ARQUITETURA ROBUSTA E ESCALÁVEL PARA DESCOBERTA DE SERVIÇOS EM REDES AD HOC

Carlos Henrique Pereira Augusto

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

Aprovada por:

Prof. José Ferreira de Rezende, Dr.

Prof. Aloysio de Castro Pinto Pedroza, Dr.

Prof. Célio Vinicius Neves de Albuquerque, PhD

RIO DE JANEIRO, RJ - BRASIL

MAIO DE 2007

AUGUSTO, CARLOS HENRIQUE PEREIRA

Arquitetura Robusta e Escalável para Descoberta de Serviços em Redes Ad Hoc [Rio de Janeiro] 2007

XIII, 97 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2007)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Descoberta de Serviços

2. Redes Ad Hoc

3. Escalabilidade

I. COPPE/UFRJ II. Titulo (série)

Aos meus filhos, Lucas e Marcos, e ao meu saudoso irmão Sergio.

Agradecimentos

À Lili, minha esposa, companheira e amiga, e aos meus filhos, Lucas e Marcos, pela compreensão pelo tempo subtraído deles.

Ao Prof. José Ferreira de Rezende, pela amizade e realismo com que me orientou, sabendo compreender minhas dificuldades e ajudando-me a superá-las.

Aos amigos Marcel, Kleber, Fabiana, Rodrigo, Laila e Yuri, do GTA, pelo apoio e cumplicidade durante a realização do mestrado.

Aos amigos Luiz Antônio, José Roberto, Paulo Gaya e Gabriela, do Bacen, pelo incentivo, e ao grande amigo Rafael, também pela compreensão pelo meu afastamento.

Aos meus pais, Gerson e Ednéa, por toda dedicação e esforço ao longo da vida, que me permitiram encontrar meu caminho, e alcançar mais este objetivo.

Ao Banco Central do Brasil, pelo apoio à realização deste trabalho.

A todos aqueles que em algum momento me ajudaram a crescer, e colaboraram para meu aprendizado e minha jornada, em especial às minhas avós Juracy e Zenith, aos amigos Castañon, Dalle Ore e Veiga, à minha prima Ivanir, e aos meus grandes e saudosos amigos, Pacheco e Sergio.

A meu filho Lucas e meu pai Gerson, outra vez, pelo árduo esforço de revisão desta dissertação.

Finalmente, Àquele que tudo permitiu, doando-me saúde e os meios necessários para buscar meus objetivos, Deus.

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

ARQUITETURA ROBUSTA E ESCALÁVEL PARA DESCOBERTA DE SERVIÇOS EM REDES AD HOC

Carlos Henrique Pereira Augusto

Maio/2007

Orientador: José Ferreira de Rezende
Programa: Engenharia Elétrica

A descoberta de serviços permite a interação entre nós de uma rede, de forma a cooperarem em atividades ou usufruírem recursos, tanto em arquiteturas cliente-servidor, multicamadas ou *peer-to-peer*. Devido às características de ausência de infra-estrutura e mobilidade, as redes *ad hoc* apresentam um grande desafio na adoção de mecanismos eficientes para esta finalidade. Nossa proposta é apresentar uma arquitetura escalável e robusta para descoberta de serviços baseada numa rede sobreposta (*overlay*) de diretórios criados dinamicamente para cobrir de forma uniforme uma rede *ad hoc*, diminuindo assim o tempo de procura por um determinado serviço e o número de mensagens de controle gerados pela arquitetura.

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

ROBUST AND SCALABLE SERVICE DISCOVERY ARCHITECTURE FOR AD HOC NETWORKS

Carlos Henrique Pereira Augusto

May/2007

Advisor: José Ferreira de Rezende
Department: Electrical Engineering

Ad hoc networks pose a great challenge in the design of efficient mechanisms for service discovery due to the lack of infrastructure along with node mobility. This dissertation proposes a robust and scalable service discovery architecture based on directory nodes organized in an overlay network. In the proposed architecture, directory nodes are dynamically created and removed using adaptive promotion/demotion processes. The aim of these processes is to uniformly cover the entire network while decreasing the query latency for a service and the overhead imposed by control messages.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de acrônimos

AODV : *Ad Hoc On Demand Distance Vector*;

CAN : *Content Addressable Network*;

DHT : *Distributed Hash Table*;

DN : *Directory Node*;

DNS : *Domain Name Service*;

DSDV : *Destination-Sequenced Distance Vector routing*;

DSR : *Dynamic Source Routing*;

IEEE : *Institute of Electrical and Electronics Engineers*;

IP : *Internet Protocol*;

JVM : *Java Virtual Machine*;

JXTA : *Juxtapose*;

LAN : *Local Area Network*;

LDAP : *Lightweight Directory Access Protocol*;

OLSR : *Optimized Link State Routing*;

P2P : *Peer to Peer*;

SLP : *Service Location Protocol*;

TCP : *Transmission Control Protocol*;

UPnP : *Universal Plug and Play*;

WAN : *Wide Area Network*;

XML : *eXtensible Markup Language*;

Capítulo 1

Introdução

1.1 Motivação

Redes *ad hoc*, conforme citado em [2], são um fator chave na evolução das comunicações sem fio. Estas redes herdam os problemas tradicionais de comunicação móvel e sem fio, tais como otimização de banda, controle de potência e melhorias na qualidade de transmissão. Além disto, sua característica de ausência de infra-estrutura leva a novos problemas, como configuração da rede, descoberta de dispositivos, manutenção da topologia, bem como endereçamento *ad hoc* e roteamento.

A comunicação entre dois nós em uma rede *ad hoc* nem sempre é direta, portanto ela pode ocorrer através de múltiplos saltos, onde cada nó atua como um roteador. Além disto, conforme o trabalho em [3], as redes *ad hoc* diferem substancialmente de outras redes existentes, primeiro porque a topologia de interconexão pode ser bastante dinâmica, segundo porque os usuários não realizam ações administrativas, ou de configuração, no estabelecimento da rede, ou seja, ela possui uma característica essencialmente distribuída.

Existem vários desafios no projeto de redes *ad hoc* eficientes, envolvendo camadas de enlace sem fio e roteamento, em especial quanto ao consumo de energia e disponibilidade de banda. Entretanto, existem outros desafios em redes *ad hoc* que vêm sendo objeto de estudo nos últimos anos, tais como segurança e aplicações distribuídas, ou seja, desafios relativos às camadas superiores das arquiteturas de rede. Dentro deste contexto temos a

descoberta de serviços.

Segundo a definição da arquitetura Jini[4] [5], um serviço é uma entidade que pode ser usada por uma pessoa, programa ou outro serviço. Um serviço pode ser uma computação, um espaço ou sistema de armazenamento, um canal de comunicação, um dispositivo de *hardware*, ou outro usuário.

E conforme o trabalho em [6], a descoberta de serviço é definida como o problema de localizar diferentes serviços em uma rede, e é um importante e necessário componente das redes *ad hoc*. Em outras palavras, descoberta de serviço corresponde a fazer a associação entre a descrição do serviço desejado e as informações de rede necessárias para sua adequada utilização.

Como as redes *ad hoc* caracterizam-se pela ausência de infra-estrutura e pela interconexão dinâmica de seus nós, existe uma dificuldade na utilização de servidores específicos e previamente configurados para realizar a descoberta de serviços. Também a utilização de serviços de diretório, tais como LDAP[7], não é de simples adequação num ambiente sem infra-estrutura.

Algumas soluções para descoberta de serviços são conhecidas para redes com fio, entre as quais a já citada Jini[4], além de SLP[8] e UPnP[9], entretanto elas pressupõem a existência de servidores especializados na rede, a manutenção de conexões de transporte entre entidades, e a utilização de mecanismos de inundação, o que as tornam pouco adequadas para redes *ad hoc*.

Além destas, existem outras propostas para redes com fio e Internet, mas que não dependem da existência de servidores específicos, sendo normalmente utilizadas em ambientes *Peer-to-Peer* (P2P). Diversas pesquisas são realizadas nestes ambientes que apresentam grandes desafios em termos de escalabilidade e dinamicidade. Algumas das propostas são Pastry [10], Tapestry [11] , CAN [12], Chord [13] e Symphony [14], e existem inclusive produtos disponíveis, tais como Gnutella [15], e plataformas de desenvolvimento como o JXTA [16]. Entretanto, diferente das redes *ad hoc*, no caso das redes P2P, escalabilidade normalmente se refere à existência de milhares a milhões de nós, e dinamicidade se refere à entrada e saída de nós da rede.

Já as redes *ad hoc*, por não exigirem infra-estrutura e pela carência de recursos de energia e banda disponível, apresentam desafios de escalabilidade para se alcançar algumas centenas de nós cooperando eficientemente. Além disto, a dinamicidade é acompanhada pela mobilidade, com troca de vizinhança, quebra de rotas e demais dificuldades advindas dela, criando assim novos desafios para o projeto de mecanismos eficientes para a descoberta de serviços.

1.2 Objetivos

Neste contexto, esta dissertação apresenta uma proposta de mecanismo de descoberta de serviços em redes *ad hoc*, utilizando-se de alguns conceitos de redes P2P, principalmente aqueles relativos à descentralização, auto-organização e reorganização para recuperação de falhas (*self-healing*), e que também são típicos de redes *ad hoc*. No entanto, nossa proposta não estabelece uma aplicação direta de uma solução P2P em redes *ad hoc*, por crer que ambas possuem escopos distintos, conforme mencionado na Seção 1.1.

Os principais objetivos de nossa proposta são: alcançar escalabilidade, permitindo o seu uso tanto em pequenas redes, considerando o número de nós ou a área coberta, até em redes com número elevado de nós, e com densidade ou grau de conectividade variado; permitir robustez, ou seja, obter efetividade no procedimento de descoberta do serviço frente à mobilidade.

Existem outras propostas para realizar a descoberta de serviços em redes *ad hoc*, conforme mostraremos no Capítulo 2. Entretanto, poucas apresentam resultados analíticos ou por simulação, e muitas ainda apresentam ou problemas de escalabilidade, por exigirem um grande número de mensagens com o aumento do número de nós, ou de robustez, degradando rapidamente com a mobilidade ou com alta taxa de entrada e saída de nós na rede.

Estas propostas utilizam basicamente três tipos de abordagem:

- Inundação da rede [17, 18, 19, 20] - Nestas soluções, ou o cliente inunda a rede em busca do serviço (método *pull*) ou o servidor inunda a rede anunciando seu serviço

(método *push*). Elas sofrem problemas de escalabilidade, no primeiro caso quando o número de requisições pelos serviços aumenta, no segundo caso quando o número de serviços cresce;

- Utilização de *clusters* ou *backbone* virtual [21, 22, 23, 24, 25] - Estas soluções escolhem dinamicamente alguns nós para atuarem ou como *cluster heads* ou como nós do *backbone*. Os demais nós utilizam estes nós para encaminhar suas consultas e/ou o cadastramento de serviços. Nelas, uma vez estabelecido o conjunto de nós especiais, o número de mensagens de controle se reduz consideravelmente. Entretanto existe um custo elevado para manutenção destas estruturas, e a dinamicidade, principalmente no caso de mobilidade, reduz a eficiência de tais propostas.

- Aplicação de soluções de P2P [26, 27] - Propõe a adaptação de soluções de redes P2P para redes *ad hoc*. Normalmente, utilizam técnicas de *cross-layer* para otimizar o funcionamento de uma específica solução P2P associada a um dado protocolo de roteamento para *ad hoc*. Apesar de obter ganhos com a redução do número de mensagens de controle, estas propostas perdem em flexibilidade, e normalmente têm o escopo de aplicação bem específico.

Para alcançar os objetivos citados, apresentamos nossa proposta nos capítulos seguintes. Ela se situa em um híbrido da adoção de *backbone* e de uso de solução de redes P2P. Primeiro, com o objetivo de obter escalabilidade, ela cria uma rede de nós diretórios que cobre (rede *overlay* ou rede sobreposta) a rede toda, mas com a vantagem de que não há necessidade dos nós clientes ou servidores se associarem a um *cluster head* ou a algum nó do *backbone*. Segundo, propõe a adoção de uma solução de redes P2P somente sobre os nós que compõem esta rede sobreposta.

A proposta apresentada constrói uma arquitetura dividida em três camadas: a rede *ad hoc* real; a rede sobreposta, composta somente de nós chamados nós diretórios (DNs), e que possuem função similar ao de *superpeers* de redes P2P [28], mas que são escolhidos de forma dinâmica e adaptativa; e o anel mantenedor da tabela de serviços, composto também por estes nós diretórios. Uma visão geral destas camadas é apresentada na Seção 3.1.

Neste modelo adotado, uma consulta por um determinado serviço deve chegar até

um nó diretório, que responderá a consulta ou a distribuirá na rede sobreposta, composta dos nós diretórios. Para que isto funcione adequadamente algumas premissas devem ser atendidas, tais como:

- Cada nó cliente deve ser capaz de enviar mensagens para pelo menos um nó diretório;
- Cada nó servidor deve ser capaz de enviar mensagens para pelo menos um nó diretório;
- Os nós servidores devem inscrever seus serviços nos nós diretórios;
- Cada nó diretório deve ter conhecimento da existência de outros nós diretórios para que possa encaminhar as consultas a eles, mesmo que indiretamente;
- Os nós diretórios devem oferecer uma cobertura completa da rede, tanto espacial quanto temporalmente;
- Os nós diretórios devem poder ser escolhidos dinamicamente, de forma a atender os requisitos de inexistência de infra-estrutura e pré-configuração em redes *ad hoc*.

