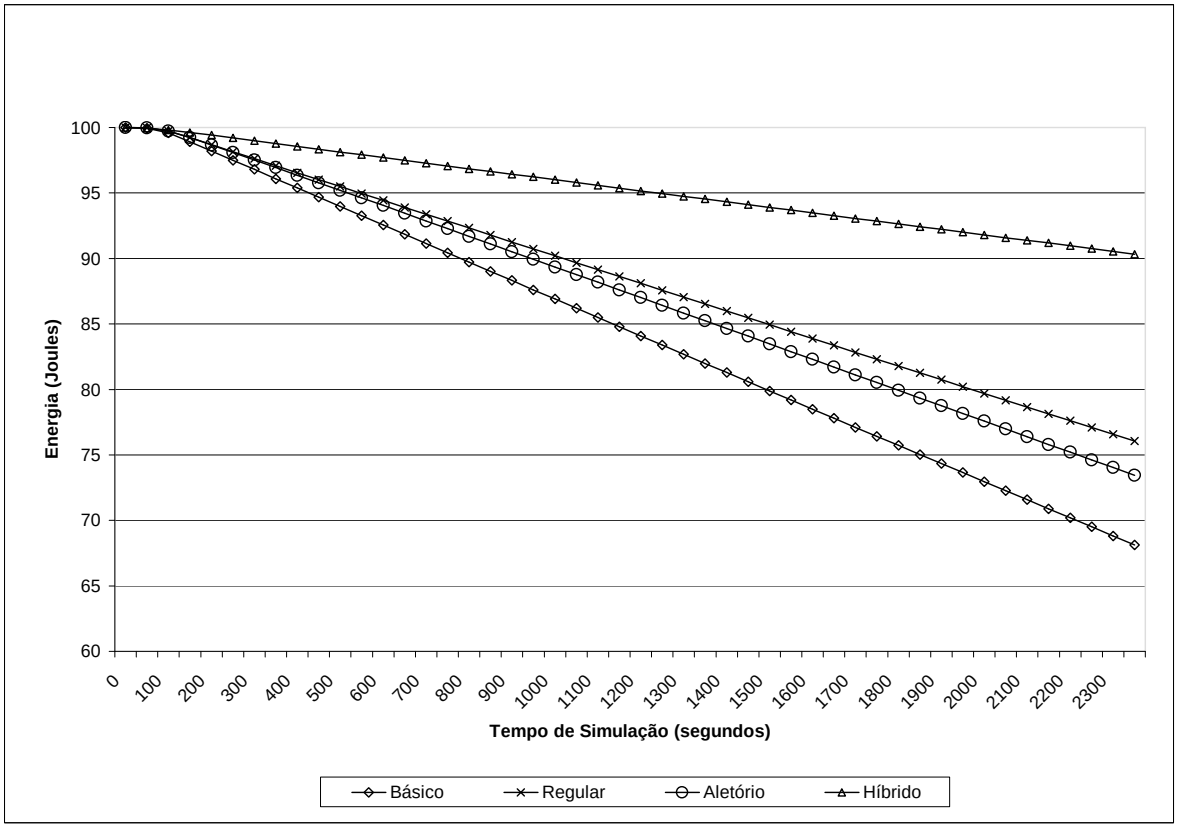

Figura 5.5: Energia média da rede ao longo da simulação (6 conexões, estático, AODV, sem morte de nós).

Aleatório continuaram bem semelhantes ao apresentado no cenário estático (3 conexões e sem morte de nós). A maior semelhança ocorreu por causa do aumento do consumo de energia do Híbrido. Isto demonstra que, para cenários sem morte de nós, o Regular e o Aleatório respondem melhor à inserção de movimentação que o Híbrido. O algoritmo Básico, por sua vez, apresenta um consumo bem mais elevado e sua curva posiciona-se isoladamente das demais.

Assim como no cenário estático, o consumo de energia mostra-se razoavelmente constante ao longo do tempo. Reforça-se, então, a diferenciação entre as curvas dos cenários sem morte e com morte de nós, sendo as primeiras com aspecto mais linear independentemente da mobilidade. Diante disto, pode-se estimar que a morte dos nós impacta mais o consumo de energia que a presença de mobilidade.

Os próximos cenários, com resultados mostrados nos gráficos 5.8 e 5.9, têm o número máximo de conexões igual a 6. No primeiro, com morte de nós, novamente o Híbrido

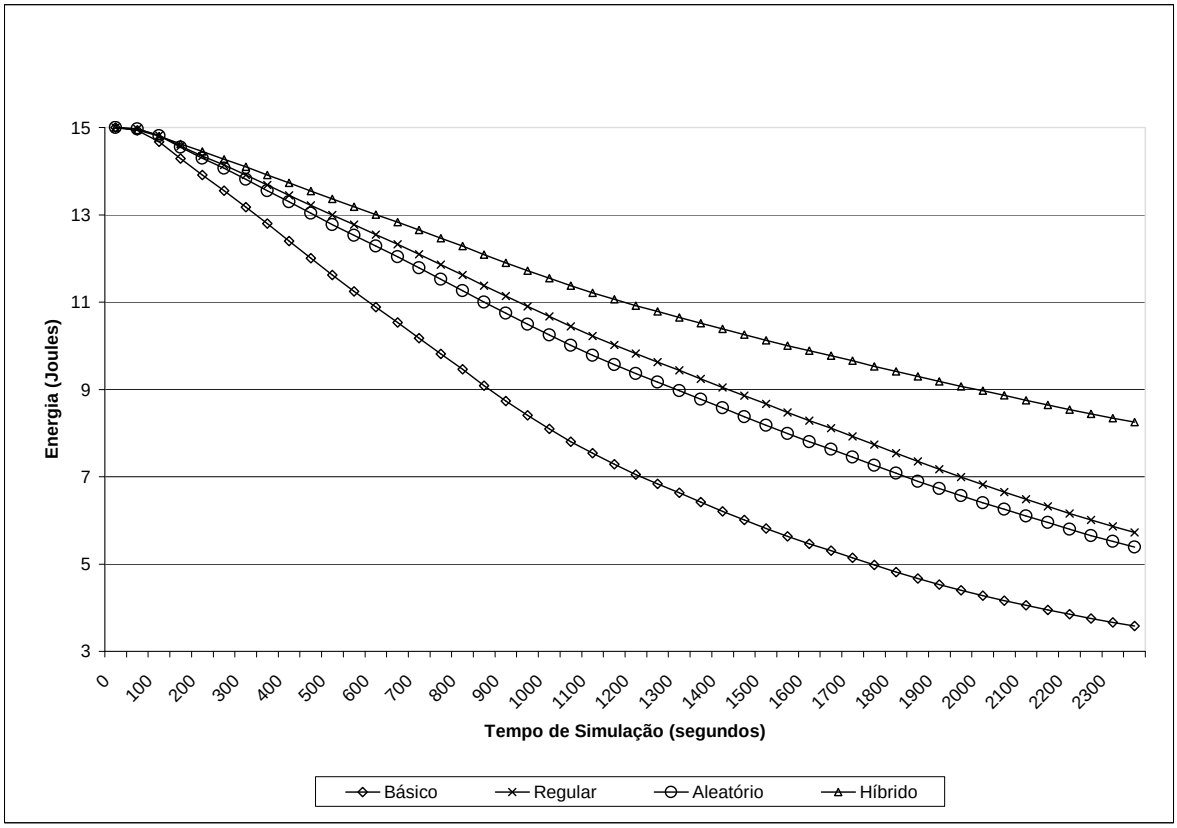

Figura 5.6: Energia média da rede ao longo da simulação (3 conexões, dinâmico, AODV, com morte de nós).

destaca-se dos demais algoritmos, chegando ao final da simulação com o dobro de energia média que o Regular e o Aleatório. Estes últimos, mais uma vez, demonstram um consumo praticamente igual durante todo o teste. O algoritmo Básico, tem o pior desempenho, mostrando uma taxa de consumo um pouco mais elevada na porção média do intervalo de tempo.

No segundo caso, gráfico 5.9, sem morte de nós, o comportamento do algoritmo Regular e do Aleatório, muito semelhantes entre si, mostram-se como uma média entre o Básico e o Híbrido. Outra vez, a diferença entre o Híbrido e os demais é considerável: o consumo médio foi aproximadamente metade daquele do algoritmo Regular. Entretanto, quando comparado ao cenário estático equivalente (6 conexões, sem morte de nós), percebe-se que os algoritmos Híbrido e Básico apresentam maior consumo de energia, enquanto que o Regular e Aleatório demonstram diminuição do consumo. Ou seja, o Híbrido continua apresentando melhor desempenho, porém numa escala menor.

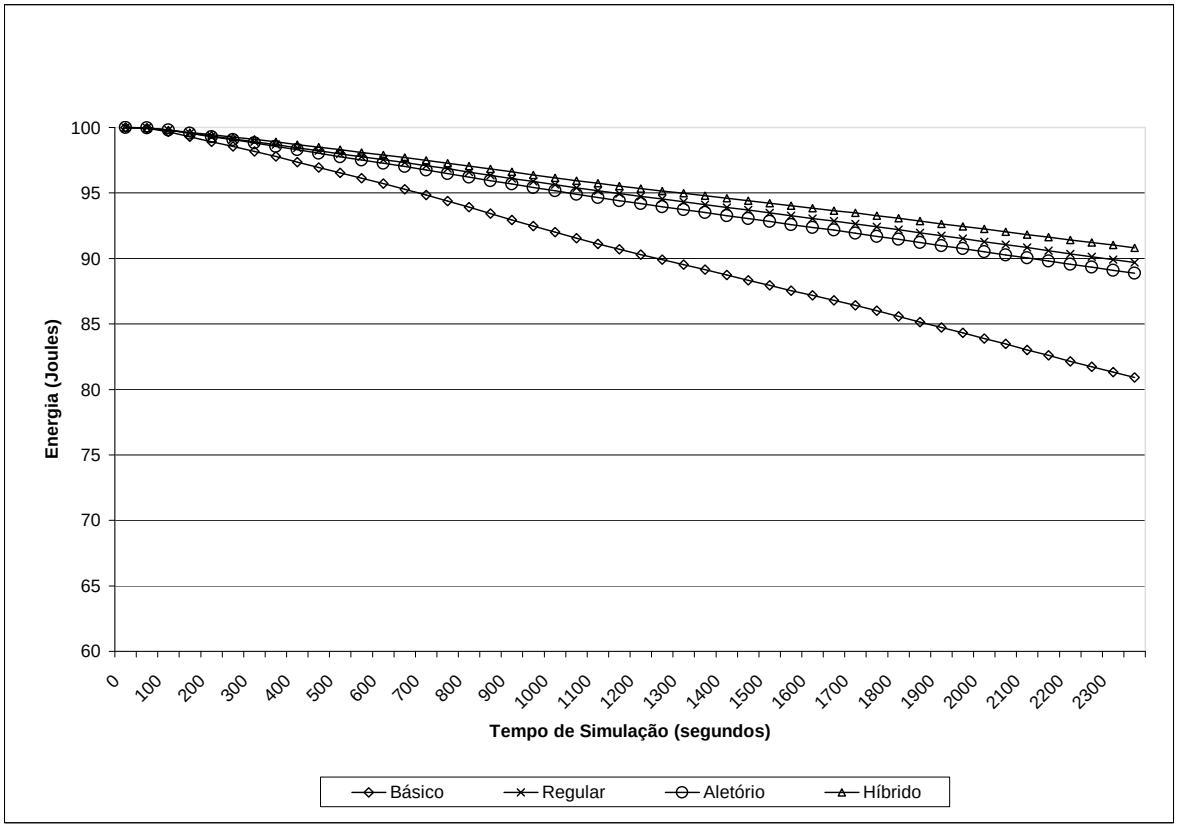

Figura 5.7: Energia média da rede ao longo da simulação (3 conexões, dinâmico, AODV, sem morte de nós).