1.3 Organização da Dissertação

Esta dissertação apresenta os conceitos gerais envolvidos na arquitetura completa, incluindo um modelo analítico para estimar número de mensagens de controle, e detalha e avalia por simulação os dois primeiros planos da proposta. Para isto ela está organizada da seguinte forma: no Capítulo 2 são apresentados os conceitos gerais sobre serviços, descoberta de serviços e redes *ad hoc* e seus protocolos de roteamento mais comuns, em seguida os trabalhos relacionados, primeiramente em redes centralizadas, posteriormente em redes P2P e finalmente em redes *ad hoc*; no Capítulo 3 é feita a descrição teórica da proposta completa, são apresentadas as mensagens e algoritmos utilizados pela arquitetura, e são descritos diversos mecanismos de otimização da arquitetura com suas respectivas motivações; no Capítulo 4 é apresentado um modelo analítico da construção da rede sobreposta; no Capítulo 5 são discutidos detalhes da implementação da primeira

e segunda camada em ambiente de simulação e são apresentados resultados obtidos por simulação; e no Capítulo 6 finalizamos com as conclusões sobre os resultados obtidos, discorremos sobre as principais contribuições e apresentamos os trabalhos futuros.

Capítulo 2

Descoberta de Serviços

Este capítulo apresenta inicialmente alguns conceitos sobre serviços, redes *ad hoc* e seus protocolos de roteamento. Em seguida, são apresentados alguns trabalhos relacionados divididos em três seções: a primeira abordando trabalhos para redes com administração centralizada; a segunda sobre redes P2P, de forma a esclarecer algumas opções adotadas pela presente proposta; e a terceira apresentando trabalhos referentes à descoberta de serviço em redes *ad hoc*. Na última seção, apresentamos detalhes da proposta "Abordagem por Teoria de Campo" [17], da qual utilizamos diversos conceitos para nossa arquitetura, sugerida inicialmente em [29] e [30].

2.1 Serviços

Um serviço é uma entidade que pode ser usada por uma pessoa, programa ou outro serviço, podendo ser uma computação, um espaço ou sistema de armazenamento, um canal de comunicação, um dispositivo de *hardware*, ou outro usuário, conforme definição da arquitetura Jini [4, 5].

Em Jini, serviços são programas executados em uma Máquina Virtual Java (JVM) que podem ser chamados remotamente por clientes Jini ou outros serviços. Por outro lado, em soluções de P2P para compartilhamento de arquivos, tais como Kazaa [31] e Napster [32], serviços referem-se a arquivos disponíveis.

Estes e outros casos podem se aplicar a redes *ad hoc*. Por exemplo, nos trabalhos em [23, 33] é muito bem detalhada uma aplicação de descoberta de serviços em redes *ad hoc*, onde é descrito o cenário de hospitais de campanha da Cruz Vermelha, e os serviços tanto podem ser os profissionais de saúde, como equipamentos médicos, por exemplo, aparelhos de Raio X.

Em todos estes casos, há a necessidade da descoberta de serviço, ou em outras palavras, associar a descrição do serviço desejado ao endereço de um nó da rede que possa provê-lo. Desta forma, o objetivo básico dos mecanismos de descoberta de serviço é fazer a associação entre o serviço desejado e as informações de rede necessárias para sua adequada utilização pelos demais nós da rede.

A descoberta de serviço é um mecanismo aplicável a diversos tipos de redes, e seu objetivo principal é minimizar tarefas administrativas e aumentar a usabilidade. Diante desta diversidade, é importante a adoção de uma terminologia uniforme e uma forma de classificação das diversas propostas e soluções existentes. Com este objetivo, o trabalho em [1] propõe uma taxonomia para Descoberta de Serviços em Ambientes de Computação Pervasiva que utiliza os dez critérios a seguir para classificação dos mecanismos:

- Nomes e atributos dos serviços – diz respeito a forma como o serviço é descrito. Clientes buscam serviços especificando nomes e atributos, o que é mais flexível do que a adoção de identificadores específicos para computadores e serviços. Alguns protocolos de descoberta de serviço utilizam uma abordagem através de moldes (*templates*), isto é, definem um formato para descrever nomes e atributos, visando reduzir ambigüidades e aumentar a facilidade de adição de novos serviços. Já outros protocolos adotam, além de moldes, identificadores pré-definidos para serviços freqüentemente utilizados.

- Método inicial de comunicação – diz respeito ao modo de transmissão das mensagens iniciais para a descoberta, e pode ser *unicast*, *multicast* ou *broadcast*. *Unicast* é o menos custoso, entretanto pressupõe o conhecimento prévio dos endereços, normalmente através de configuração. *Multicast* permite que clientes, servidores e diretórios troquem algumas mensagens utilizando endereço de grupo *multicast*, determinando dinamicamente quais endereços *unicast* devem ser utilizados. *Bro-*

adcast é similar ao *multicast*, sem a necessidade da criação do grupo. Caso seja usado o *broadcast* na camada de enlace, as mensagens chegam a todos os nós dentro de 1 salto, o que limita o alcance da descoberta. Por outro lado, associando-o ao roteamento e realizando uma inundação, terá a desvantagem de gerar um número muito grande de mensagens, quando aplicado sobre comunicação através de vários saltos.

- Registro e Descoberta – refere-se a forma como as informações sobre os serviços são trocadas, e podem ser baseados em anúncios (registro), consulta (descoberta) ou ambos. Quando baseado em anúncios, os servidores enviam mensagens sobre seus serviços, e os demais nós armazenam esta informação. Quando baseado em consultas, onde clientes enviam mensagens buscando o serviço, recebendo respostas específicas sobre suas requisições, não precisando processar anúncios desnecessários.

- Infra-estrutura de descoberta de serviço – diz respeito à existência de componente especializado na tarefa de descoberta de serviço. Divide-se: em baseado em diretório, que são nós com atuação específica, armazenando informação sobre serviços e respondendo a consultas; e não-baseado em diretório, onde clientes e servidores realizam suas interações diretamente. O modo baseado em diretório ainda pode ser classificado pela forma como os diretórios se organizam, podendo ser plana ou hierárquica.

- Estado da informação sobre serviço – refere-se a forma de manutenção de informação sobre o estado do serviço. Pode ser *soft state*, quando as informações sobre o estado do serviço possuem um tempo de vida, expirando caso não sofra atualização; ou *hard state*, quando uma vez registrado, o serviço não expira. A solução de *hard state* exige menor tráfego de mensagens, entretanto é necessário mecanismo adicional para manter a tabela de serviços consistente.

- Escopo da descoberta - refere-se a opções que podem ser utilizadas para reduzir o espaço de busca. A utilização de escopo da descoberta adequado minimiza a computação em clientes, servidores ou diretórios. Os escopos podem ser baseados na topologia da rede, regras de usuários, informação de contexto ou uma combinação

dos três. Escopo baseado em topologia assume implicitamente que clientes, diretórios e serviços estão em locais próximos. Escopo baseado em regras do usuário oferece outra abordagem, onde clientes, servidores e diretórios são organizados em domínios de acordo com regras, e portanto uma consulta pode ser enviada somente para um subconjunto de nós, aqueles que atendem a regra. Informações de contexto de alto nível definem outro escopo da descoberta, e elas podem ser informações temporais, espaciais ou de atividade do usuário. Por exemplo, pode ser especificada uma localização ou região onde o serviço deve ser buscado, ou restringir a busca a um grupo de nós que estejam realizando alguma tarefa.

- Seleção do serviço – uma vez recebida as respostas às consultas, a escolha do serviço a ser utilizado pode ser automática ou manual. Automática quando o mecanismo escolhe automaticamente um serviço entre os que atendem a consulta. Manual quando o mecanismo retorna ao cliente a lista completa dos serviços que atendem à consulta, e o próprio cliente é que deve realizar a seleção do mais adequado.

- Chamada do serviço - refere-se à associação entre o mecanismo de descoberta e o método de utilização do serviço após a descoberta. Os mecanismos podem ser do tipo localização do serviço, onde só é fornecida a localização ou endereço de rede do serviço; do tipo mecanismo de comunicação, onde a descoberta de serviço também fornece a forma de comunicação, como por exemplo RPC (*Remote Procedure Call*) ou RMI (*Remote Method Invocation*), este último adotado pelo Jini; e por último, do tipo operação da aplicação, como ocorre em UPnP, que define especificamente a forma de operação das aplicações.

- Utilização do serviço – pode ser por aluguel, quando os clientes informam um tempo previsto de uso do serviço, ou liberação explícita, quando é necessário que os clientes informem a liberação do recurso. Estes mecanismos são particularmente importantes quando os serviços necessitam de alguma forma de bloqueio para impedir uso simultâneo.

- Verificação do estado do serviço – pode ser por notificação, quando o serviço notifica o cliente caso ocorra alguma mudança em seu estado; ou por varredura, quando os clientes consultam periodicamente o serviço para obter seu estado.

Com esta taxonomia, o trabalho em [1] apresenta uma comparação, resumida na tabela 2.1, entre os mecanismos INS[34], Ninja SDS[35], DEAPspace[36], Jini[4], UPnP[9], Rendezvous, atualmente chamado de Bonjour[37], Salutation, cujo consórcio foi dissolvido em junho de 2005, SLP[8] e Bluetooth SDP[38].

Em [39] é apresentada outra taxonomia. Neste trabalho, são utilizados oito aspectos do projeto de mecanismos de descoberta de serviço para realizar uma classificação. Estes aspectos são:

- Provedor do serviço – diz respeito a que tipo de nó fornece o serviço de descoberta, ou seja, um terceiro tipo de nó (algum servidor especializado) ou genuinamente distribuído quando todos os clientes e provedores participam;
- Construção – diz respeito à forma de construção da rede sobreposta, podendo ser: configurada manualmente; auto-organizada; ou híbrida;
- Conhecimento prévio – diz respeito à configuração dos nós antes do início de funcionamento da rede. Esta característica é fortemente ligada ao método de construção e pode ser: endereços bem conhecidos; identificador único; ou sem conhecimento prévio.
- Arquitetura – refere-se à arquitetura da rede sobreposta, pode ser caracterizada usando-se a teoria de grafos, podendo ser: grafo trivial, no caso centralizado, quando um único servidor registrando todos os serviços e respondendo as consultas; grafo em árvore; grafo regular, quando os nós são ordenados e nós adjacentes são interconectados; grafo aleatório; e grafo completo.
- Registro – diz respeito à forma de registro dos serviços, que podem ser: locais; por referência; servidor local ou manual.
- Descoberta – chamado no mesmo trabalho também de roteamento das consultas. Dividida em duas estratégias: servidores centrais ou replicados; e o encaminhamento de consultas. Esta segunda estratégia se divide em três técnicas: inundação; reversão (*backtracking*); e roteamento através de estrutura regular.

Tabela 2.1: Resumo da Taxonomia Apresentada em [1]

Critério	INS	Ninja	DEAPspace	Jini	UPnP
Nomeação	N/D	N/D	N/D	Molde	Molde e pré-definido
Comunicação inicial	Uni e multicast	Uni, multi e broadcast	Broadcast	Uni e multicast	Uni e multicast
Descoberta e Registro	Consulta e anúncio	Consulta e anúncio	Anúncio	Consulta e anúncio	Consulta e anúncio
Infra-estrutura	Diretório, plana ou hierárquica	Diretório, hierárquica	sem diretório	Diretório, plana ou hierárquica	sem diretório
Estado	soft	soft e hard	soft	soft	soft
Escopo	Regra	Regra, contexto e topo=LAN	Topo=ad hoc 1 salto	Regra, contexto, e topo=LAN	Topo=LAN
Seleção	Automática	Manual	Manual	Manual	Manual
Chamada	N/D	N/D	N/D	Comunicação	Operação
Utilização	N/D	N/D	N/D	Aluguel	Liberação
Verificação	N/D	N/D	N/D	Notificação	Notificação e varredura

Critério	Rendevouz	Salutation	SLP	SDP	
Nomeação	Molde	Molde e pré-definido	Molde	Molde e pré-definido	
Comunicação inicial	Uni e multicast	Uni e broadcast	Uni, multi e broadcast	Uni e broadcast	
Descoberta e Registro	Consulta	Consulta e anúncio	Consulta e anúncio	Consulta	
Infra-estrutura	Diretório e hierárquica	Diretório e plana	com ou sem diretório	sem diretório	
Estado	soft e hard	hard	soft	soft	
Escopo	Regra	Topo=LAN	Regra e Topo=LAN	Topo=ad hoc 1 salto	
Seleção	Manual	Manual	Manual	Manual	
Chamada	Localização	Operação	Localização	Localização	
Utilização	N/D	Liberação	Liberação	N/D	
Verificação	N/D	Notificação	N/D	N/D	

- Recursos suportados – de acordo com os tipos de recursos suportados, que podem ser: fixos; replicáveis; móveis; dinâmicos; e móveis-dinâmicos.

- Consultas e nomeação – refere-se à forma de nomear e localizar os recursos, e pode ser através de: identificadores únicos e *hashes*; nomeação por cadeia de caracteres (*string*); diretórios; e atributos.