Até este momento, observamos os gráficos relativos ao consumo médio de energia ao longo do tempo nos cenários que utilizam o AODV como algoritmo de roteamento. No total, foram 8 cenários distintos, nos quais variaram-se a movimentação dos nós (estático ou dinâmico), o número máximo de conexões (3 ou 6) e a energia inicial dos nós (com ou sem morte de nós).

A partir dos gráficos, pode-se fazer uma diferenciação clara entre os cenários com morte e sem morte de nós. Nos últimos, o consumo de energia é mais homogêneo ao longo do tempo, mesmo quando se tem mobilidade (dinâmico). Conclui-se, então, que nos cenários simulados a morte dos nós, ou seja, a diminuição no número de nós traz maior impacto no consumo de energia que a movimentação. Esta observação poderia indicar que cenários menos densos são mais críticos que os dinâmicos, sob o aspecto de o consumo de energia. Para a consolidação de tal afirmação, entretanto, seria importante avaliar variações na mobilidade, como aumento da velocidade média, por exemplo.

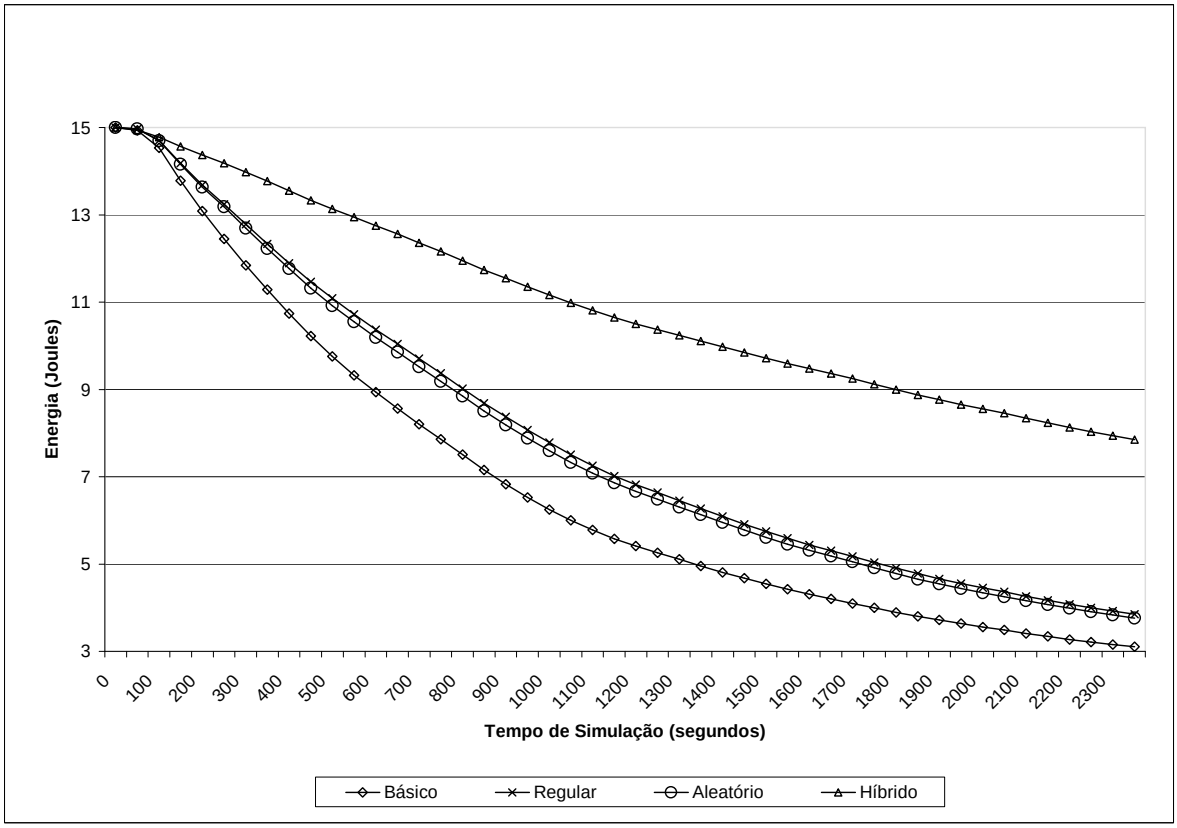

Figura 5.8: Energia média da rede ao longo da simulação (6 conexões, dinâmico, AODV, com morte de nós).

Ainda sobre a diferenciação de cenários, é interessante observar a alteração do comportamento dos algoritmos com a inserção de mobilidade dos nós. Para cenários com mesmo número de conexões e com morte de nós, percebe-se que a inserção de mobilidade ocasiona aumento no consumo de energia por todos os algoritmos. Esta mudança é esperada, uma vez que a movimentação torna o cenário mais dinâmico e com conexões potencialmente mais efêmeras.

Nos cenários sem morte de nós, a inserção de mobilidade gera maior consumo pelos algoritmos Híbrido e Básico. Os algoritmos Regular e Aleatório, entretanto, passam a exibir consumo de energia um pouco menor. Uma possível explicação para este fato pode ser que a mobilidade, apesar de causar rompimento de conexões, seja mais impactante na diminuição da possibilidade de um nó ficar isolado por muito tempo. Assim, há maior chance do nó gastar proporcionalmente mais energia com a manutenção de conexões do que com estabelecimento de novas conexões. Uma vez que a manutenção das conexões

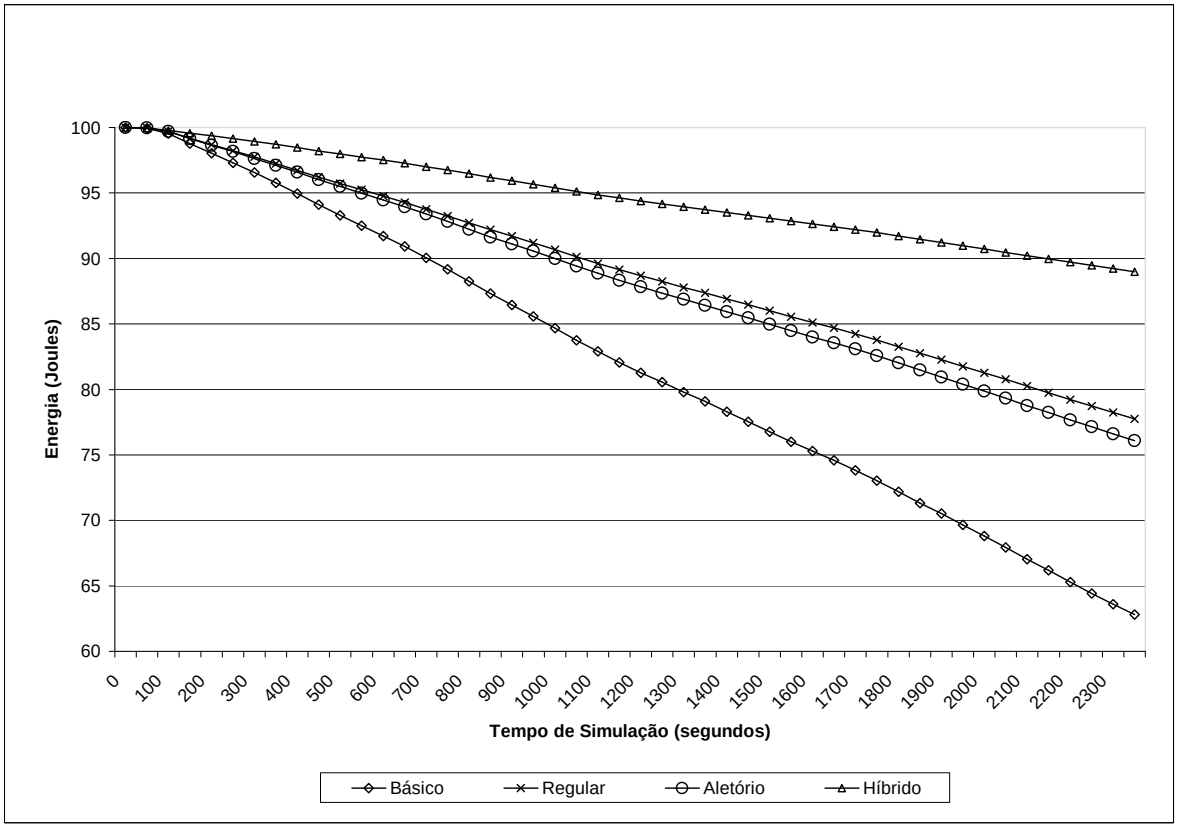

Figura 5.9: Energia média da rede ao longo da simulação (6 conexões, dinâmico, AODV, sem morte de nós.

destes dois algoritmos são testadas somente por um vértice, isto representa economia no consumo médio de energia pela rede.

Mesmo com algumas variações, em todas as simulações, o algoritmo Híbrido apresentou o menor consumo, que, de uma forma geral, foi bastante inferior ao dos demais algoritmos. Esta economia deve-se à estruturação hierárquica da rede, na qual os escravos possuem uma rotina que despende menos energia que os nós dos demais algoritmos. Os mestres, por sua vez, se assemelham aos nós dos demais algoritmos. Assim, caso o cenário seja favorável a esta hierarquização, na média, o consumo de energia tende a ser mais baixo.

Uma vez que a fonte de energia limitada é uma das grandes restrições do ambiente estudado (redes P2P sobre ad hoc), as informações sobre consumo contidas nesses gráficos nos permitem avaliar o *benefício* alcançado pelos algoritmos otimizados.

Os próximos gráficos, mostrados nas figuras 5.10 e 5.11, referem-se à taxa de sucesso das consultas. Ao observar-se a energia média da rede ao longo do tempo, pôde-se identificar o

benefício alcançado por nossas otimizações. Agora, comparando a porcentagem de sucesso das consultas, poder-se-á avaliar qual foi o custo de tais ganhos.