Através destes aspectos, o trabalho em [39] também propõe a classificação dos mecanismos de descoberta de serviço em quatro classes principais:

- Sistemas centralizados com terceira parte – são sistemas que têm um ponto de contato único para os usuários. Normalmente, são configurados manualmente e possuem arquitetura do tipo grafo trivial.

- Sistemas distribuídos com terceira parte – são sistemas com múltiplos contatos. Normalmente, requerem alguma configuração manual, mas estabelecem algum grafo mais complexo que o trivial.

- Sistemas com multicast (ou broadcast) – são sistemas que utilizam *multicast* ou *broadcast*. Normalmente não requerem conhecimento prévio e configuração manual e são restritos a ambientes locais (LAN).

- Sistemas P2P – são os sistemas genuinamente distribuídos. Existem duas subclasses possíveis: aleatórios e regulares. Nos aleatórios, a rede sobreposta é criada aleatoriamente e não há uma estratégia determinística para as buscas, sendo usada normalmente alguma forma de inundação. Nas redes P2P com estruturas regulares, a rede sobreposta é criada baseada em identificadores normalmente obtidos através de função *hash*, e permitem buscas determinísticas na rede.

Os autores ainda identificam um hiato no desenvolvimento de mecanismos de descoberta de serviços, pela falta de mecanismos genuinamente distribuídos que suportem dispositivos móveis e dinâmicos e que usem atributos para busca. Finalmente, os autores apresentam também uma tabela taxonômica com os 8 aspectos, classificando 25 mecanismos de descoberta de serviços, incluindo os já citados Chord, Tapestry, Pastry, CAN, Gnutella, Salutation, Jini, UPnP, SLP, Ninja, LDAP e Napster.

2.2 Redes Ad Hoc e Protocolos de Roteamento

Redes *ad hoc*, conforme definido em [3], são coleções de nós móveis reunidos para cooperação sem a necessidade de intervenção ou ponto de acesso centralizado. Elas diferem substancialmente de outras redes existentes, porque a topologia de interconexão pode ser bastante dinâmica.

Existem vários desafios no projeto de redes *ad hoc* eficientes, envolvendo camadas de enlace sem fio, em especial quanto ao consumo de energia e disponibilidade de banda, uma vez que a comunicação sem fio implica em compartilhamento do meio. Além disto, a falta de hierarquização, a topologia dinâmica, e a falta de configuração prévia são desafios importantes para o estabelecimento de protocolos de roteamento eficazes.

Quatro dos principais protocolos de roteamento para redes *ad hoc* são: AODV[40], DSR[41], DSDV[3] e OLSR[42]. Nosso principal interesse no entendimento destes protocolos se deve às similaridades entre o processo de descoberta de rotas e o processo de descoberta de serviços, e também porque a descoberta de serviço pode fazer uso das mensagens de inundação dos protocolos de roteamento, para realizar a própria descoberta. Diante destas semelhanças, algumas propostas para descoberta de serviços em redes *ad hoc*, como veremos na Subseção 2.5.1, realizam a descoberta de serviço de forma associada à descoberta de rota.

Estes protocolos de roteamento se dividem em reativos, onde a descoberta de rota é realizada sob demanda, ou seja, quando necessária, sendo o caso do AODV e do DSR, e em pró-ativos, quando as rotas são criadas e mantidas previamente em tabelas, já estando disponíveis quando ocorrer a necessidade de uso, sendo este o caso do DSDV e do OLSR. A seguir, apresentamos algumas das características principais destes protocolos.

AODV: *Ad Hoc On demand Distance Vector routing* é um protocolo baseado em vetor distância e do tipo reativo, ou seja, que lança uma descoberta de rota quando há a necessidade de transferência de dados. Ele baseia-se em mensagens de requisição de rotas, os RREQ (*Route Request*), e de resposta de descoberta de rota, os RREP (*Route Reply*). O pacote RREQ contém o endereço do nó originador e do destino que se deseja alcançar, e é enviado quando um nó necessita trocar dados com um destino para o qual ele não conhece

uma rota. Este pacote é enviado em difusão para todos os vizinhos, e encaminhado em um processo de inundação. Um nó que conhece uma rota para o destino que consta no RREQ e recebe este pacote, responde enviando em *unicast* um pacote RREP para a origem. O caminho seguido pelo RREQ estabelece a rota até o destino. Para evitar uma descoberta de rota a cada novo pacote, as rotas descobertas são mantidas em *cache*. Caso uma rota em uso seja rompida, um pacote RERR (*Route Error*) é enviado para a origem, indicando a necessidade de se utilizar outro caminho.

OLSR: *Optimized Link State Routing protocol* é um protocolo do tipo estado de enlace e pró-ativo, ou seja, cria e atualiza tabelas de rotas periodicamente, independente do uso das mesmas. Por ser um protocolo do tipo estado de enlace, ele necessita que cada nó realize uma inundação periódica na rede, informando o estado dos enlaces com seus vizinhos. Uma vez que inundação é uma tarefa muito custosa em redes sem fio, em especial em *ad hoc*, o OLSR utiliza uma forma de otimizar esta inundação, através da seleção de nós que são escolhidos para difundir a inundação, os *MultiPoint Relays* (MPR). A seleção de MPRs é uma heurística para escolha de um grupo de vizinhos de 1 salto que permita o alcance de todos os vizinhos de 2 saltos, com a finalidade de realizar inundação de forma mais eficiente. Um exemplo de funcionamento de MPR pode ser visto na Figura 2.1, onde se o nó O é origem de uma inundação normal, então cada um de seus seis vizinhos deveria retransmitir a mensagem. Por outro lado se somente os nós marcados em cinza forem escolhidos por O como MPRs, então só eles devem retransmitir a mensagem de inundação, que mesmo assim chegará a todos os demais nós.

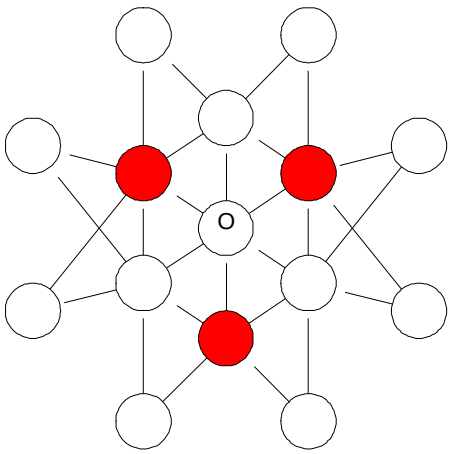


Figura 2.1: Exemplo de Funcionamento de MPR

DSR: *Dynamic Source Routing Protocol for Mobile Ad Hoc Networks* é um protocolo de roteamento reativo para *ad hoc* e baseado em roteamento pela fonte. Ele possui funcionamento semelhante ao AODV, com pacotes de requisição de rotas e de resposta. Entretanto, após a descoberta da rota, o encaminhamento dos pacotes é feito através de roteamento pela fonte, ou seja, cada pacote sai da origem com a informação de todos os saltos que deve percorrer para chegar até o destino. No mecanismo de descoberta de rotas, o DSR possui mensagens também chamadas de RREQ, enviadas em inundação, e RREP, retornadas em *unicast* e que vão acumulando o endereço de cada salto percorrido para permitir o roteamento pela fonte. O DSR também utiliza o *cache* de rotas de forma mais agressiva que o AODV, fazendo que os nós aprendam as rotas utilizadas por pacotes previamente encaminhados por eles na rede.

DSDV: *Highly dynamic destination-sequenced distance vector routing*, é um protocolo pró-ativo do tipo vetor distância, e portanto se assemelha ao RIP [43]. Sua principal diferença é a identificação de pacotes para permitir mais rápida convergência e evitar problema do tipo *contagem para infinito*.

2.3 Descoberta de Serviços em Redes com Administração Centralizada

Descrevemos brevemente nesta seção três soluções para descoberta de serviço em ambientes com administração centralizada, Jini[4], SLP[8] e UPnP[9], largamente citadas pela bibliografia existente. É importante observar que os objetivos destes mecanismos são bem diferentes daqueles existentes em redes *ad hoc*. Neles, a principal motivação é a redução das tarefas de configuração ou a distribuição de aplicação.

Jini: Consiste de um modelo de programação e uma infra-estrutura para atingir os objetivos fundamentais de como clientes e serviços se descobrem e se conectam formando uma comunidade. Escrito inteiramente em Java, Jini usa os mecanismos de RMI (*Remote Method Invocation*) para mover objetos através da rede. Além do serviço e do cliente, Jini usa o conceito de *Lookup Service* que atua como um diretório, onde o serviço se cadastra e o cliente faz a busca pelo serviço. O acesso aos *Lookup Services* é feito através

de endereçamento *unicast* quando previamente configurado e conhecido, ou através de grupo *multicast*.

SLP (*Service Location Protocol*): utiliza três entidades chamadas de *User Agents* (UA) que são os clientes, *Service Agents* (SA) que são os servidores, e *Directory Agents* (DA) usados para prover escalabilidade ao protocolo. Os UAs enviam a mensagem *Service Request* especificando as características do serviço que desejam, e aguardam a mensagem *Service Reply*, que especifica a localização de todos os serviços que atendem a requisição. O SLP permite que os UAs enviem requisições diretamente aos SAs e neste caso elas são enviadas em *multicast*.

UPnP (*Universal Plug-and-Play Architecture*): atua de forma similar ao SLP. Algumas mudanças de nomenclatura existem, e os serviços são normalmente tratados como *devices* e os diretórios são chamados de *Control Points*. Os serviços se cadastram nos *Control Points* através de mensagens de anúncio enviadas em *multicast*. Similarmente, quando um novo *Control Point* entra na rede, ele se anuncia através de mensagens em *multicast*. Detalhes do procedimento de descoberta do UPnP são encontrados na especificação do *Simple Service Discovery Protocol* (SSDP)[44].

2.4 Descoberta de Serviços em Redes P2P

Segundo o trabalho em [45], existem três formas básicas de atuação das redes P2P: centralizada; descentralizada e não estruturada; e descentralizada e estruturada.

A forma centralizada não é uma boa opção para uso em redes *ad hoc*, pois pressupõe a utilização de servidores específicos para armazenamento da tabela de serviços. Os demais nós consultam estes servidores para encontrar que outros nós possuem os serviços desejados. Um dos exemplos de P2P centralizada é o Napster[32].

P2P descentralizada e não estruturada, conforme o próprio nome diz, não possui servidores centralizados nem alguma organização topológica entre os nós. Para encontrar um serviço, os nós consultam seus vizinhos, sendo comum o uso de inundação progressiva, onde a consulta é propagada a todos os vizinhos dentro de um raio, que é sucessivamente aumentado até que o serviço seja encontrado ou um limite máximo de propagação seja

alcançado. Exemplos são Gnutella [15] e Freenet [46].

P2P descentralizada e estruturada também não utiliza servidores centralizados, mas cria uma organização topológica na rede para distribuir as informações sobre serviços e permitir uma busca controlada pelos mesmos. São exemplos de sistemas P2P descentralizados e estruturados o Pastry [10], Tapestry [11], CAN [12], Chord [13] e Symphony [14], onde cada descrição de serviço ou recurso é transformada em uma chave de acesso rápido, índice, através de uma função *hash*, e esta chave é utilizada para a busca do serviço na tabela de serviços.

Entretanto, por ser uma arquitetura descentralizada, esta tabela não pode ficar armazenada em um único local, e portanto, uma vez que tenha sido criada, o grande desafio é distribuir fragmentos dela de forma eficiente e consistente na rede. Então é utilizado o conceito de *Distributed Hash Table* ou DHT.

Nestas propostas de P2P, quando uma consulta é enviada, ela é roteada através de uma rede sobreposta até alcançar o nó responsável por armazenar o fragmento da tabela que contém aquela chave. A estrutura geométrica da rede sobreposta, isto é, a organização da DHT, é a principal diferença entre as propostas existentes.

Pastry [10] utiliza um espaço de endereçamento circular de 128 bits, mantendo uma tabela de roteamento de $\frac{128}{2^b}$ linhas e 2^b colunas, onde b é um parâmetro de configuração normalmente igual a 4. As entradas da linha n desta tabela são endereços de nós que possuem os mesmos n primeiros bits de endereço que o nó em questão, de forma que esta organização permita que o encaminhamento de mensagens seja realizado de modo equivalente a uma árvore.

Tapestry [11] é bastante similar ao Pastry, pois utiliza o mesmo processo de roteamento por prefixos, mas difere na forma como faz o mapeamento de chaves e endereços.

CAN [12] utiliza coordenadas Cartesianas d-dimensionais onde cada nó mantém uma tabela de roteamento com $O(d)$ entradas, e cada destino pode ser encontrado com $O(dN^{\frac{1}{d}})$ passos.

Em Chord [13] e Symphony [14], os nós são dispostos em um anel virtual, onde a cada segmento de arco corresponde uma fração da tabela *hash*. Inicialmente cada nó tem

conhecimento do nó seguinte e do anterior no anel, entretanto uma consulta, somente com este conhecimento, levaria a uma busca circular em todo o anel, que considerando todas as variáveis uniformemente distribuídas, corresponderia em média N/2 nós, onde N é o número de nós da rede.