Nos cenários com morte —figura 5.10, é possível observar que o algoritmo Regular sempre apresenta percentuais de sucesso ligeiramente superiores àqueles do Aleatório. Estes algoritmos exibem pequena variação de desempenho entre os diversos cenários, apresentando-se como as opções mais vantajosas nas simulações com movimentação de nós. Sob o aspecto de sucesso das consultas, esperava-se que o algoritmo Aleatório proporcionasse taxas mais elevadas, pois foi projetado visando uma cobertura potencialmente maior. O efeito, entretanto, não foi alcançado. Desta forma, acredita-se que as conexões aleatórias não foram capazes de conectar os diversos grupos ou que, talvez, a rede não apresentou particionamento em grupos isolados, o que tira a eventual vantagem das conexões aleatórias. O algoritmo Híbrido, por sua vez, é aquele que apresenta o melhor desempenho quando os nós permanecem estáticos, mas, nos cenários dinâmicos, ele é o de pior desempenho quando o número máximo de conexões é 3. Quando o máximo de conexões passa para 6, o Híbrido mostra-se pouco melhor que o último colocado, o Básico. O algoritmo Básico também atinge a menor porcentagem de sucesso nas duas simulações estáticas. A discriminação dos percentuais é feita na tabela 5.2.

Nestes cenários com morte de nós, os percentuais de sucesso dos algoritmos otimizados (Híbrido, Aleatório e Regular) ficaram bem próximos entre si e sempre superiores às taxas do Básico. Este resultado é bem positivo, pois, como as conexões do Básico não são simétricas, elas poderiam representar uma cobertura maior e conseqüentemente proporcionarem maior sucesso das consultas.

O destaque do algoritmo Híbrido nos cenários estáticos provavelmente ocorreu porque a ausência de mobilidade torna propícia a formação e permanência de um grande grupo de mestres com grande capilarização através de seus escravos, o refletiria em aumento dos /textslhits.

Cenário	**Algoritmos de (Re)Configuração**			
	Básico	**Regular**	**Aleatório**	**Híbrido**
3 Conexões, Estático	30,93%	33,40%	33,05%	34,69%
3 Conexões, Dinâmico	29,94%	31,98%	31,68%	29,32%
6 Conexões, Estático	28,67%	30,38%	30,20%	34,36%
6 Conexões, Dinâmico	29,52%	31,64%	30,92%	29,64%

Tabela 5.2: Percentuais de sucesso (*hits*) das consultas (AODV, com morte de nós).

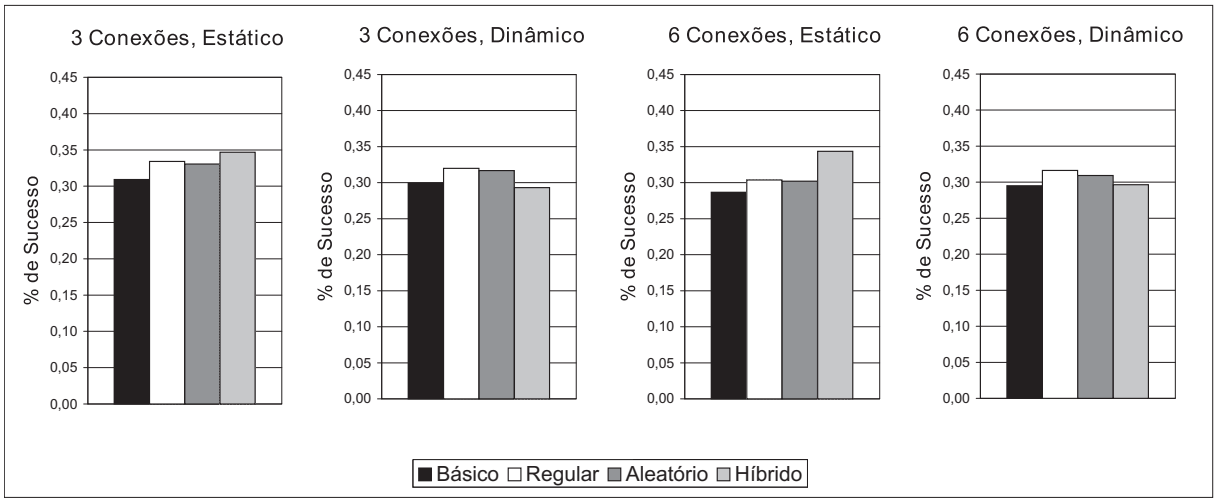

Figura 5.10: Porcentagem de sucesso (*hits*) das consultas (AODV, com morte de nós).

Nas simulações sem morte de nós, entretanto, o algoritmo Básico exibe uma melhora de desempenho no sucesso das consultas, como pode ser visto na figura 5.11. Nos cenários dinâmicos, ele apresenta os melhores percentuais de sucesso, sendo seguido pelo algoritmo Regular, Aleatório e Híbrido, respectivamente. Novamente, o comportamento dos algoritmos Regular e Aleatório é praticamente o mesmo em todas as situações, como pode ser melhor observado através dos valores discriminados na tabela 5.3.

O melhor desempenho do algoritmo Básico, esperado também para os cenários com morte de nós, acaba se manifestando nos cenários dinâmicos sem morte de nós. Além da potencial maior cobertura propiciada pelas conexões assimétricas, o Básico possui a característica de não controlar a distância até seus parceiros. Tendo em vista que os nós não morrem, após estabelecerem uma conexão, os nós estarão unidos enquanto houver caminho possível entre eles, independentemente do número de passos (*hops*). Desta forma, a mobilidade impacta pouco na manutenção das conexões, ou seja, uma conexão que seria desfeita nos demais algoritmos é preservada no Básico. Isto possibilita níveis de conectividade mais altos durante mais tempo, provavelmente resultando em maior disseminação e conseqüente sucesso das consultas.

Nos cenários estáticos, ainda sem morte de nós, o desempenho relativo entre os algoritmos mostra-se bastante diferente quando o número máximo de conexões passa de 3 para 6. No cenário de 3 conexões, o Básico é o pior colocado e o Híbrido, o melhor. O Regular e o Aleatório ficam na média entre os outros dois, havendo uma distinção clara entre

as 3 posições. Com 6 conexões, entretanto, o desempenho de todos fica praticamente o mesmo, como explicita a tabela 5.3. Neste último cenário, o não controle da distância dos parceiros já não representa vantagem para o Básico, pois não há mobilidade. Ao mesmo tempo, o elevado número de conexões estáveis permite ao Aleatório e ao Regular competirem com a capilaridade do Híbrido. Esta se mostra como a explicação mais provável para a semelhança das taxas.

Cenário	**Algoritmos de (Re)Configuração**			
	Básico	**Regular**	**Aleatório**	**Híbrido**
3 Conexões, Estático	38,36%	40,30%	40,44%	42,06%
3 Conexões, Dinâmico	34,37%	33,05%	33,04%	31,74%
6 Conexões, Estático	42,40%	42,76%	42,68%	42,29%
6 Conexões, Dinâmico	38,99%	38,23%	38,10%	32,80%

Tabela 5.3: Percentuais de sucesso (*hits*) das consultas (AODV, sem morte de nós).

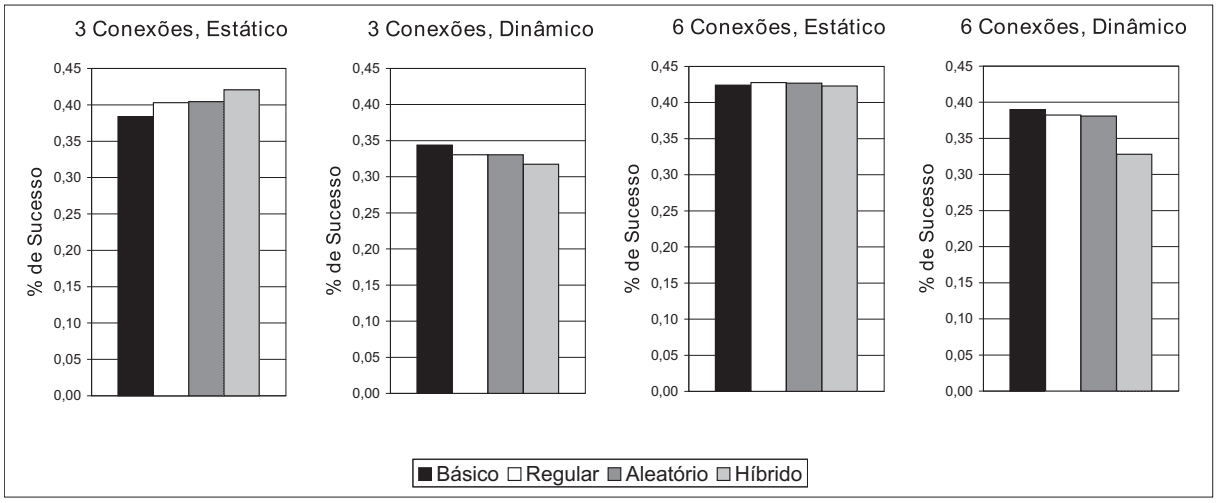

Figura 5.11: Porcentagem de sucesso (*hits*) das consultas (AODV, sem morte de nós).

Os gráficos 5.12 e 5.13 tratam do último aspecto observado: a composição média do gasto de energia. Durante a análise preliminar dos resultados, observou-se que os algoritmos otimizados atingiam ganhos de desempenho significativos referentes à troca de mensagens de (re)configuração. Entretanto, ao analisar as curvas da energia média gasta ao longo do tempo, os benefícios mostravam-se proporcionalmente menores. A fim de investigar melhor a causa desta diferença, foi feita a análise da composição do gasto da energia,

que evidenciou que a parcela de energia gasta com mensagens relacionadas a roteamento prevalecia sobre as demais.

Cenário	Tipo de Gasto	Algoritmos de (Re)Configuração			
		Básico	**Regular**	**Aleatório**	**Híbrido**
3 Con., Estático	Alg. (Re)Configuração	5,09%	2,47%	2,51%	2,86%
	Aplicação P2P	6,29%	7,99%	7,72%	7,35%
	Rel. a Roteamento	88,62%	89,55%	89,77%	89,79%
3 Con., Dinâmico	Alg. (Re)Configuração	3,04%	1,89%	1,92%	2,33%
	Aplicação P2P	3,74%	4,67%	4,35%	3,80%
	Rel. a Roteamento	93,22%	93,43%	93,73%	93,88%
6 Con., Estático	Alg. (Re)Configuração	5,57%	2,41%	2,40%	2,86%
	Aplicação P2P	5,94%	7,91%	7,79%	7,29%
	Rel. a Roteamento	89,68%	89,68%	89,80%	89,86%
6 Con., Dinâmico	Alg. (Re)Configuração	2,93%	1,82%	1,93%	2,23%
	Aplicação P2P	4,23%	5,32%	5,01%	3,91%
	Rel. a Roteamento	92,84%	92,86%	93,07%	93,86%

Tabela 5.4: Composição da energia gasta (AODV, com morte de nós).

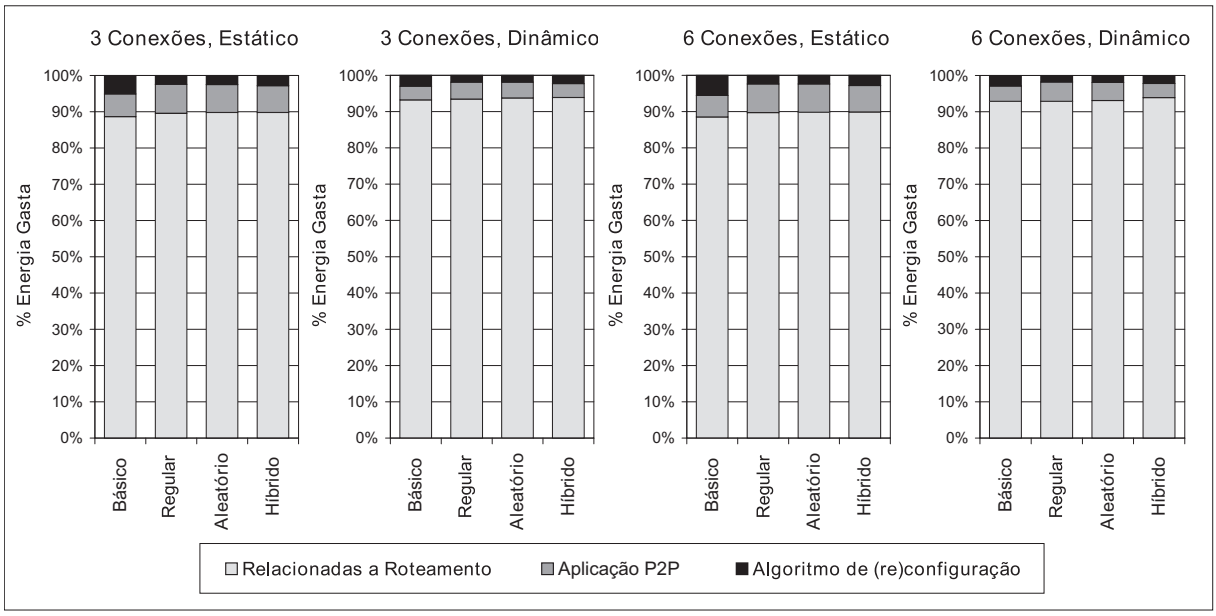

Figura 5.12: Composição Média do Gasto de Energia (AODV, com morte de nós).

Em ambos os ambientes —com ou sem morte de nós, 3 ou 6 conexões, estático ou dinâmico— a energia gasta com o recebimento e transmissão de mensagens relacionadas a roteamento representa mais de 85% do consumo total. Nos cenários dinâmicos, devido à maior efemeridade das conexões, são medidos os percentuais mais elevados, próximos de 93%.

Os gráficos da figura 5.12 nos mostram que, em cada cenário, a participação do gasto com as mensagens relacionadas a roteamento é similar para todos os algoritmos. Esta semelhança pode ser considerada natural, uma vez que o roteamento independe do algoritmo de reconfiguração. Nos cenários estáticos, os percentuais ficam próximos a 89%, enquanto que, nos dinâmicos, a cifra sobe para a casa dos 93%. Analisando os percentuais discriminados na tabela 5.4, podemos comparar melhor a participação dos gastos com as mensagens de (re)configuração, como visto a seguir.

As simulações com morte de nós representam o cenário que exibe os maiores ganhos dos algoritmos otimizados com relação ao Básico (considerando-se a energia relativa gasta com as mensagens de configuração). Quando o número máximo de conexões é igual a 6 e não há movimentação dos nós, o percentual de gasto do algoritmo Básico chega a ser 2,3 vezes maior que o do algoritmo com menor consumo, o Aleatório. A menor diferença ocorre também no cenário com 6 conexões, porém dinâmico: o Básico consome 60% mais que o Regular, que se mostra como o mais econômico.

Em todos os cenários, os percentuais dos algoritmos Regular e Aleatório se mostraram bastante semelhantes, e ambos sempre foram inferiores ao do Híbrido.

Como mostrado em todos os gráficos de energia média —figuras 5.2 a 5.9, o consumo de energia do algoritmo Básico é superior ao dos demais algoritmos. Desta forma, podemos afirmar que, se ele é o que proporcionalmente gasta mais com as mensagens de (re)configuração, o seu gasto de energia *absoluto* com tais mensagens também é superior ao dos demais.

Os gráficos da figura 5.13, que se referem ao cenário sem morte de nós, se assemelham aos do cenário com morte de nós. Novamente, a participação do gasto com as mensagens relacionadas a roteamento é similar para todos os algoritmos. Uma diferença é que, nos cenários estáticos, os percentuais ficam próximos a 91%, um pouco a mais que nas simulações com morte de nós. Nos cenários dinâmicos, a cifra também sobe para a casa dos 93%.

A tabela 5.5 discrimina os percentuais da composição nos cenários sem morte de nós. Ao compararmos a participação dos gastos com as mensagens de (re)configuração, constatamos

Cenário	Tipo de Gasto	Algoritmos de (Re)Configuração			
		Básico	**Regular**	**Aleatório**	**Híbrido**
3 Con., Estático	Alg. (Re)Configuração	2,99%	1,70%	1,61%	1,71%
	Aplicação P2P	5,93%	7,66%	7,20%	6,95%
	Rel. a Roteamento	91,08%	90,63%	91,20%	91,35%
3 Con., Dinâmico	Alg. (Re)Configuração	2,49%	1,79%	1,78%	1,97%
	Aplicação P2P	3,60%	4,66%	4,32%	3,72%
	Rel. a Roteamento	93,91%	93,55%	93,90%	94,31%
6 Con., Estático	Alg. (Re)Configuração	2,29%	1,18%	1,09%	1,48%
	Aplicação P2P	6,05%	7,49%	7,02%	6,61%
	Rel. a Roteamento	91,66%	91,33%	91,90%	91,91%
6 Con., Dinâmico	Alg. (Re)Configuração	1,92%	1,34%	1,38%	1,75%
	Aplicação P2P	4,35%	5,44%	5,09%	3,87%
	Rel. a Roteamento	93,74%	93,22%	93,53%	94,38%

Tabela 5.5: Composição da energia gasta (AODV, sem morte de nós).

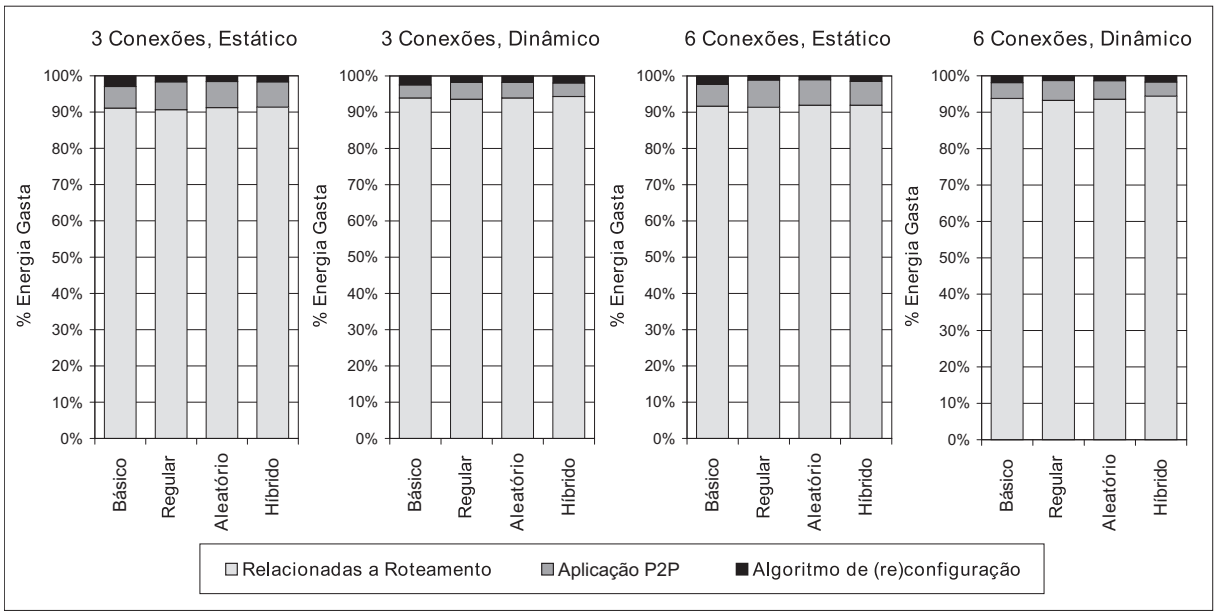

Figura 5.13: Composição Média do Gasto de Energia (AODV, sem morte de nós).

que o algoritmo Básico continua exibindo o maior consumo relativo (e absoluto, como explicado anteriormente). A diferença com relação aos demais algoritmos, entretanto,

diminuiu. Nos testes com movimentação de nós e número máximo de conexões igual a 6, tem-se a menor diferença: o percentual do Básico é 43% superior ao do algoritmo com menor consumo, o Regular. Possivelmente isto ocorre pois no algoritmo Básico não há controle da distância entre os *peers*, o que possibilita a maior permanência das conexões mesmo com os nós se movendo. Parte de tais conexões, de acordo com os outros algoritmos, seriam desfeitas, demandando nova troca de mensagens para estabelecimento de novas conexões. De qualquer forma, apesar da economia de mensagens de (re)configuração, potencialmente há um gasto maior com mensagens de roteamento já que há o aumento da distância entre os nós na rede ad hoc.

Por outro lado, a maior margem entre o Básico e os algoritmos otimizados ocorre no outro cenário de 6 conexões, o estático, no qual a contribuição do gasto com mensagens de (re)configuração do algoritmo Básico é 2,1 vezes maior que o do Aleatório.

Assim como nas simulações com morte de nós, em todos os cenários, os percentuais dos algoritmos Regular e Aleatório se mostraram bastante semelhantes, e ambos sempre foram inferiores ao do Híbrido. Novamente tem-se a similaridade entre os algoritmos Aleatório e Regular. Ou seja, não se ter obteve o ganho esperado do efeito *small-world*, mas, pelo menos, também não se teve perdas conseqüentes do estabelecimento da conexão aleatória. Com relação ao ganho em comparação ao algoritmo Híbrido, torna-se difícil estimá-lo pois, em termos de consumo total de energia, o Híbrido sempre se mostrou melhor.

Conclusões

A análise dos resultados alcançados nas simulações que utilizaram o AODV como protocolo de roteamento baseou-se em três aspectos básicos: energia média da rede ao longo do tempo, porcentagem de sucesso (*hits*) das consultas e composição média do gasto de energia.

O algoritmo Híbrido apresentou, em todos os cenários simulados, a maior energia média da rede ao longo do tempo, ou seja, ele foi o que menos consumiu energia. Estima-se que a organização da rede em mestres e escravos propiciou tal desempenho pois os escravos, por se comunicarem somente com seus mestres, têm uma rotina que demanda muito menos energia que os nós dos demais algoritmos. Os mestres, por sua vez, podem consumir mais que os nós dos outros algoritmos. Entretanto, a rede híbrida potencialmente pode ser formada por bem mais escravos que mestres, de forma que o consumo total de energia pela rede seja menor.

O Básico, por sua vez, exibiu o maior consumo em todos os casos. Este comportamento

realmente era esperado, pois as melhorias dos algoritmos otimizados visavam especialmente à economia do consumo de energia. Tal fato, então, confirma a eficiência das otimizações propostas.