Para reduzir este espaço de busca, Chord cria conexões entre os nós em aproximadamente N/2, N/4, N/8, etc. do anel. Isto permite saltos dentro do anel e reduz o espaço de busca para $O(logN)$, tornando a busca eficiente. Entretanto, o custo de manutenção e troca de conexões se torna elevado quando ocorrem muitas entradas e saídas de nós da rede. Este funcionamento do Chord pode ser visto na Figura 2.2. Nela, temos um exemplo onde o espaço de endereçamento é de 4 bits, e portanto temos 2^4 endereços. Entretanto, na rede existem somente os nós marcados em cinza, ou seja, 1, 4, 9, 11 e 12. Com esta configuração, cada nó fica responsável por armazenar um fragmento da tabela de serviços correspondente aos endereços entre seu identificador e o do seu antecessor. Por exemplo, o nó 4 armazena as chaves 4, 3 e 2, o nó 1 armazena as chaves 1, 0, 15, 14 e 13, e assim por diante.

Para criar os atalhos, o nó 1, por exemplo, também deverá armazenar informação sobre os nós responsáveis pelas chaves 9, 5 e 3, correspondentes aos endereços $(n+2^{i-1})$, onde n é o identificador do nó (1) e i varia de 1 até o número de bits do endereço (4).

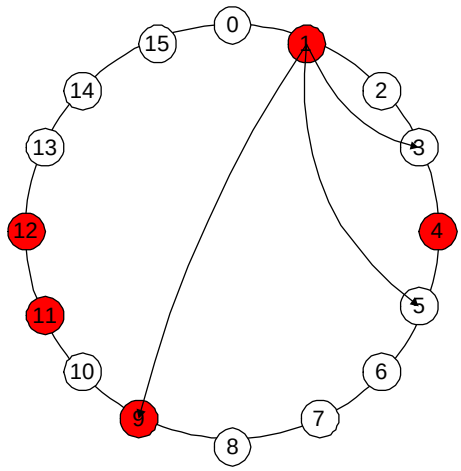


Figura 2.2: Funcionamento do Chord

Alternativamente, Symphony adota a criação de atalhos estabelecidos de forma aleatória através de uma função com distribuição de probabilidade dependente do inverso

do quadrado da distância no anel. Esta idéia é uma aplicação do estudo em [47] sobre *Small World* [48], e torna o custo de manutenção da estrutura por conta da dinamicidade da rede inferior ao do Chord, pois não há mais a necessidade do recálculo das frações do anel quando da entrada e saída de nós, somente o estabelecimento das vizinhanças e das conexões de longa distância.

2.5 Descoberta de Serviços em Redes *Ad Hoc*

Nesta seção apresentamos outras propostas para realização da descoberta de serviços em redes *ad hoc*. Classificamos estas propostas em três tipos: as que utilizam inundação completa na rede, apresentada na Seção 2.5.1; as que criam *clusters* ou *backbones* virtuais, apresentadas na Seção 2.5.2; e por último as que aplicam soluções de redes P2P em redes *ad hoc*, apresentadas na Seção 2.5.3.

2.5.1 Por Inundação da Rede

Em [49] é apresentado Nom, uma proposta onde a rede é inundada pelas mensagens de busca de serviços enviadas pelos clientes, de forma independente do protocolo de roteamento utilizado. Este procedimento, onde o cliente atua ativamente lançando a busca, enquanto os servidores aguardam passivamente pela mensagem, é chamado de método *pull*. Em contraposição, temos o método *push*, onde os servidores anunciam seus serviços ativamente, e os clientes os utilizam conforme aprendem. É possível também conjugar as duas formas em um método híbrido, onde ambos atuam ativamente, tanto os servidores divulgando ou cadastrando seus serviços como os clientes enviando consultas. Estes métodos sofrem problemas de escalabilidade, agravados em redes *ad hoc* pelo compartilhamento do meio de comunicação sem fio.

DEAPSpace [36] é uma das propostas que utilizam inundação para descoberta de serviços em redes *ad hoc* através do método *push*, entretanto, somente considera redes de 1 salto.

No trabalho em [6], há uma avaliação de desempenho de algumas estratégias para descoberta de serviço em rede *ad hoc*. As estratégias avaliadas são referentes a opções

de inundação, chamadas pelos autores de *Post-query*, que correspondem a formas de associação dos métodos *push* e *pull*. Elas são: gulosa (*greedy*), onde todos os nós inundam a rede com anúncios e consultas; incremental, onde a inundação é limitada e crescente a cada rodada; uniforme sem memória, onde os servidores se anunciam para um conjunto aleatório de nós e os clientes também consultam um conjunto aleatório de nós a cada rodada; com memória, onde cada nó armazena os identificadores dos nós para os quais já se anunciou ou os quais já consultou, e a cada rodada somente acrescenta novos nós; e a conservativa, onde cada servidor se anuncia, e cada cliente consulta, somente os vizinhos de 1 salto a cada rodada. Estas cinco estratégias são avaliadas associadas aos protocolos de roteamento DSR e DSDV.

Uma possibilidade para aumentar a escalabilidade nestes métodos de inundação é fazer uso das funcionalidades dos protocolos de roteamento. Como os protocolos de roteamento já possuem a necessidade de realizar algum tipo de inundação, ao se fazer a associação dos dois mecanismos, obtém-se uma grande economia no número de mensagens enviadas.

Em [19], temos um dos trabalhos que avaliam a associação da descoberta de serviço em redes *ad hoc* e protocolos de roteamento. Nele, uma busca por serviço é associada a uma busca por nó através do protocolo AODV, ou seja, quando há a necessidade de se descobrir um serviço, o cliente emprega o método *pull* e inunda a rede junto com a mensagem RREQ do protocolo AODV. O nó servidor, ao receber tal mensagem, envia a mensagem RREP, tanto informando os detalhes do serviço solicitado como permitindo a descoberta da rota para ele. Em [18], temos proposta bem semelhante, também associando a descoberta com mensagens do AODV, na forma de *Internet draft*, porém já expirado.

Em [20], há uma proposta associando a descoberta de serviço ao protocolo de roteamento OLSR. Novamente, o conceito utilizado é propagar a inundação, neste caso, a descrição dos serviços pelo método *push*, associada às mensagens de inundação do protocolo de roteamento.

O que tais propostas fazem é a aplicação de conceitos de otimização de camadas (*cross-layer*), mas essencialmente fazem com que mensagens específicas dos protocolos de roteamento carreguem as mensagens de descoberta de serviço (*piggybacking*). Esta

técnica apresenta bons resultados, pois não altera substancialmente a sobrecarga na rede, conforme resultados que apresentam, entretanto perdem em flexibilidade por ficarem atreladas aos protocolos específicos de roteamento. Desta forma, cremos que a adoção de mecanismos de *cross-layer* deva ser adaptativa e permitir a utilização com qualquer protocolo de roteamento.

Em [50], é proposto o protocolo Konark para descoberta e oferta de serviços, onde os serviços são descritos através de XML [51], aumentando significativamente a flexibilidade. Konark suporta os métodos *push* e *pull*, entretanto o processo de descoberta em si é dependente da pré-existência de roteamento *multicast* na rede *ad hoc*, que os nós utilizam para propagar as diferenças entre as novas mensagens de serviço e as informações já armazenadas.

Em [17] é apresentada uma "Abordagem por Teoria de Campo", uma proposta classificada como ortogonal por não depender de característica alguma do protocolo de roteamento adotado, e que ainda assim pode se valer de mecanismos de *cross-layer* para ser otimizada, por exemplo no processo de inundação da rede. Esta proposta utiliza a idéia de modelar cada instância do serviço como uma carga eletrostática, que gera um campo sobre a rede, e considerando as consultas como cargas de polaridade inversa, sendo atraídas para o serviço de acordo com o gradiente de campo. Entretanto, continua sendo necessária uma inundação na rede para que a informação de campo gerado por uma instância de serviço se propague a todos os nós.

Para diminuir este problema, o mesmo trabalho ainda propõe um mecanismo de redução de inundação, através de *cache* nos nós, mas que só funciona para múltiplas instâncias do mesmo tipo de serviço. Considerando que uma rede pode possuir diversos tipos de serviços distintos, irá existir um tipo de campo para cada tipo de serviço. Além disto, para alguns tipos de serviço, tais como servidores de arquivos e de aplicações *Web*, ainda existe a necessidade de se encontrar o servidor que dispõe do serviço específico, levando não a um anúncio do serviço "compartilhamento de arquivos", por exemplo, mas sim à descrição ou nome do arquivo disponível, o que leva necessariamente a um aumento na relação número de serviços/nó.

Por outro lado, a grande vantagem desta proposta é sua robustez frente à mobilidade,

uma vez que a mensagem de busca por serviço não possui um destinatário específico. Por conta desta característica, que adotamos em nossa arquitetura, este trabalho é descrito com mais detalhes na Seção 2.6.

2.5.2 Utilizando *Cluster* ou *Backbone* Virtual

Alguns trabalhos utilizam teoria de grafos e propõem a adoção de *Cluster* (agrupamento ou aglomeramento de nós) ou a criação de *Backbones* Virtuais para otimizar a descoberta de serviços. O trabalho em [24] propõe uma avaliação de sobrecarga dos protocolos de descoberta de serviço em rede *ad hoc* através de modelos de grafos. Tal trabalho utiliza modelo de grafos aleatórios e faz comparações entre três abordagens: um servidor; por inundação, tanto de clientes quanto de provedores; e através do uso de mediadores.

Na notação utilizada no trabalho em [24], a rede possui n nós. Como é utilizado modelo de grafos aleatórios, existe a probabilidade p de que um nó A receba os pacotes do nó B. Além destes parâmetros, existe a probabilidade P_s de um nó ser um provedor, e portanto o número de provedores b é igual a $P_s \times n$. Analogamente, existe a probabilidade de um nó ser cliente, P_c, e o número de clientes c é igual a $P_c \times n$. Os autores também introduzem a probabilidade $P_{response}$, que é a probabilidade de um provedor ser capaz de responder a requisição de um serviço.

A abordagem com um servidor corresponde a configuração de grafo trivial, e conforme citado pelos autores, não apresenta robustez. Pelo modelo dos autores, a sobrecarga na rede nesta abordagem, desconsiderando o processo de escolha do servidor, corresponde a $n + (b + 2c)(2 - p)$.

Já a sobrecarga na abordagem por inundação por provedores é igual a bn, ou $P_s n^2$. Quando a inundação é realizada pelos clientes, a sobrecarga é igual a $cn + cbP_{response}(2 - p)$. Em ambos casos, podemos verificar uma sobrecarga em $O(n^2)$, indicando limitação quanto à escalabilidade.

A abordagem por mediadores é similar a da adoção de nós diretórios, e corresponde a uma proposta dos autores também apresentada nos trabalhos em [52, 53]. Esta aborda-

gem apresenta uma equação significativamente mais complexa do que as anteriores, e no trabalho em [24], os autores demonstram que ela se torna mais adequada à medida que o número de nós da rede aumenta, indicando escalabilidade.

Em [54, 22, 25], também é apresentada uma proposta deste tipo, onde alguns nós são escolhidos em função da baixa taxa de quebra de enlaces, e com eles é criado um *backbone* virtual através do mecanismo chamado de *Backbone Management* (BBM). Estes nós atuam como nós diretórios, sendo chamados de *Service Broker Nodes* (SBN), e os serviços são registrados e consultados nestes nós através de outro mecanismo, o *Distributed Service Discovery* (DSD), que se utiliza de *multicast* sobre o *backbone* formado.

Em [23, 33], é proposto o mecanismo Hydra, que utiliza conceito de *cluster* para registro e descoberta de serviços, e visa atender a necessidade de hospitais de campanha da Cruz Vermelha. Através de um algoritmo simples, cada nó verifica a existência de um nó diretório (DN) atuando como *cluster head*. Caso não exista nenhum, o nó se torna um DN, garantindo o funcionamento do *cluster*. Entretanto ambos trabalhos não explicitam as mensagens trocadas entre o DN e os demais nós, e também não explicitam como é realizada a comunicação entre os *clusters*, não sendo evidente como é a descoberta de serviços que estejam localizados fora da área do DN.

Em [55, 56], é proposto CARD (*Contact-based Architecture for Resource Discovery in wireless ad hoc networks*), um mecanismo de descoberta de serviço que utiliza conceitos de *Small World* para estabelecer uma rede de contatos de longa distância na rede *ad hoc*. Entretanto, por não usar nenhum mecanismo de localização geográfica dos nós, os contatos são estabelecidos de forma aleatória, se contrapondo ao trabalho em [47], o qual indica que o encaminhamento de mensagens no conceito de *Small World* necessita do conhecimento, pelo menos aproximado, da distância entre os contatos e o destino. Além disto, CARD também necessita da adoção de um protocolo de roteamento que forneça informação completa sobre a topologia numa vizinhança de H saltos de cada nó.

Em [21], o mecanismo proposto utiliza intensivamente informações que são típicas do protocolo OLSR, tais como escolha de *MultiPoint Relays* (MPR), tornando sua implementação mais complexa em conjunto com outros protocolos de roteamento. Esse trabalho propõe a utilização de uma rede sobreposta, onde alguns nós são eleitos como nós diretó-

rios e os serviços são cadastrados nestes nós. Esse processo de eleição é iniciado quando algum nó não identifica nenhum nó diretório em sua vizinhança de H saltos, e ele consiste em uma mensagem de eleição que inunda esta vizinhança. Os nós que aceitarem o pedido de eleição respondem ao iniciador informando suas características. Após um intervalo de tempo, o iniciador escolhe o nó adequado para se tornar DN. Caso ocorra de dois ou mais nós iniciarem a eleição simultaneamente, aquele que possuir menor endereço assumirá o processo de coordenação, realizando sua eleição primeiro.