Por fim, tem-se os algoritmos Regular e Aleatório. O primeiro apresentou consumo menor que o do segundo em todas as simulações, mas suas curvas sempre estiveram muito próximas, especialmente nos cenários com morte de nós. A similaridade do comportamento destes algoritmos e especialmente o fato do Regular ser melhor demonstram que o efeito *small-world* não se manifestou, ou não se traduziu em economia de energia. Entretanto, ao mesmo tempo em que a conexão aleatória não trouxe o ganho esperado, ela também não gerou sobrecarga acentuada –mas talvez tenha sido a responsável pelo desempenho um pouco pior do Aleatório com relação ao Regular.

A não constatação do efeito *small-world* pode ter várias explicações. Uma delas é que o número de conexões aleatórias pode ter sido proporcionalmente grande com relação às conexões regulares. Assim, a conformação do grafo ficaria mais próxima à de um grafo aleatório que de um grafo *small-world*. Outra possível explicação seria que todas as conexões foram efetivamente estabelecidas em um raio de passos similar, não havendo a diferenciação entre conexões regulares e aleatórias. Por fim, pode-se pensar que o cenário simulado, especialmente com relação à área abrangida e número de nós, propiciou a formação de redes com nós bem interconectados ou com grupos totalmente isolados na rede ad hoc. Desta maneira, as conexões aleatórias não foram capazes de agir como as desejadas pontes (atalhos), ou se mostraram como pontes desnecessárias entre nós já unidos.

Sob o aspecto do sucesso das consultas, a o algoritmo Híbrido já não é mais o melhor em todos os cenários. Considerando-se os 8 cenários (4 com morte e 4 sem morte de nós), o melhor desempenho é atribuído da seguinte forma: 3 vezes para o Híbrido, 3 vezes para o Regular e 2 vezes para o Básico. Apesar de nunca apresentar o maior percentual de sucesso, o algoritmo Aleatório sempre se mostra perto dele.

O pior desempenho nunca é atribuído ao algoritmo Regular ou Aleatório, sendo dividido igualmente entre Híbrido e o Básico: cada um apresenta o índice mais baixo em 4 cenários.

O comportamento simples do algoritmo Básico, com a assimetria de suas conexões e não controle da distância dos *peers*, apesar de gerar maior consumo de energia, poderia significar importante vantagem durante as consultas P2P por potencialmente ampliar a cobertura das consultas. Este benefício, entretanto, não foi relevante. Isto permite considerar as melhorias dos algoritmos otimizados com pouco –ou nenhum– efeito colateral. Ou seja, interpretando os ganhos no consumo de energia como *benefício* e o eventual impacto das

otimizações no sucesso das consultas como *custo*, podemos concluir que os algoritmos otimizados apresentam uma boa relação custo benefício. A seguir são apresentadas algumas cifras que detalham melhor o ganho obtido.

Nos cenários com morte de nós, o algoritmo Híbrido mostra-se como uma excelente opção: em dois casos exibe os maiores percentuais e, nos outros dois, seu índice é 91,68% e 93,68% dos maiores percentuais, que pertencem a outro algoritmo otimizado, o Regular.

Nos cenários sem morte de nós, o algoritmo Regular se destaca. Apesar de apresentar o melhor desempenho em apenas 1 dos 4 cenários, o algoritmo Regular fica bem próximo do melhor colocado nas outras 3 situações. Nos 2 cenários em que o Básico exibe o maior percentual de sucesso, o índice do Regular atinge 96,16% e 98,05% de tal percentual. No quarto cenário, o melhor colocado é outro algoritmo otimizado, o Híbrido, e o percentual do Regular atinge 95,81% do melhor índice.

De uma forma geral, os algoritmos Regular e Aleatório apresentam-se com comportamento mais homogêneo tanto com relação ao consumo de energia e sucesso das consultas. Este sucesso pode ser atribuído ao fato que eles têm muito da simplicidade do Básico, a qual é enriquecida com otimizações também presentes no Híbrido, ao passo que não necessitam de uma conformação mais específica da rede –como é o caso da diferenciação dos nós das redes Híbridas. É importante ressaltar que, mesmo quando não têm o melhor desempenho, estão proporcionalmente perto de atingi-lo.

O último aspecto observado, a composição média do gasto de energia, confirma o grande ganho atingido pelos algoritmos otimizados com relação às mensagens de (re)configuração. Independentemente do cenário, no entanto, constatou-se que a energia gasta com tais mensagens tem uma pequena participação na energia total consumida pelos algoritmos, a qual é dominada pelo consumo gerado por mensagens relacionadas a roteamento. Diante disso, vislumbrou-se a possibilidade de maximizar os benefícios das otimizações propostas, tomando-se como ponto de partida uma análise mais detalhada da influência do protocolo de roteamento utilizado. Desta forma, deu-se início a um segundo conjunto de testes, nos quais utilizou-se o algoritmo de roteamento DSR. Por serem mais dinâmicos e talvez representarem uma realidade mais comum, foram escolhidos os cenários com morte de nós para as novas simulações.

5.3.2 Resultados DSR

A observação da prevalência do gasto com mensagens relacionadas a roteamento no consumo total de energia, motivou-nos a observar o comportamento dos algoritmos propostos

sobre outro protocolo de roteamento. Nesta seção, exibimos os resultados das simulações com morte de nós, utilizando o DSR. A análise dos dados coletados será direcionada à comparação com aqueles relativos aos testes com o AODV.

Novamente iniciaremos nossa análise pela energia média da rede ao longo da simulação.

O cenário estático, com número máximo de conexões igual a 3, tem seus resultados mostrados na figura 5.14. As curvas se assemelham às do AODV, porém o consumo de todos os algoritmos foi menor com o uso do DSR: o Híbrido gastou 15% a menos, o Regular, 16%, o Aleatório, 17% e o Básico, 8%. Uma vez que a economia dos algoritmos otimizados no novo cenário não foi acompanhada pelo Básico, sua curva distanciou-se ainda mais das demais. Outra alteração observada em comparação com as simulações que utilizaram o AODV é que o comportamento dos algoritmos Regular e Aleatório ficaram ainda mais similares (a energia final difere em menos de 1%).

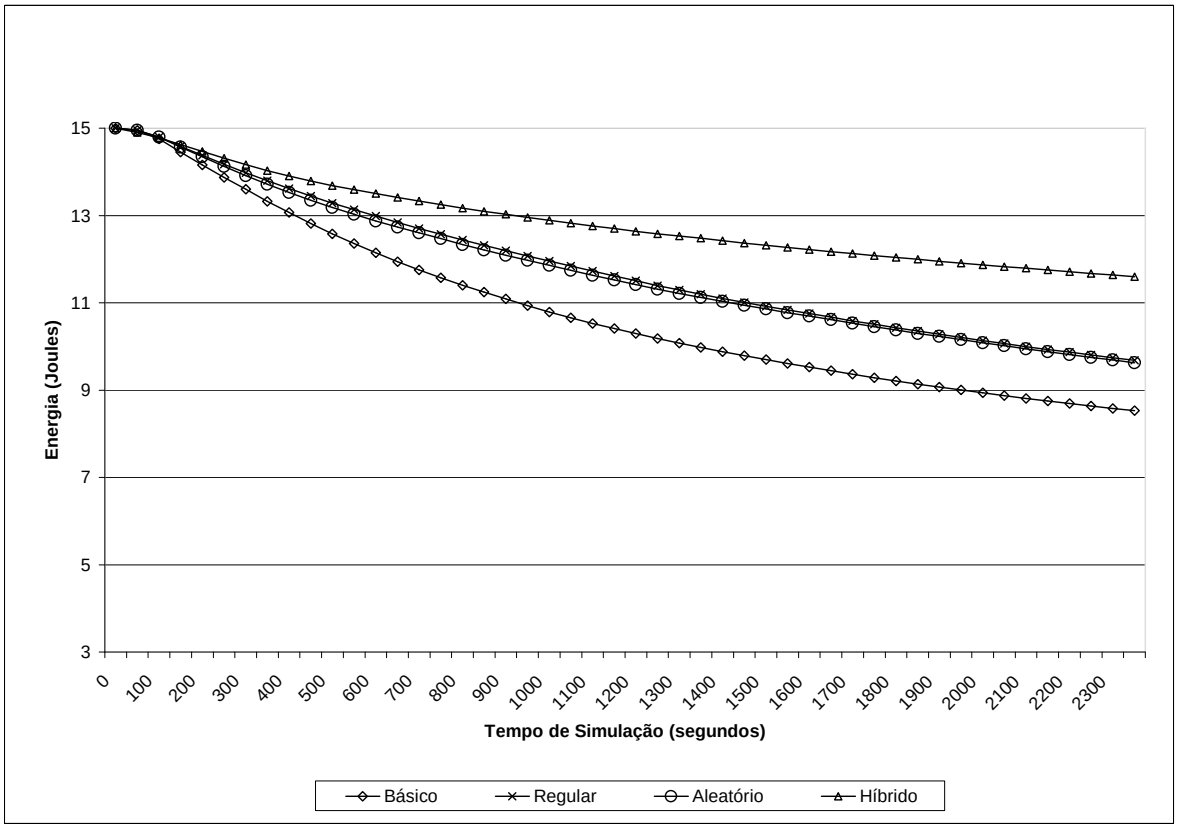

Figura 5.14: Energia média da rede ao longo da simulação (3 conexões, estático, DSR, com morte de nós.

Ao aumentarmos o número máximo de conexões para 6, mantendo o cenário estático,

percebe-se novamente que o DSR proporcionou economia no consumo de energia: o Híbrido gastou 15% a menos, o Regular, 10%, o Aleatório, 10% e o Básico, 6%. O comportamento relativo dos algoritmos, entretanto, não mudou: as curvas somente se deslocaram para patamares superiores de energia.

Neste cenário estático, as rotas já descobertas somente tornam-se inválidas com a morte de algum dos nós que a compõem. Além disso, ambos os protocolos –AODV e DSR– são reativos, ou seja, a atualização das rotas ocorre apenas no momento em que elas são demandadas. Desta forma, ele é menos crítico para os algoritmos de roteamento estudados. A diferença de consumo detectada pode ser um indicativo de que o DSR responde melhor à perda de nós.

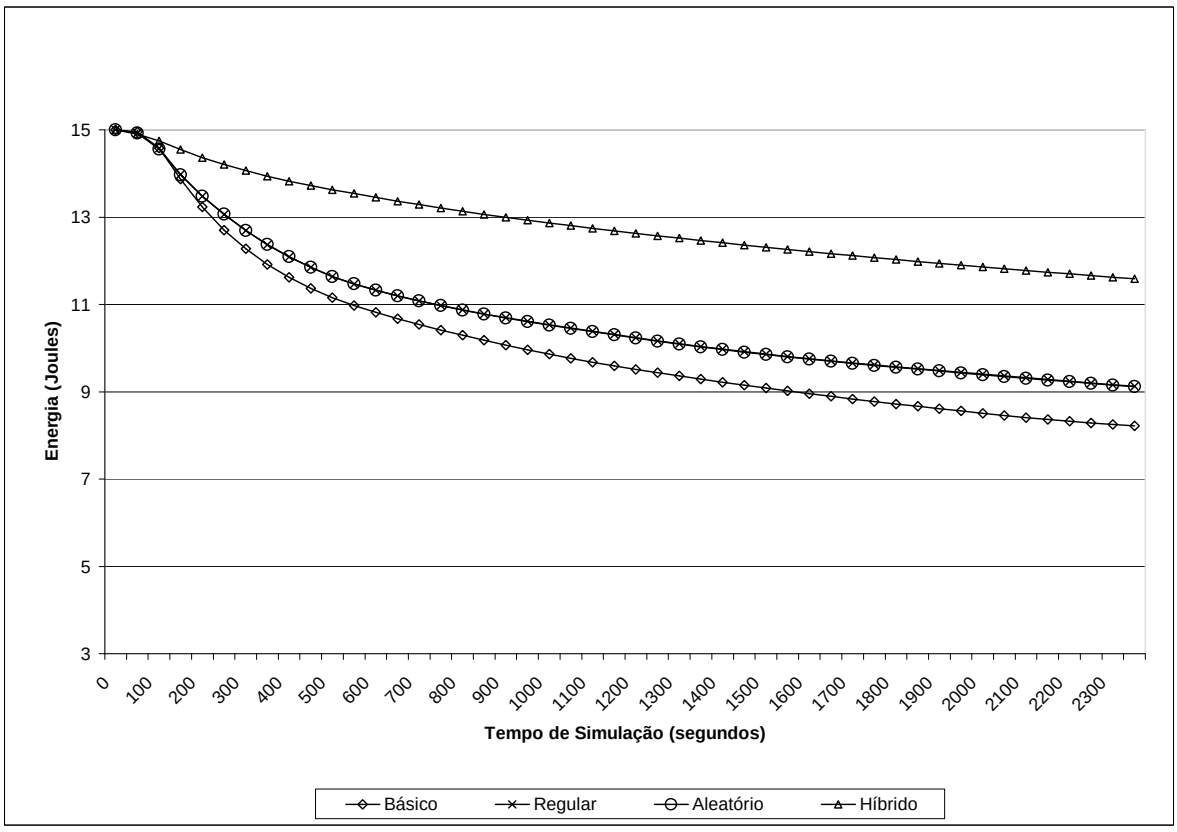

Figura 5.15: Energia média da rede ao longo da simulação (6 conexões, estático, DSR, com morte de nós.

As figuras 5.16 e 5.17 referem-se aos cenários dinâmicos, que, ao contrário dos estáticos, exibem um maior consumo com a utilização do DSR como protocolo de roteamento. Outra diferença com relação aos cenários estáticos são as curvas de energia. Com a ausên-

cia de movimentação, há uma caracterização maior da fase de formação da rede, que apresenta uma taxa de consumo maior, seguida pela fase de manutenção, com curvas mais linearizadas. Após a inserção de mobilidade aumenta a efemeridade das conexões, e a rede mostra-se em processo mais contínuo de formação, com taxas altas durante um intervalo de tempo mais prolongado.

Quando o número máximo de conexões é 3 (figura 5.16), o algoritmo Híbrido aumenta seu consumo em 26%, o Regular, 14%, o Aleatório, 11% e o Básico, apenas 2%. Com estas alterações, o comportamento dos algoritmos Regular e Aleatório praticamente se sobrepõem, além de se aproximarem do Básico. Outra alteração é que as curvas dos algoritmos otimizados, que exibiam um aspecto mais linear, passam a ilustrar um consumo de energia mais acentuado na metade inicial do tempo de simulação.

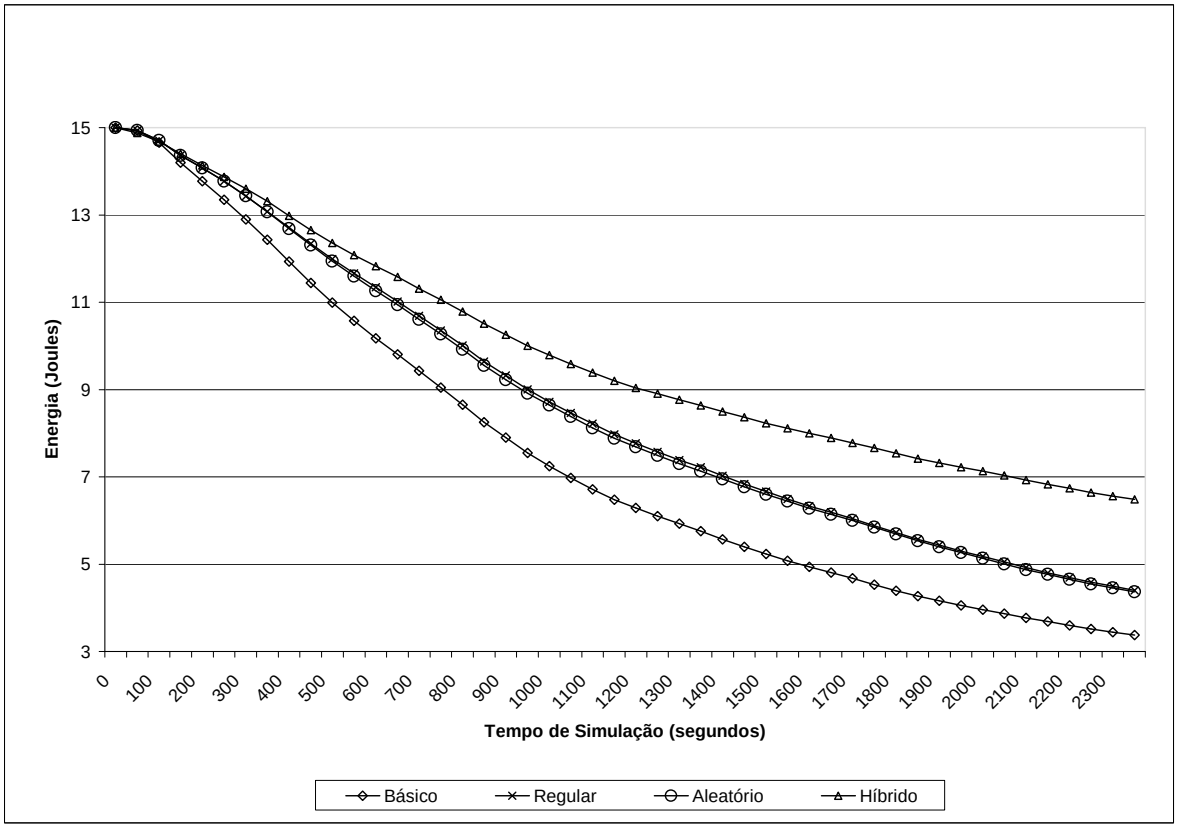

Figura 5.16: Energia média da rede ao longo da simulação (3 conexões, dinâmico, DSR, com morte de nós.

No último cenário (dinâmico, 6 conexões), os resultados retratados pelo gráfico 5.17 confirmam o pior desempenho do DSR quando os nós se movimentam. O aumento de

consumo de energia foi de 25%, 5%, 4% e 1% para os algoritmos Híbrido, Regular, Aleatório e Básico, respectivamente. Novamente, as curvas do Regular e do Aleatório aproximam entre si, e em relação ao Básico. Assim como no cenário com 3 conexões, também se percebe um aumento na taxa de consumo de energia na primeira metade do intervalo de tempo. Apesar de, com relação ao cenário equivalente que usa o AODV, o desempenho do algoritmo Híbrido ter se aproximado daquele dos outros algoritmos, ele ainda se destaca positivamente, consumindo 24% menos que o segundo colocado.

O cenário dinâmico é mais crítico para os protocolos de roteamento. Apesar de ambos o DSR e o AODV serem reativos, eles têm uma diferenciação importante, que potencialmente justifica a diferença de consumo de energia: a utilização de *source routing*. Como visto na seção 2.2, a utilização de *source routing* proporciona economia de memória aos nós, pois não têm que armazenar o caminho em tabelas internas. Em contrapartida, entretanto, o tamanho de cada pacote aumenta devido a conter a informação completa sobre a rota. Já no AODV, os pacotes são menores e o uso de memória, maior. Apesar da limitação de memória ser relevante nos dispositivos móveis, a transmissão de pacotes maiores representa impacto maior no consumo de energia. Sob este aspecto, então, o AODV mostra-se mais eficiente.

Uma vez observado o consumo de energia, passamos ao estudo do possível impacto causado pelos algoritmos de (re)configuração sobre a aplicação P2P através da análise do percentual de sucesso das consultas (tabela 5.6 e figura 5.18).

A figura 5.18 assemelha-se bastante à sua equivalente das simulações com o AODV: nos cenários estáticos, o Híbrido exibe os maiores percentuais de sucesso, com uma diferença significativa com relação aos demais; nos cenários dinâmicos, todos os algoritmos apresentam índices de sucesso bastante semelhantes, com uma pequena superioridade do Regular. A diferença mais significativa entre os testes com o AODV e o DSR, com relação aos *hits*, é que, no último caso, o Básico se aproximou do desempenho do Regular e do Aleatório, chegando a ultrapassá-los no cenário estático com número máximo de conexões igual a 3. A alteração, entretanto, não chega a caracterizar uma mudança significativa de comportamento, pois as diferenças percentuais –tanto antes, quanto agora– são muito pequenas. A tabela 5.6 discrimina as porcentagens de sucesso, permitindo uma melhor diferenciação entre os algoritmos (especialmente nos cenários dinâmicos).

De uma forma geral, percebe-se que o impacto da troca do protocolo de roteamento foi praticamente imperceptível, ou seja, é possível reafirmar que o ganho obtido pelos algoritmos otimizados no consumo de energia não teve efeito colateral sobre o sucesso das