Uma vez que os DNs são eleitos, os serviços são cadastrados neles, que trocam mensagens de forma a todos manterem a tabela de serviços. Com o objetivo de sumarizar a informação, os DNs utilizam uma estrutura de dados chamada filtro de Bloom para armazenar os serviços. Para permitir que os diretórios troquem entre si as informações armazenadas, os DNs se anunciam através de inundação em uma área com alcance duas vezes maior que a vizinhança utilizada pelos nós ($2 \times H$).

2.5.3 Aplicando Soluções de P2P

Outras propostas adaptam soluções de P2P para realizar descoberta de serviços em redes *ad hoc*. Normalmente, elas são associadas a protocolos de roteamento *ad hoc* objetivando alcançar eficiência na solução.

A proposta em [26, 57] associa a aplicação Gnutella[15] com o protocolo de roteamento OLSR. O principal foco destes trabalhos é avaliar a otimização entre camadas (*cross-layer*), ao associar mensagens e mecanismos do Gnutella a mensagens do OLSR. É importante observar que esta proposta não difere muito daquelas baseadas em inundação, uma vez que este é o mecanismo adotado pelo Gnutella, sendo que esta inundação é realizada junto com àquela realizada pelo OLSR com as mensagens de estado do enlace.

O trabalho em [27] apresenta o mecanismo Ekta, que integra Pastry com DSR. Nesta proposta não é construída uma rede sobreposta, e portanto todos os nós da rede *ad hoc* fazem parte da estrutura Pastry. A principal característica de Ekta é transformar cada entrada da tabela de roteamento do Pastry de forma a não mais guardar um endereço, mas sim um vetor de rotas do DSR. Neste trabalho, há também uma avaliação analítica

indicando que Ekta apresenta número de mensagens de controle em $O(Nlog_{2^b}N)$, onde N é o número de nós da rede e b é o parâmetro de configuração do Pastry.

O trabalho em [58] utiliza Tapestry para descoberta de serviços, mas não em redes *ad hoc*, e sim em redes de sensores. Nesta proposta, um sensor publica sua informação aplicando uma função *hash* sobre a descrição do seu evento e através do roteamento Tapestry armazena a informação <sensor,evento> no nó raiz, ou seja, aquele cujo endereço corresponde, na estrutura Tapestry, à aplicação da função *hash* ao identificador do evento. Quando um cliente deseja consultar o evento, realiza o mesmo procedimento, alcançando a mesma raiz onde está armazenado o endereço do sensor que o publicou.

Em [59] é proposto MPP (*Mobile Peer-to-Peer Protocol*) que também estabelece mecanismos de P2P sobre rede móvel usando otimização entre camadas e adotando uma implementação modificada do protocolo de roteamento DSR chamada de *Enhanced Dynamic Source Routing* (EDSR). Novamente nesta proposta, a inundação realizada pelo protocolo de roteamento é utilizada para realizar também a inundação do mecanismo de descoberta de serviço.

2.6 Abordagem por Teoria de Campo

Nossa arquitetura utiliza, para obter grande robustez frente à mobilidade, algumas características da "Abordagem por Teoria de Campo" proposta em [17].

Conforme citamos na Seção 2.5.1, o trabalho em [17] propõe modelar cada instância do serviço como uma carga eletrostática, que gera um campo sobre a rede. Para que esta informação de campo seja percebida por toda rede, cada servidor inunda a rede com uma mensagem periódica de anúncio. Esta mensagem é retransmitida por todos os nós e possui a informação de número de saltos percorridos, permitindo que cada nó calcule o campo elétrico induzido pela carga (serviço) através de uma função $f(Q,h)$, onde Q é o valor de carga atribuído pelo servidor ao serviço, e h é a distância em número de saltos que a mensagem de anúncio percorreu até chegar ao nó. A função sugerida pelos autores é $f(Q,h) = \frac{Q}{h}$

Cada nó então computará um valor de contribuição de campo eletrostático para cada

instância de serviço da qual ele receba mensagens de anúncio. O somatório destes valores é o valor de campo eletrostático medido por este nó, conforme (2.1).

$$\varphi(n) = \sum_{j=1}^{N} \frac{Q_j}{|n - n_j|} \tag{2.1}$$

O nó n calcula o campo eletrostático $\varphi(n)$ onde n_j são todos os nós servidores e Q_j é a carga do serviço de cada um, informada na mensagem de anúncio, e $|n - n_j|$ é a distância em saltos de n ao servidor n_j.

Cada nó deve enviar este valor aos seus vizinhos através de outra mensagem periódica, a mensagem de *hello*. A partir deste ponto, cada nó tem conhecimento do seu campo eletrostático e o de seus vizinhos, e então considera as mensagens de consulta, que buscam por um determinado serviço, como cargas de polaridade inversa àquelas estabelecidas para os serviços, sendo atraídas para eles de acordo com o gradiente de campo. Ou seja, quando um cliente deseja acessar um determinado serviço, ele envia uma mensagem de consulta (*query*), mas esta mensagem não possui um endereço de destino especificado, pois ela será roteada de forma semelhante a uma mensagem em *anycast*[60]. Portanto, o cliente envia a consulta para o vizinho que informou maior valor de campo na mensagem de *hello*. Este nó, ao receber a mensagem, a encaminha para o próximo vizinho com maior campo, e assim sucessivamente, até que a consulta alcance o servidor, que a responde com uma mensagem em *unicast*.

A grande vantagem que observamos pelos resultados apresentados é a robustez obtida pelo mecanismo mesmo frente a grande mobilidade. Esta característica é alcançada principalmente pelo fato de que as mensagens de consulta não possuem destinatários específicos, podendo ser direcionadas e encaminhadas pelos nós intermediários com base em informações ainda não disponíveis ao cliente no momento em que a mensagem é originada.

Um exemplo de funcionamento da proposta [17] pode ser visto na Figura 2.3. Nela, os nós 0 e 12 são provedores e inundam a rede. Cada nó calcula o campo gerado, representado pelo valor fora do nó. Este valor é enviado aos vizinhos, de forma que cada nó tem conhecimento do campo calculado por todos os vizinhos. Assim, quando o nó 3

necessita acessar o serviço, envia a mensagem de consulta ao vizinho com maior valor de campo, no caso, o nó 2. Esta mensagem segue então o gradiente de campo até chegar ao destino, que para o nó 3 poderia ser tanto o nó 0 como o nó 12, mas pela distribuição de campo será o nó 0.

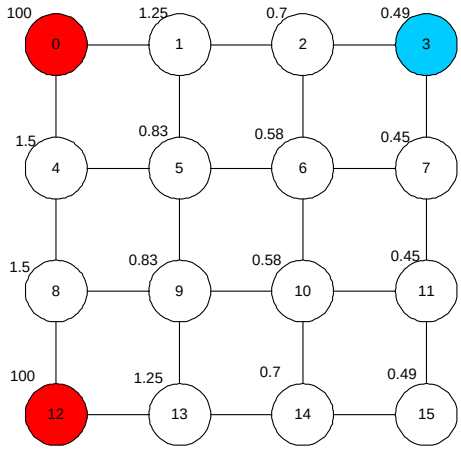


Figura 2.3: Exemplo de Campo Eletrostático

Apesar da grande eficiência mesmo frente à mobilidade, em [29] mostramos que tal proposta tem melhor desempenho para uma dada relação entre número de instâncias do mesmo serviço e número de nós. Para isto, fizemos uma implementação desta proposta em ambiente de simulação e nossos resultados confirmaram também a análise apresentada em [24] para abordagem com inundação usando método *push*, ou conforme nomenclatura usada por aqueles autores, declaração do provedor por inundação, expressa por eles como o produto bn ou $P_s n^2$, conforme citamos na Seção 2.5.2.

Estes resultados foram obtidos através de simulação no ns-2 [61], sem utilizar protocolo de roteamento, uma vez que não consideramos o retorno da mensagem ao cliente, que é realizada em *unicast*. Foram utilizadas as condições descritas em [17], ou seja, área de 1300 por 1500 metros, 100 nós na rede, 10 clientes gerando 4 consultas por segundo, cada um, e nós com mobilidade aleatória com velocidade de até 20 m/s, sem pausa, e mensagens de anúncio e de *hello* enviadas a cada 5s. Optamos, entretanto, por reduzir o tempo de simulação de 1000 para 200 segundos, mas somente iniciando o envio de consultas depois de transcorridos 16 segundos, de forma a retirar o período inicial de convergência do mecanismo. Variamos o número de servidores em 1, 2, 3, 4, 5, 10 e 15, obtendo os gráficos das Figuras 2.4 e 2.5.

Na Figura 2.4 isto é constatado, pois vemos que o número de mensagens por nó por segundo - $NrMsg/nó$ - na proposta em [17], cresce linearmente com o número de servidores. Portanto podemos escrever que $NrMsg/nó = O(nrservidores)$.

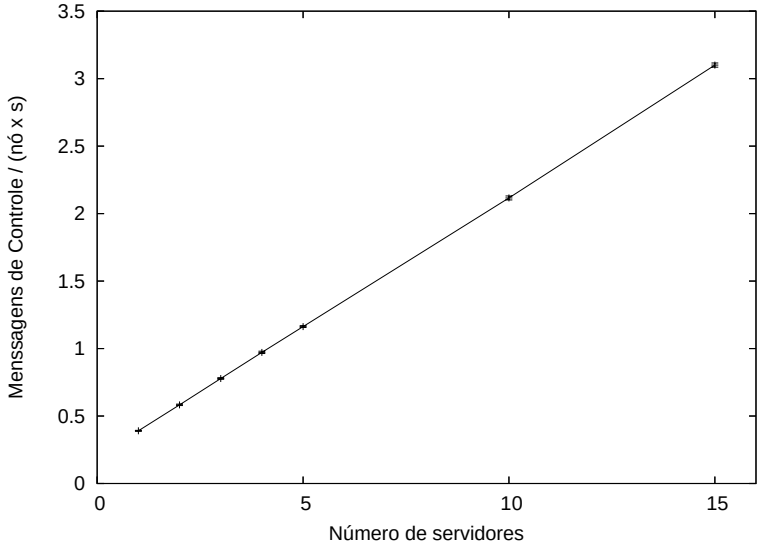


Figura 2.4: Campo Eletrostático - Número de Mensagens de Controle

Se considerarmos que o número de servidores é uma fração constante do número de nós da rede, ou $nrservidores = K * nrnós$, teremos então $NrMsg/nó = O(K * nrnós) = O(nrnós)$.

Esta equação indica um limitante em termos de escalabilidade. Além disto, para alguns tipos de serviço, conforme mencionado na Seção 2.5.1, podemos ter uma elevada relação serviços/nós.

Destes gráficos destaca-se o fato de que o protocolo sofre com a redução do número de servidores, conforme Figura 2.5, apresentando menor índice de descoberta de serviço e maior intervalo de confiança. Desta forma, podemos concluir que a proposta em [17] tem melhor desempenho para uma relação ótima de número de servidores por nó (aproximadamente 1 para 10). Caso o número de servidores seja baixo, a eficiência em termos de descoberta será baixa, e caso o número seja muito elevado, a sobrecarga na rede cresce. Quando considerado o método de redução de inundação através de *cache* intermediário, que reduz o número de mensagens, ainda ocorre o problema de crescimento da sobrecarga na rede, pois o número de *bytes* de controle cresce linearmente com o número de servidores.

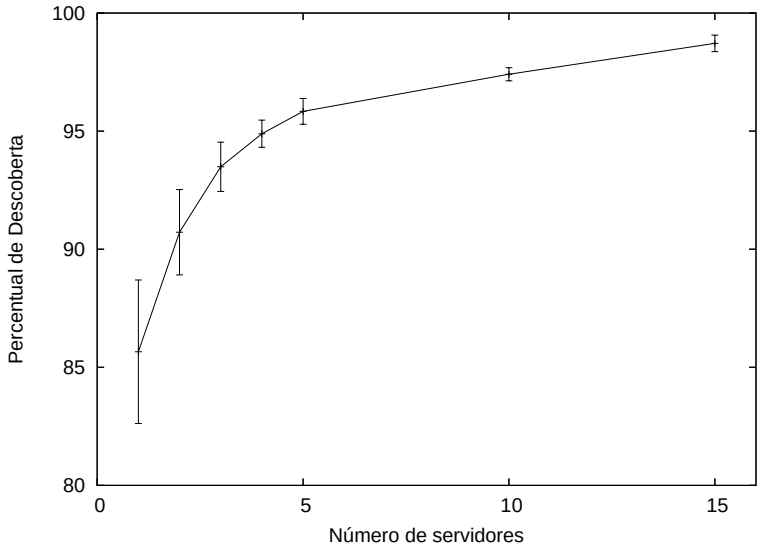


Figura 2.5: Campo Eletrostático - Percentual de Descobertas de Serviço

Estes argumentos reforçam a idéia de utilizar uma rede sobreposta, e portanto nos levou a considerar somente um único tipo de serviço básico - o serviço de nó diretório (DN) - que fornece o serviço de descoberta para os demais nós, de forma análoga à proposta em [21].

Capítulo 3

Arquitetura Proposta

Neste Capítulo é feita a descrição completa da arquitetura de descoberta de serviços, incluindo as 3 camadas que a compõe, e os mecanismos específicos dessa arquitetura proposta, tais como os processos de promoção e despromoção, os *hellos* disparados e a supressão de *hellos*, o algoritmo de conectividade da segunda camada, e os mecanismos de procura no anel da terceira camada.

3.1 Visão Geral

Nossa arquitetura possui três níveis de abstração, ou camadas: a primeira referente à rede *ad hoc* real, definida pela distribuição geográfica ou topográfica dos nós; a segunda obtida com a rede sobreposta, composta de uma parcela dos nós da rede real, que são os nós diretórios ou DNs; e a terceira composta pelos mesmos nós da segunda camada, entretanto arranjados logicamente em uma estrutura para realizar a distribuição da tabela de serviços.

Na arquitetura, existem 4 tipos de nós móveis: *clientes*, os que desejam fazer uso de algum serviço; *provedores*, aqueles que mantêm alguma instância de serviço; *nós diretórios* (DNs), aqueles que armazenam uma parte da tabela de serviços; e *nós comuns*, qualquer nó da rede *ad hoc* sem função especial no momento, mas que executam os algoritmos da arquitetura e podem encaminhar mensagens. Qualquer nó da rede *ad hoc* pode assumir qualquer dos 3 primeiros tipos de papéis de nós definidos, a qualquer momento, inclusive simultaneamente.

Nesta composição em camadas, o serviço da primeira camada é realizar a seleção dos nós diretórios e permitir que clientes e provedores enviem suas mensagens a algum DN. Já a segunda camada fornece o serviço de transformar a rede sobreposta criada pela seleção dos DNs em uma rede conectada. Por último, a terceira camada cria uma organização lógica na rede conectada estabelecida pela segunda camada, de forma a facilitar a busca e cadastramento de serviços nela.

Nossa proposta é adotar um mecanismo de gradiente de campo similar à proposta em [17] para o encaminhamento de mensagens na primeira camada, associado a um mecanismo que propomos, chamado de Processo de Promoção, descrito na Seção 3.3. Para a segunda camada propomos um algoritmo de conectividade, descrito na Seção 3.5. Finalmente, na terceira camada, propomos que os nós diretórios se organizem em um anel similar a idéia proposta em Symphony [14] e citada ao final da Seção 2.4, entretanto não realizamos esta implementação, tendo sido adotado para avaliação do mecanismo uma solução de inundação do cadastro na segunda camada, conforme descrito no Capítulo 5 na Seção 5.1.

Objetivando a escalabilidade, nossa proposta evita fazer uma inundação completa na rede. Cada nó diretório (DN) inunda periodicamente somente sua vizinhança por H saltos, através de mensagens de anúncio, de forma que todos os nós dentro desta vizinhança tenham conhecimento sobre sua existência e calculem uma contribuição gerada por ele, de acordo com a distância em número de saltos a que eles se encontram do DN.

Uma vantagem da utilização de nós diretórios é a possibilidade de resposta múltipla para uma consulta. No caso da existência de diversas instâncias de um mesmo serviço, o nó diretório pode responder com todas as possibilidades, deixando a cargo do cliente a escolha do servidor mais adequado.

De forma análoga à proposta em [17], descrita na Seção 2.6, cada nó calcula um somatório correspondente à composição das contribuições de todos os DNs que se anunciaram a ele. Também como o proposto em [17], ou mesmo na nossa proposta inicial em [29] e [30], esta informação é inserida em uma mensagem de *hello* que é enviada para os vizinhos de 1 salto. Entretanto, a técnica de supressão de *hellos*, descrita posteriormente na Seção 3.4.1, faz com que essa informação seja também inserida na mensagem de anúncio,

reduzindo significativamente o número de mensagens utilizadas pela arquitetura. Todas estas informações sobre vizinhos de 1 salto e DNs são mantidas em *soft state*, expirando e sendo removidas caso não sejam atualizadas após um tempo configurável.

Para proporcionar a cobertura de toda a rede por nós diretórios, as regiões sem DNs deverão realizar a promoção de um nó comum a nó diretório. Escolhemos nomear este processo de promoção para distinguí-lo de um processo de eleição como o proposto em [21] ou em outros trabalhos, onde há uma troca de mensagens entre os diversos nós para decidirem, segundo algum critério, qual será o nó escolhido. No nosso caso, a decisão de promoção é local e independente de trocas adicionais de mensagens além das já realizadas pelo mecanismo de anúncio. Apesar de baseado somente em informações locais, este processo de promoção é adaptativo, e permite uma cobertura uniforme da rede com um número controlado de nós diretórios, conforme demonstraremos no Capítulo 4. Este processo de promoção é detalhado junto com o processo de despromoção na Seção 3.3.

Os nós podem possuir uma predisposição a se tornarem DNs. Isto pode ser obtido através de configuração e os principais fatores de escolha utilizados seriam: energia disponível; capacidade de processamento; mobilidade; e interesse intrínseco em promoção, que pode ocorrer, por exemplo, quando um nó possuir muitos serviços disponíveis. No processo de promoção, a predisposição somente afeta a escolha do tempo aleatório utilizado para uma nova verificação das contribuições dos nós diretórios, fazendo os nós menos dispostos aguardarem intervalos maiores. Em nossa implementação não fizemos nenhuma distinção entre os nós, possuindo todos a mesma predisposição.

Para o funcionamento da arquitetura são necessárias outras duas mensagens na primeira camada: as mensagens de cadastro, que permitem aos nós provedores cadastrarem seus serviços nos DNs, e as mensagens de consulta, utilizadas pelos clientes para obterem dos DNs as informações sobre os serviços. Quando do envio ou encaminhamento destas mensagens, tanto de consulta quanto de cadastramento, cada nó envia ao seu vizinho que possua maior somatório de contribuições, seguindo este gradiente até o nó diretório destino.

Na segunda camada, cada nó diretório mantém uma tabela de vizinhos virtuais, ou vizinhos na rede sobreposta, através do recebimento das mensagens de anúncio, que atuam

para esta segunda camada de forma similar às mensagens de *hello* para a primeira. Esta tabela de vizinhos é utilizada para encaminhamento das mensagens de consulta e cadastro na rede sobreposta. Ao receber uma consulta, o DN verifica se possui a informação sobre o serviço em *cache*. Caso contrário, deverá encaminhar a mensagem na rede sobreposta utilizando-se desta vizinhança virtual. Portanto, a rede formada pelos nós diretórios, considerando os enlaces de múltiplos saltos estabelecidos pelas mensagens de anúncio, deve ser conectada. Para isto propomos um algoritmo de conectividade da segunda camada, descrito na Seção 3.5, que proporciona a promoção de mais alguns nós para exercerem a atividade de DN, e permitir o estabelecimento desta conectividade.

Na terceira camada, para a distribuição da tabela de serviços na rede sobreposta, propomos uma simplificação do trabalho em [14]. O mecanismo cadastra cada serviço numa tabela *hash*, e distribui fragmentos desta tabela, conforme o que é chamado de DHT, mas considerando que os requisitos de escalabilidade no caso de redes *ad hoc* são de ordem de grandeza inferior aos de redes P2P. Para isto, o anel estabelecido é dividido em setores, e cada nó diretório escolhe seu setor de localização, através da aplicação de uma função *hash* ao seu identificador. Entretanto, em nossa proposta os contatos de longa distância são também vizinhos virtuais na rede sobreposta, e portanto são vizinhos de H saltos na primeira camada, isto é, a rede real. Esta proposta é similar à adoção de *Proximity neighbor selection* (PNS) no Chord, avaliada em [62], com a vantagem de que a proximidade é estabelecida pelas mensagens de anúncio, que já são enviadas periodicamente na arquitetura.

As três camadas da arquitetura podem ser observadas na Figura 3.1, onde no plano inferior está a rede real, da qual alguns nós são destacados para serem nós diretórios. No plano intermediário está a rede sobreposta que é composta somente de nós diretórios, no caso, nomeados A, B, C, D, E e F. No plano superior, está o anel que mantém a DHT, onde os nós se organizam de acordo com a seqüência de setores, mas estabelecem vizinhanças de longa distância semelhante à descrita em [14], sendo que estas ligações de longa distância são criadas quando da existência de vizinhanças virtuais na segunda camada. Por exemplo, a ligação A-C no anel decorre do fato de existir vizinhança entre A e C na rede virtual, e que é conseqüência de A e C serem vizinhos de até H saltos no primeiro plano.

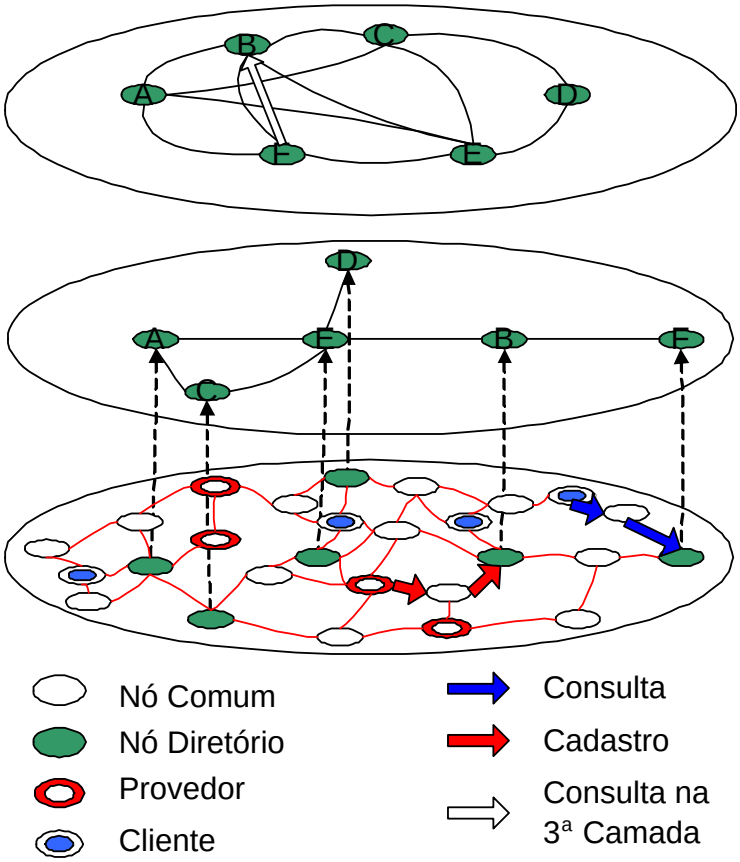


Figura 3.1: Diagrama de Camadas da Arquitetura

3.2 Descrição da Arquitetura

Nesta seção apresentamos os detalhes das 3 camadas que compõem nossa arquitetura, incluindo as mensagens utilizadas e suas funções.

3.2.1 Primeira Camada: Rede *Ad Hoc* Real

É a rede composta por todos os nós, e portanto nela temos os quatro tipos de nós definidos na arquitetura. É importante observar que todos os nós podem assumir um ou mais tipos de papéis definidos na arquitetura.

As mensagens utilizadas nesta camada são:

- Mensagem de Anúncio: enviada periodicamente pelos DNs em *broadcast*, inundando a rede em até H saltos. Esta mensagem possui um identificador, e cada nó

que a recebe pela primeira vez, decrementa o ttl, e se este ttl não se tornar nulo, encaminha a mensagem aos vizinhos. Além destas informações, diferentemente do que propomos em [29] e [30], ou do proposto por [17], cada nó também insere nesta mensagem o mesmo valor de contribuição que é enviado na mensagem de *hello*, permitindo a implementação da técnica de supressão de *hellos*, descrita na Seção 3.4.1.

- Mensagem de *Hello*: é enviada periodicamente por todos os nós, em *broadcast*, aos seus vizinhos de 1 salto. Esta mensagem carrega a informação de contribuição dos DNs que o nó calcula usando a expressão:

$$Contrib_n = \sum_{\forall\, DN \in V_{H_n}} \frac{1}{h}$$

onde V_{H_n} é a vizinhança de H saltos do nó n e h é a distância em saltos de cada DN até n. Outras funções ou métricas poderiam ser adotadas, como por exemplo velocidade ou qualidade dos enlaces, entretanto não foram objeto de estudo desta dissertação.

- Mensagem de Consulta: enviada pelos clientes em busca de um serviço, seguindo o gradiente de contribuições até alcançar um DN. Esta mensagem se comporta como uma mensagem em *anycast* [60].

- Mensagem de Cadastro: enviada pelos provedores para cadastrar seus serviços, seguindo o gradiente de contribuições até alcançar um DN. Esta mensagem utiliza o mesmo mecanismo de gradiente da Mensagem de Consulta.

- Mensagem de Resposta à Consulta: enviada pelo DN, em *unicast*, ao cliente que enviou uma mensagem de consulta. Esta mensagem carrega as informações que o DN armazenou quando recebeu a mensagem de cadastro correspondente ao serviço que está sendo buscado.

- Mensagem de Despromoção: enviada pelo DN quando ele, através do processo de despromoção descrito na Seção 3.3, decide se tornar um nó comum. Esta mensagem é bem similar à mensagem de anúncio, pois é um *broadcast* de H saltos, mas não é periódica, sendo enviada somente na ocasião da despromoção. Todos os

Tabela 3.1: Mensagens da Primeira Camada da Arquitetura

Código	Mensagem	Modo de Envio	Expiração
0x01	Hello	broadcast	1 salto
0x02	Anúncio	broadcast	H saltos
0x03	Consulta	anycast	N $\times$ H saltos
0x04	Resposta à Consulta	unicast	
0x05	Despromoção	broadcast	H saltos
0x06	Cadastro	anycast	N $\times$ H saltos

nós, ao receberem esta mensagem, removem a informação de contribuição do DN correspondente.