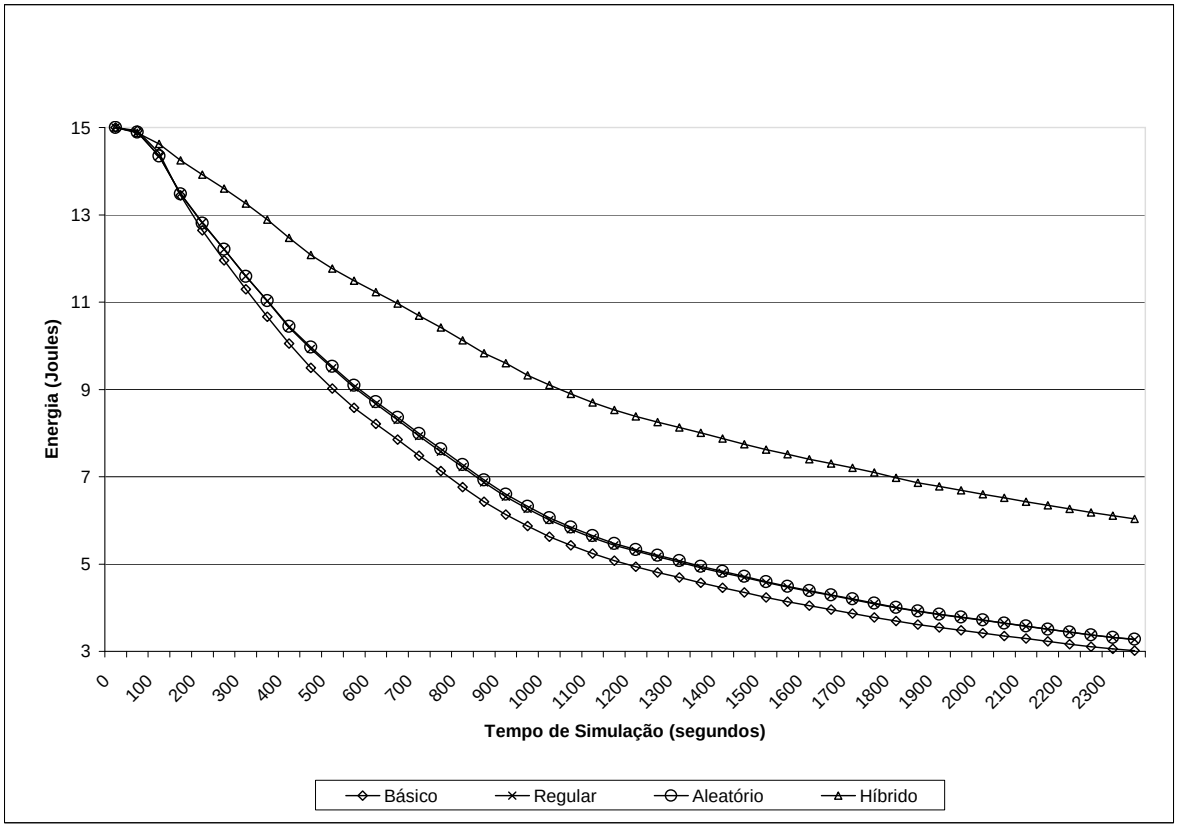

Figura 5.17: Energia média da rede ao longo da simulação (6 conexões, dinâmico, DSR, com morte de nós.

Cenário	Algoritmos de (Re)Configuração			
	Básico	**Regular**	**Aleatório**	**Híbrido**
3 Conexões, Estático	31,38%	30,99%	31,17%	35,32%
3 Conexões, Dinâmico	29,94%	31,18%	30,71%	30,18%
6 Conexões, Estático	28,95%	30,50%	30,42%	35,06%
6 Conexões, Dinâmico	29,38%	30,38%	30,19%	29,86%

Tabela 5.6: Percentuais de Sucesso (*hits*) das Consultas (DSR, com morte de nós).

consultas.

A realização de simulações variando-se o protocolo de roteamento foi motivada pelo gasto proporcionalmente grande de energia com mensagens relacionadas a roteamento. Neste momento, através da figura 5.19, teremos oportunidade de verificar se este fato, constatado com o AODV, se repete com o uso do DSR.

Novamente, observando-se a composição do consumo total, há a indiscutível prevalência do gasto de energia com mensagens relacionadas a roteamento. A participação relativa de cada tipo de mensagem, entretanto, sofreu alterações. Especialmente nos cenários estáticos, enquanto que com o AODV os percentuais do gasto relacionado a roteamento ficavam na casa dos 89%, agora, com o uso do DSR, eles passaram para aproximadamente 86%. Essa pequena diferença, na verdade, significa que a participação do gasto com mensagens de (re)configuração cresceu, em alguns casos, até 44% nos cenários estáticos com DSR.

Quanto maior a participação do gasto com mensagens de (re)configuração, maior o impacto das otimizações propostas para os algoritmos no consumo total de energia. Isto pôde ser observado na análise do consumo de energia dos cenários estáticos, quando os algoritmos otimizados apresentaram ganhos muito mais significativos que o Básico.

Todavia, nos cenários dinâmicos, a modificação foi bem mais sutil. Com a ajuda da tabela 5.7, percebemos que houve pequeno acréscimo na participação do gasto com mensagens de (re)configuração no caso dos algoritmos Básico, Regular e Aleatório. Com o Híbrido, entretanto, ocorreu o oposto: os percentuais sofreram diminuição com relação ao mesmo cenário do AODV. A diminuição, entretanto, foi pequena: por volta de 5%. Ao mesmo tempo em que o aumento constatado no cenário estático foi acima de 30%. Assim, podemos concluir que a influência mais significativa do DSR acontece no cenário estático, sob o aspecto de mensagens de (re)configuração. Esta observação, somada à economia de energia proporcionada pelo DSR nos cenários estáticos, demonstra a maior adequação deste protocolo às situações sem mobilidade.

Independentemente do cenário, a contribuição do consumo do algoritmo Básico com mensagens de (re)configuração é nitidamente superior ao dos algoritmos otimizados. Uma vez que o consumo absoluto do Básico é sempre superior ao dos demais, essa maior contribuição significa que o seu gasto absoluto com os algoritmos de (re)configuração também é maior que o dos outros –como já havia sido observado com o AODV. Assim, mais uma vez confirmamos que nossos algoritmos alcançaram seus objetivos com relação à preservação deste escasso recurso que é a energia.

Conclusões

A análise dos resultados alcançados nas simulações que utilizaram o DSR como protocolo de roteamento baseou-se em três aspectos básicos: energia média da rede ao longo do tempo, porcentagem de sucesso (*hits*) das consultas e composição média do gasto de energia. Deu-se enfoque às alterações de comportamento com relação aos testes que utilizaram o AODV.

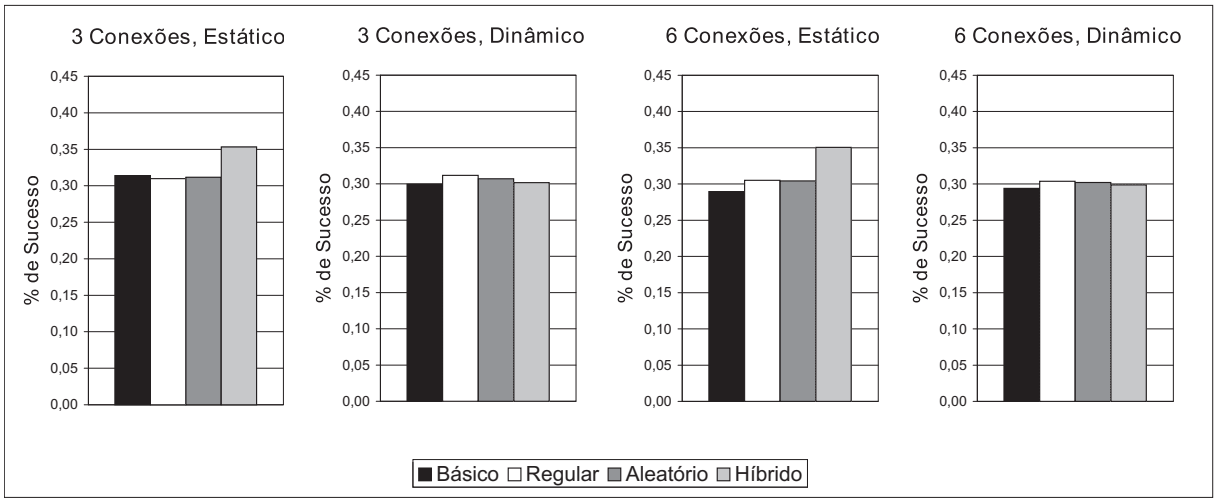

Figura 5.18: Porcentagem de Sucesso (*hits*) das consultas (DSR, com morte de nós).

Cenário	**Tipo de Gasto**	**Algoritmos de (Re)Configuração**			
		Básico	**Regular**	**Aleatório**	**Híbrido**
3 Con., Estático	Alg. (Re)Configuração	6,35%	3,55%	3,57%	3,95%
	Aplicação P2P	8,08%	9,90%	9,75%	10,11%
	Rel. a Roteamento	85,56%	86,54%	86,67%	85,94%
3 Con., Dinâmico	Alg. (Re)Configuração	3,24%	1,93%	2,00%	2,23%
	Aplicação P2P	4,12%	4,58%	4,40%	3,89%
	Rel. a Roteamento	92,64%	93,49%	93,60%	93,87%
6 Con., Estático	Alg. (Re)Configuração	6,58%	3,17%	3,19%	3,94%
	Aplicação P2P	7,22%	9,95%	9,89%	9,98%
	Rel. a Roteamento	86,19%	86,88%	86,92%	86,08%
6 Con., Dinâmico	Alg. (Re)Configuração	3,01%	1,82%	1,94%	2,09%
	Aplicação P2P	4,64%	5,28%	5,11%	3,96%
	Rel. a Roteamento	92,35%	92,90%	92,96%	93,95%

Tabela 5.7: Composição da energia gasta (DSR, com morte de nós).

Nos cenários estáticos, o uso do DSR proporcionou uma diminuição significativa do consumo de energia por todos os algoritmos, que variou entre 8 e 17%. O contrário, entretanto, ocorreu nos cenários dinâmicos: todos aumentaram o gasto de energia. O Básico foi o que teve a menor alteração, enquanto que o Híbrido chegou a apresentar acréscimo de 26% no seu consumo. Estima-se que o aumento do consumo ocorreu por

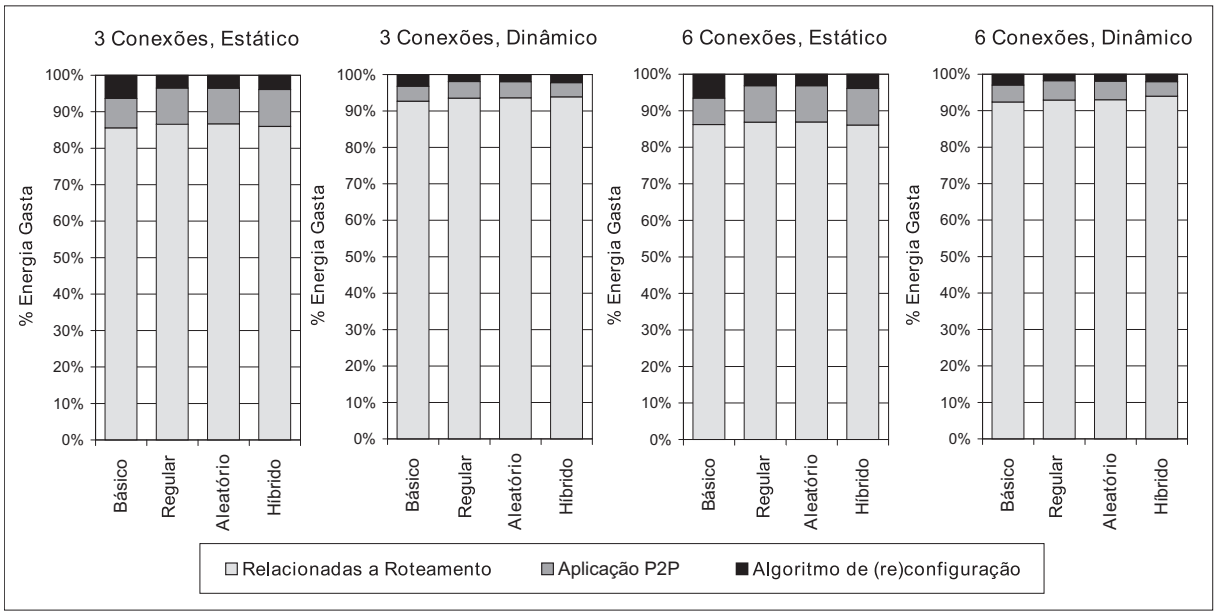

Figura 5.19: Composição Média do Gasto de Energia (DSR, com morte de nós).

causa do *source routing*, que representa sobrecarga na transmissão dos pacotes.