Na Tabela 3.1 podemos ver a relação das mensagens da primeira camada e seus respectivos códigos de identificação, modo de envio e número de saltos que elas alcançam.

3.2.2 Segunda Camada: Rede Sobreposta de DNs

A segunda camada cria uma rede sobreposta composta de nós diretórios escolhidos pelo processo de promoção. Por esta camada, são encaminhadas as mensagens de cadastro e consulta de forma a construir a terceira camada, e por isso é necessário que ela seja uma rede conectada. Para isto, as próprias mensagens de anúncio da primeira camada são utilizadas para estabelecer a conectividade, entretanto o processo de promoção não é necessariamente suficiente para que a rede sobreposta seja conectada, e por este motivo criamos um algoritmo, detalhado na Seção 3.5, que torna alguns nós em candidatos a promoção quando eles identificam que podem eventualmente servir para interligar duas ou mais possíveis partições da rede sobreposta. Neste caso, a promoção não é mais realizada através de um processo local, mas estabelecida em função das respostas obtidas da rede de nós diretórios.

Nesta camada, as mensagens utilizadas são:

- Mensagem Sobreposta de Consulta: enviada pelo DN a outro DN, quando não é possível responder imediatamente a uma mensagem de consulta. Esta mensagem

segue pela rede sobreposta conforme a estrutura criada pelos algoritmos da terceira camada. Por exemplo, em uma estrutura tipo Chord, a mensagem seria enviada ao nó sucessor no anel, ou ao nó correspondente ao atalho referente à chave que está sendo buscada.

- Mensagem Sobreposta de Cadastro: enviada pelo DN que recebe a mensagem de cadastro na primeira camada, ao DN que armazena a chave correspondente ao serviço, de forma similar à mensagem sobreposta de consulta.

- Mensagem Sobreposta de Resposta: enviada pelo DN que possui o serviço cadastrado em resposta à mensagem de consulta.

- Mensagem Nó Candidato a Ponte: esta mensagem é enviada por um nó comum quando ele constata que pode haver um particionamento da rede sobreposta e que sua promoção pode permitir que a rede sobreposta se torne conectada. Esta mensagem é similar às mensagens de consulta, seguindo o gradiente de contribuições até alcançar um DN. Nela, o nó candidato informa quais nós diretórios que ele se candidata a estabelecer conexão da segunda camada.

- Mensagem Busca por DN: o DN quando recebe uma mensagem de nó candidato a ponte, deve verificar se já possui conectividade de segunda camada, direta ou indiretamente, com o DN informado. A conectividade direta é verificada na tabela de vizinhos de segunda camada, e caso ela exista, a mensagem de busca por DN não é enviada. A conectividade indireta é verificada através do envio desta mensagem, que segue a estrutura da terceira camada até chegar a um DN que tenha conhecimento do DN buscado, ou esgotar a busca sem encontrá-lo.

- Mensagem DN Encontrado: enviada em *unicast* pelo DN que recebeu a mensagem nó candidato a ponte, quando constata a existência de todos os DNs que poderiam ser conectados, na rede sobreposta. O destino desta mensagem é a origem da mensagem nó candidato a ponte, e o mesmo quando a recebe, toma conhecimento que a rede não está particionada e portanto não necessita se promover.

Na Tabela 3.2, podemos ver a relação das mensagens da segunda camada e seus res-

Tabela 3.2: Mensagens da Segunda Camada da Arquitetura

Código	Mensagem	Origem	Destino
0x07	Sobreposta de Cadastro	DN que recebe Cadastro	DN que armazena dados do serviço
0x08	Nó Candidato a Ponte	Nó comum	Qualquer DN
0x09	Busca de DN	DN que recebe Nó Candidato a Ponte	DN que consta na lista da Nó Candidato a Ponte
0x0A	DN Encontrado	DN que recebe Nó Candidato a Ponte	nó origem da Nó Candidato a Ponte
0x0B	Sobreposta de Consulta	DN que recebe Consulta	DN que armazena dados do serviço

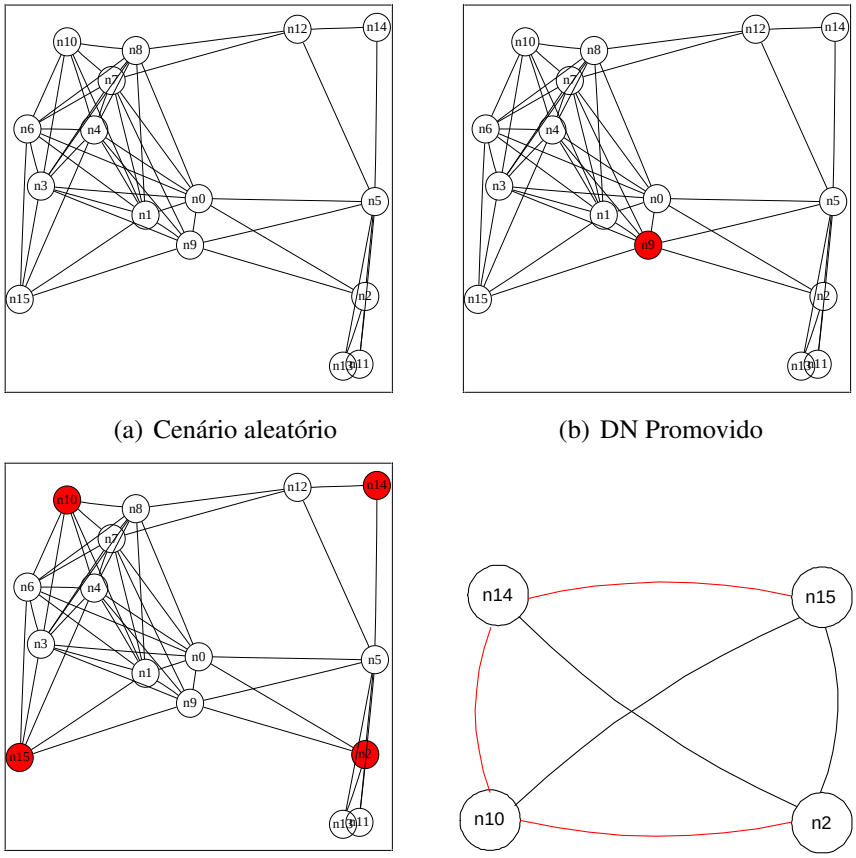
pectivos códigos de identificação, bem como os nós de origem e de destino destas mensagens.

3.2.3 Terceira Camada: Rede de Distribuição dos Serviços

Esta camada possui os mesmo nós da segunda camada, entretanto os organiza em uma estrutura, de forma a minimizar as mensagens trocadas. Nossa proposta é utilizar uma solução de estruturação similar a de P2P nestes nós, mas levando em consideração que os DNs são somente uma parcela da rede, e a sobrecarga de mensagens envolvidas deve ser proporcionalmente menor. É importante observar que nossa proposta utiliza a rede sobreposta somente para encaminhamento das mensagens de consulta e cadastro de serviço, e desta forma todos os demais funcionamentos da rede podem se basear no protocolo de roteamento adotado. Para superar os problemas de dinamicidade impostos pelas redes *ad hoc*, propomos a adoção de uma simplificação do Symphony [14] ou do Chord com PNS [62] para organização da rede sobreposta, entretanto sem o estabelecimento de conexões TCP, mas somente de conexões virtuais mantidas como *soft-state* por mensagens periódicas.

Podemos ver um exemplo da construção do anel na terceira camada na execução da arquitetura apresentada na Figura 3.2.

Na Figura 3.2(a), temos um cenário construído de forma aleatória, constando de 16 nós com alcance de $200m$ distribuídos em uma área de $400m \times 400m$. O ttl adotado para



(a) Cenário aleatório

(b) DN Promovido

(c) DNs Promovidos

(d) Anel DHT

Figura 3.2: Exemplo de Execução da Arquitetura

os anúncios é igual a 2. Na Figura 3.2(b), vemos uma execução da arquitetura sobre este cenário. Nesta execução, o nó *n9* foi promovido e seu anúncio alcança toda a rede, e portanto, o anel de distribuição dos serviços é composto de um único nó e a arquitetura tem o comportamento correspondente a solução do tipo *um servidor*, apresentada no trabalho em [24].

Por outro lado, vemos na Figura 3.2(c) outra execução da arquitetura sobre a mesma rede. Como as escolhas de tempos de envio de mensagens são aleatórios, cada execução pode levar a diferentes configurações. Neste caso, ocorreu a promoção de 4 nós, *n2, n10, n14 e n15*. Na rede sobreposta, vemos que *n14* é vizinho de *n2*, que é vizinho de *n15*, que por sua vez é vizinho de *n10*. Os nós *n10* e *n14* não são vizinhos, pois estão a 3 saltos de distância, e o anúncio de um não alcança o outro. Esta rede é conectada, e para o estabelecimento do anel é necessário que cada nó identifique seu sucessor. Utilizando o identificador dos nós para ordená-los, o anel se estabelece com a seqüência *n2, n10, n14,*

n15, conforme apresentado na Figura3.2(d). A conectividade da rede sobreposta, ou seja, segunda camada, estabelece também o arco *n15, n2* do anel. Os demais arcos devem ser estabelecidos por um Algoritmo para Construção do Anel, que não especificamos, e os atalhos *n2, n14* e *n10, n15* são obtidos também dos enlaces da segunda camada.

Este Algoritmo para Construção do Anel não foi especificado e foi considerado trabalho futuro, para com isto, aplicarmos o tempo e os esforços de desenvolvimento da arquitetura nas duas primeiras camadas. Desta forma, implementamos uma simplificação, baseada na simples inundação da segunda camada, sem criar a estrutura na rede sobreposta. Esta solução gera uma sobrecarga na arquitetura da ordem do número de arestas da rede sobreposta, ou no pior caso, de $O({N_{DN}}^2)$. Ao utilizar uma estrutura em anel, uma busca ou cadastro geraria uma sobrecarga de $O(N_{DN})$, e usando Symphony ou Chord a sobrecarga seria somente de $O(logN_{DN})$.

3.3 Processos de Promoção e Despromoção

Em [29] e [30], propomos a adoção do processo de promoção para escolha dos nós que se tornam DN. O processo é bem simples: se um nó possui valor de contribuição igual a 0, então ele não possui nenhum DN na sua vizinhança de H saltos, e então passa ao estado "Promovendo". Este estado caracteriza-se simplesmente pela espera de um tempo aleatório dentro de uma janela. Após este tempo, se o seu valor de contribuição continua nulo, ou seja, nenhum outro nó se promoveu antes, então o nó se promove, indo para o estado "Nó Diretório" e passa a emitir mensagens de anúncio, que informam a outros possíveis nós que estejam no estado "Promovendo" que devem sair deste estado e retornar ao estado "Nó Comum".

Portanto, o estado "Promovendo" é transitório com duração de um valor aleatório, entre 0 e 1, multiplicado pela janela máxima de promoção. Após este período transitório, o nó ou é promovido, ou seja, passa para o estado "Nó Diretório", ou retorna para o estado "Nó Comum", e continua periodicamente realizando a mesma verificação para decidir se permanece "Nó Comum" ou passa ao estado "Promovendo".

Nos trabalhos iniciais, verificamos a efetividade do processo de promoção para rea-

lizar a cobertura da rede, e também verificamos que o mesmo permite obter um número limitado de mensagens de controle da primeira camada. Entretanto, para permitir além da cobertura da primeira camada, também a conectividade na segunda, introduzimos uma pequena modificação no processo de promoção, alterando o limiar de promoção de 0, para uma variável configurada, que chamamos "Limiar de Promoção". O comportamento do processo permanece o mesmo, e pode ser visualizado no lado direito da Figura 3.3.

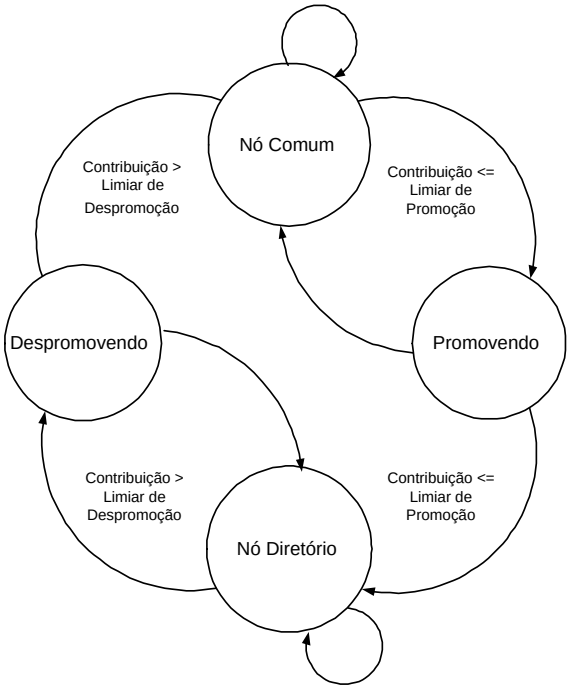


Figura 3.3: Máquina de Estado - Processos de Promoção/Despromoção

No lado esquerdo da mesma figura, observamos o processo inverso correspondente, o processo de despromoção. A principal motivação para utilização deste processo é a possibilidade de após a estabilização do processo de promoção, com a cobertura completa da rede, a mobilidade dos nós provocar a concentração de DNs em uma área em detrimento de outras, que necessitariam de realizar novas promoções. Este tipo de mobilidade induz um aumento desnecessário do número de DNs na rede. Para resolver este problema, introduzimos o processo de despromoção, similar ao de promoção. Nele, um DN ao constatar que possui informação de contribuição muito elevada, ou seja, que há um grande número de DNs a poucos saltos de distância, entra no estado "Despromovendo". Após uma janela de tempo aleatória, caso a situação persista, ou seja, um número suficiente de DNs não se deslocou ou se despromoveu, este nó se despromove, passando ao estado de "Nó Co-

mum" e enviando uma mensagem de despromoção à sua vizinhança de H saltos. Cada nó que recebe esta mensagem de despromoção remove o valor de contribuição que possuía para aquele nó. Também de forma similar ao processo de promoção, caso após a janela de tempo o valor de contribuição não seja mais superior ao "Limiar de Despromoção", então o nó permanece atuando como DN, e verificando periodicamente a situação de seu total de contribuições frente ao limiar.