Apesar das variações no consumo, novamente o algoritmo Híbrido apresentou a maior energia média da rede ao longo do tempo, e o Básico, a menor. Os algoritmos Regular e Aleatório passaram a exibir curvas ainda mais próximas, praticamente sobrepostas, e, nos cenários dinâmicos, se aproximaram do comportamento do algoritmo Básico.

A aproximação das curvas do Aleatório e do Regular pode significar que o gasto de energia com as conexões mais longas (maior número de *hops*), como a aleatória, torna-se menor com a utilização do DSR. Isto também explicaria a maior semelhança com o Básico, já que ele, por não controlar a distância entre os *peers*, potencialmente também possui conexões longas.

De qualquer forma, a alteração do protocolo de roteamento não impactou a (não) manifestação do efeito *small-world*, mantendo-se as suposições feitas nas 5.3.1 da seção 5.3.1.

Sob o aspecto do sucesso das consultas, os percentuais atingidos e o comportamento relativo entre os algoritmos não sofreram alterações significativas com relação às simulações que utilizaram o AODV. O Híbrido continuou sendo o melhor nos cenários estáticos, e o Regular, o melhor nos dinâmicos (com pequena diferença com relação aos demais). Assim, como a economia de energia do Híbrido é maior e o seu índice de sucesso é praticamente o mesmo do Regular, ele se consolida como a melhor opção nos cenários com morte de nós.

Interpretando os ganhos no consumo de energia como *benefício* e o eventual impacto das otimizações no sucesso das consultas como *custo*, mantemos a conclusão da seção 5.3.1 de que os algoritmos otimizados apresentam uma boa relação custo benefício.

O último aspecto observado, a composição média do gasto de energia, confirma o grande ganho atingido pelos algoritmos otimizados com relação às mensagens de (re)configuração.

As simulações com o DSR confirmaram que, apesar da troca do protocolo, a energia gasta com mensagens relacionadas a roteamento prevalece sobre o gasto com as demais mensagens. Diante disso, caso se deseje explorar ainda mais o potencial das otimizações propostas, torna-se interessante o aprofundamento na questão do roteamento.

Capítulo 6

Conclusões e Trabalhos Futuros

A Computação Móvel tem como um de seus objetivos permitir às pessoas acessarem os mais distintos tipos de informação *a qualquer momento* e *em qualquer lugar*. No geral, a arquitetura cliente/servidor não é adequada para satisfazer a essa demanda devido a vários motivos: o servidor pode estar indisponível (pouca tolerância a falhas), sobrecarregado (problemas de escalabilidade), ou pode não haver nenhuma infra-estrutura de acesso ao servidor ou a outras entidades.

Neste contexto, a combinação de redes P2P sobre redes ad hoc mostra-se como uma solução interessante. No paradigma P2P, os *peers* podem atuar tanto como clientes quanto como servidores, o que aumenta a tolerância a falhas e a disponibilidade dos dados. Nas redes ad hoc, por sua vez, cada nó é capaz de estabelecer uma comunicação ponto-a-ponto com outros nós que estejam dentro da sua área de cobertura, sem a necessidade de infra-estrutura fixa. O uso de redes P2P sobre redes ad hoc, entretanto, leva a um cenário altamente dinâmico, no qual as referências entre os nós mudam constantemente, demandando uma freqüente reconfiguração. Isto, por sua vez, pode causar um grande impacto nos recursos escassos da rede, tais como energia dos dispositivos e largura de banda.

6.1 Conclusões

Neste trabalho, a fim de promover um compartilhamento eficiente de informação e de aprimorar a longevidade da rede, foram propostos quatro algoritmos para a configuração, manutenção e reorganização da redes P2P sobre redes ad hoc. Estes algoritmos não dependem de entidades externas, ou seja, eles permitem que a rede seja plenamente autônoma

em sua organização.

O primeiro algoritmo apresentado, o *Básico*, destina-se a representar um algoritmo de (re)configuração simples e, portanto, a servir como uma base para comparação e avaliação dos demais algoritmos propostos. Ele foi inspirado no reconhecido protocolo *Gnutella*. Sua principal característica — simplicidade — proporciona uma fácil implementação e manutenção, mas ignora parcialmente a natureza dinâmica da rede.

Os demais algoritmos foram propostos no intuito de otimizar a (re)configuração da rede, tendo um enfoque especial de sanar alguns problemas detectados no algoritmo *Básico*. Dois deles, o *Regular* e o *Aleatório*, também são descentralizados, possuindo diversas semelhanças com o *Básico*. Por fim, o algoritmo *Híbrido*, como o próprio nome indica, segue um paradigma um pouco diferenciado, o da topologia híbrida.

Os quatro algoritmos foram implementados na linguagem C++ e testados através de simulação utilizando-se o *Network Simulator ns-2* [20], que provê grande suporte à construção de modelos de redes ad hoc. Os algoritmos foram submetidos a diversos cenários, nos quais foram variados parâmetros como energia inicial, movimentação e número máximo de conexões entre os nós.

Os resultados obtidos mostram que os algoritmos *Regular* e o *Aleatório* apresentam praticamente o mesmo comportamento, apesar das otimizações inseridas nesse último, na tentativa de se obter o efeito *Small World*. De uma forma geral, eles mostraram um bom desempenho em todos os cenários, demonstrando um consumo moderado de energia sem perda para a aplicação P2P.

O algoritmo *Híbrido*, no geral, apresentou um desempenho superior aos demais, especialmente no tocante ao gasto de energia. Sua utilização, entretanto, demanda um maior conhecimento da rede, pois ele parte do princípio que os nós constituem um grupo heterogêneo.

Finalmente, mostrou-se que o algoritmo *Básico*, baseado em uma solução tradicional para redes fixas, apresentou o pior desempenho geral, exibindo um custo alto (consumo elevado de enrgia) sem compensação no sucesso das consultas.

6.2 Trabalhos Futuros

O tema abordado é muito rico e apresenta diversos desafios, de forma que temos vários possíveis caminhos a seguir no futuro.

Uma primeira linha de investigação seria a variação de alguns parâmetros atuais, como

número de elementos, densidade de nós na área, porcentagem de elementos da rede ad hoc na rede P2P, área de cobertura do rádio. Algumas destas variáveis já foram testadas em fases preliminares deste trabalho, quando ainda não se observava o consumo de energia. Futuramente, então, pode-se eleger um dos cenários aqui apresentados como pano de fundo para a experimentação destas variáveis.

A semelhança entre os comportamentos dos algoritmos *Básico* e *Aleatório* nos instiga a aprofundar, também, na pesquisa do efeito *Small World*. Neste sentido, acreditamos que seriam interessantes simulações com um grande número de elementos, de forma que o número de conexões seja pequeno diante da quantidade de nós. Talvez assim, as conexões aleatórias passem realmente a atuar como pontes.

Diante da representatividade da energia gasta com mensagens relacionadas a roteamento, um outro caminho muito interessante a ser seguido seria a busca de uma maior sinergia entre os algoritmos propostos e os protocolos de roteamento. Este caminho demandaria um estudo bem mais detalhado e profundo das soluções de roteamento adotadas nas redes ad hoc.

Por fim, tem-se aquele que talvez seja o maior desafio de todos: desenvolver um modelo teórico que possibilite, entre outros, a análise da complexidade dos algoritmos propostos. Já foram feitas alguns estudos preliminares neste sentido, mas que precisam ser aprofundados.

Referências Bibliográficas

[1] F. Franciscani, M. Vasconcelos, R. Couto, and A. A. Loureiro. Peer-to-peer over ad-hoc networks: (re)conguration algorithms. In *17th International Parallel and Distributed Processing Symposium - IPDPS 2003*, page 10pp, Nice, França, Abril 2003.

[2] F. Franciscani, M. Vasconcelos, R. Couto, and A. A. Loureiro. Peer-to-peer over ad-hoc networks: (re)conguration algorithms. *Journal of Parallel and Distributed Computing - Special Issue*.

[3] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella Network. *IEEE Internet Computing Journal*, 6(1), 2002.

[4] T. Hong. Performance. In Andy Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 14, pages 203–241. O'Reilly and Associates, Sebastopol, CA, EUA, Março 2001.

[5] N. Minar. Distributed Systems Topologies. In *The O'Reilly P2P and Web Services Conf*, 2001.

[6] SETI@Home. http://setiathome.ssl.berkeley.edu/.

[7] Freenet. http://freenetproject.org/.

[8] KazaA. http://www.kazaa.com/.

[9] R. Schollmeier and I. Gruber. Routing in Peer-to-peer and Mobile Ad Hoc Networks. A Comparison. In *International Workshop on Peer-to-Peer Computing*, Maio 2002.

[10] E. M. Royer and C. K. Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. *IEEE Personal Communications*, 2:46–55, Abril 1999.

[11] L. B. Oliveira, I. G. Siqueira, and A. A. Loureiro. Evaluation of Ad-hoc Routing Protocols under a Peer-to-Peer Application. In *IEEE Wireless Communication and Networking Conference*, 2003.

[12] C. Perkins, E. Royer, and S. Das. Ad-Hoc On-Demand Distance Vector (AODV) Routing. IETF Internet draft, draft-ietf-manet-aodv-11.txt, Junho 2002.

[13] D. Johnson, D. Maltz, and J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[14] S. Milgram. The Small-World Problem. *Psychology Today*, 1(1):60–67, 1967.

[15] D. Watts and S. Strogatz. Collective Dynamics of 'Small-World' Netwoks. *Nature*, 393:440–442, 1998.

[16] G. Ding and B. Bhargava. Peer-to-peer file-sharing over mobile ad hoc networks. In *Second IEEE Annual Conference on Pervasive Computing and Communications*, pages 104–108, Orlando, EUA, Março 2004.

[17] M. Papadopouli and H. Schulzrinne. A performance analysis of 7DS: A peer-to-peer data dissemination and prefetching tool for mobile users. In *2001 IEEE Sarnoff Symposium: Advances in Wired and Wireless Communications*, Ewing, EUA, Março 2001.

[18] M. Papadopouli and H. Schulzrinne. Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination among Mobile Devices. In *ACM SIGMOBILE Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc) 2001*, pages 117–127, Outubro 2001.

[19] G. Kortuem, J. Schneider, D. Preuitt, T.G.C. Thompson, S. Fickas, and Z. Segall. When peer-to-peer comes face-to-face: Collaborative peer-to-peer computing in mobile ad-hoc networks. In *First International Conference on Peer-to-Peer Computing*, pages 75–91, Linköpings, Suécia, Agosto 2001.

[20] NS-2. http://www.isi.edu/nsnam/ns.

[21] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.

[22] G. K. Zipf. *Human Behavior and the Principle of Least-Effort.* Addison-Wesley, 1949.