3.4 Escalabilidade e Robustez

Várias técnicas foram utilizadas para aumentar a eficiência e robustez da arquitetura. Elas são descritas nas subseções seguintes.

3.4.1 *Hellos* Disparados e Supressão de *Hellos*

A primeira técnica é a utilização de *hellos* disparados. O conceito empregado é divulgar imediatamente modificações percebidas na rede. Estas modificações podem ser originadas por mobilidade, falhas, ou entrada e saída de nós na rede.

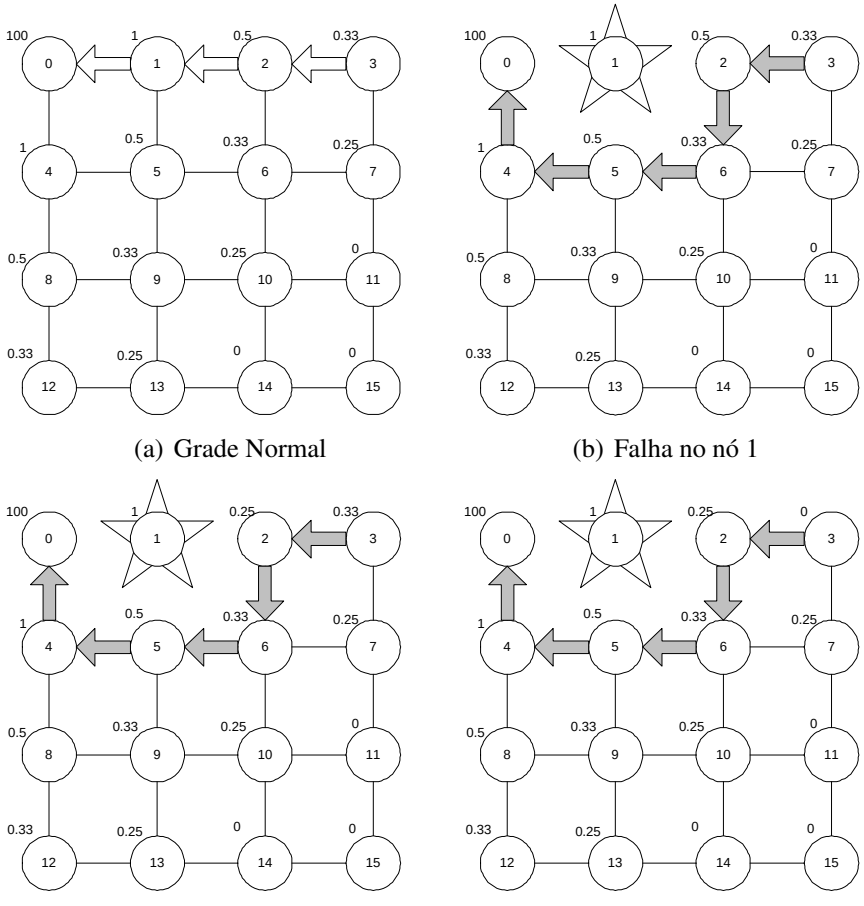
As modificações, percebidas por um nó, que selecionamos para serem divulgadas imediatamente são:

- Recebimento de anúncio de um novo DN;
- Expiração de um DN, pelo não recebimento de anúncios;
- Mudança do gradiente;
- Perda de vizinho informada pela camada de enlace.

Um exemplo destas melhorias pode ser visto na Figura 3.4. Na Figura 3.4(a), há uma rede em grade onde o nó 0 é um DN e usa uma mensagem de anúncio de 4 saltos, induzindo o campo apresentado na rede. O nó 3 é um cliente (ou provedor) e sua mensagem de consulta (ou cadastro) usa o caminho 3-2-1-0. Na Figura 3.4(b), o nó 1 falha e a camada MAC informa ao nó 2 da quebra do enlace. Então, o nó 2 remove o nó 1 da sua tabela

de vizinhos, e para encaminhar a mensagem, ele busca uma nova entrada nesta tabela. Entretanto, a restrição de retorno de mensagem, detalhada na Seção 3.4.3, não permite que o nó 2 encaminhe esta mensagem de volta para o salto anterior, e portanto o novo caminho para a mensagem será 3-2-6-5-4-1 como mostrado na Figura 3.4(c).

Apesar da arquitetura permanecer funcionando, há desta forma uma inconsistência no gradiente na rede. Os *Hellos* Disparados auxiliam na convergência mais rápida, pois os nós enviam mensagens de *hello* sempre que identificam uma mudança importante na rede. Uma das mudanças importantes é a perda de gradiente, como ocorre quando o nó 2 percebe a falha do nó 1. Outra mudança importante ocorre quando um novo valor de campo é calculado, como quando, na Figura 3.4(c), o nó 2 recebe um novo anúncio do nó 0 e computa um novo campo através de um número maior de saltos. Desta forma, o nó 2, através dos *Hellos* Disparados, imediatamente envia um *hello* com este novo valor, e a rede obtém um campo consistente como na Figura 3.4(d).



(a) Grade Normal (b) Falha no nó 1

(c) Gradiente Inconsistente (d) Convergência da grade

Figura 3.4: Grade com Falha de Nó

Entretanto, um problema introduzido pelos *hellos* disparados é o aumento do número de mensagens de *hello*, que se torna especialmente significativo quando a rede possui um grande número de nós iniciando com a função de DN já habilitada.

Para minimizar este problema, desenvolvemos a técnica de supressão de *hellos*. Esta técnica consiste na fusão das mensagens de anúncio e de *hello*. Com esta técnica, quando um nó recebe um anúncio, calcula o valor de $Contrib_n$ e insere esta informação na mensagem de anúncio que será encaminhada. Quando outros nós recebem este anúncio, eles atualizam sua tabela de vizinhos de 1 salto com esta informação. Entretanto, se nenhum anúncio é enviado ou encaminhado pelo nó dentro do período de *hello*, então o nó envia um *hello* normal. Com isto, cada vez que um nó envia ou encaminha uma mensagem de anúncio, ele também envia a informação de contribuição calculada, possibilitando ao mesmo tempo a agilização de convergência proposta pelos *hellos* disparados e a redução do número de mensagens de controle.

3.4.2 Eliminação de Máximo Local

O mecanismo de cálculo de contribuições ou campos, conforme cogitado em [17], possibilita a criação de máximos locais. A ocorrência de máximo local significa que algum nó intermediário possui valor de campo superior ao de todos os vizinhos, o que implica em um comportamento similar a *loop* de roteamento para o encaminhamento através de gradiente.

Em nossa arquitetura, com a distribuição de nós diretórios pela rede e a utilização de um ttl para descartar as mensagens de anúncio, observamos a ocorrência de eventuais máximos locais. Para eliminar estas ocorrências, extremamente danosas para a eficiência da primeira camada da arquitetura, implementamos uma solução simples. No processo de promoção é realizado mais um teste, e caso o valor de contribuição informado pelo nó vizinho identificado como gradiente seja menor que o valor de contribuição do próprio nó, então existe máximo local, e o nó entra no estado "Promovendo". Após o tempo aleatório neste estado, se a condição persiste, então o nó é promovido e se torna DN.

Com a adoção desta técnica não identificamos mais a existência de máximos locais

nas simulações realizadas.

3.4.3 Prevenção de *Loops* de Rota

Além do problema de máximo local, existe o problema da criação de *loops* de rota para encaminhamento das mensagens através do gradiente. Na realidade estes *loops* somente se formam de forma transitória, enquanto não há promoção de algum nó ou após deslocamento ou quebra de enlaces. Para diminuir os efeitos destes *loops* provisórios adotamos três técnicas:

- Não retorno de mensagens a origem: um nó ao receber uma mensagem para ser encaminhada pelo gradiente (consulta ou cadastro), busca o vizinho com maior contribuição, excetuando-se aquele do qual recebeu a mensagem, uma vez que de outra forma estaria configurado um *loop* entre estes dois nós. É esta técnica que faz com que o nó 2, na Figura 3.4(b), escolha o nó 6 para encaminhar a mensagem, e não o nó 3, uma vez que ambos possuem o mesmo valor de contribuição.

- Remoção de mensagem em *loop* pela origem: toda vez que um nó recebe uma mensagem que deve ser encaminhada através do gradiente, ele verifica se não é a origem da mesma. Caso seja, a mensagem é descartada por estar em *loop*.

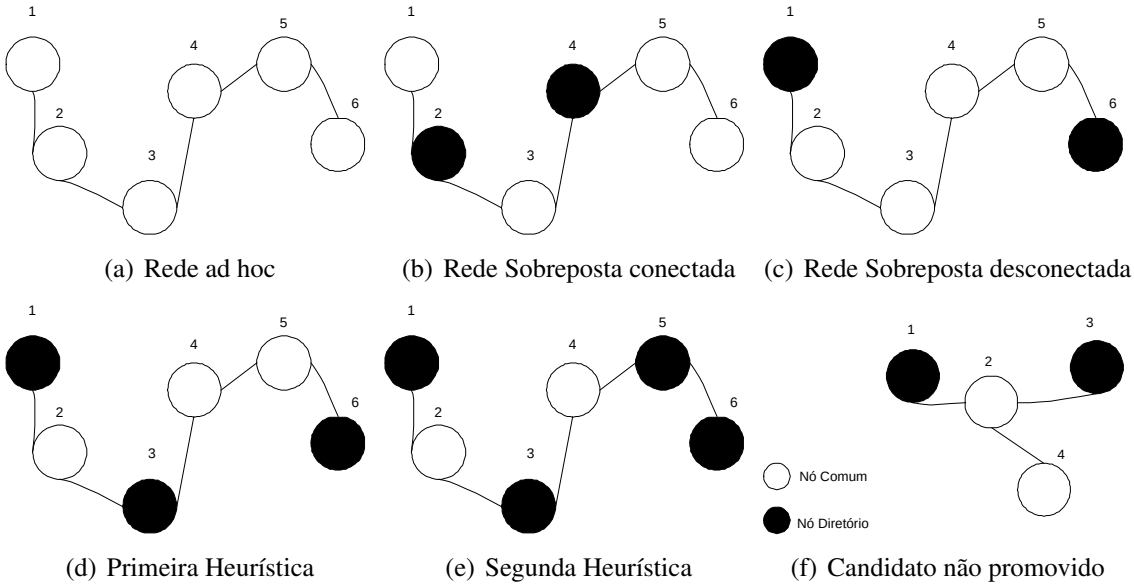
- Uso de ttl para descarte da mensagem: adotamos um ttl nas mensagens encaminhadas pelo gradiente equivalente a 4 vezes o ttl da mensagem de anúncio. O funcionamento da arquitetura prevê que não exista cliente a uma distância do DN mais próximo maior do que o ttl estabelecido na mensagem de anúncio. Entretanto, considerando-se instabilidades e mobilidade, uma mensagem pode ser enviada em uma direção e ter que ser desviada, aumentando o caminho inicial, como apresentado na Figura 3.4. Desta forma adotamos esta relação conservadora nestes $ttls$, de forma a somente descartar as mensagens quando da ocorrência de caminhos consideravelmente longos para a arquitetura.

3.5 Conectividade da Rede Sobreposta

Nossa proposta utiliza as próprias mensagens de anúncio, que criam a informação de gradiente na primeira camada, para obter a conectividade da rede sobreposta da segunda camada. Quando um nó diretório recebe uma mensagem de anúncio de outro nó diretório, eles estabelecem uma vizinhança na rede sobreposta. Entretanto, o processo de promoção e as mensagens de anúncio não são suficientes para garantir que a rede sobreposta não possua nós diretórios isolados ou agrupados em partições separadas.

Para obter a conectividade na segunda camada, adotamos um algoritmo simples, e mais um conjunto de mensagens, de forma a estabelecer esta conectividade.

Para exemplificar este algoritmo, apresentamos um exemplo de rede *ad hoc* simples na Figura 3.5(a). Supondo a utilização de um ttl da mensagem de anúncio igual a 2, podemos ver na Figura 3.5(b), uma configuração onde todos os nós da rede possuem conhecimento da existência de nós diretórios, e a rede sobreposta de nós diretórios está conectada.



(a) Rede ad hoc (b) Rede Sobreposta conectada (c) Rede Sobreposta desconectada

(d) Primeira Heurística (e) Segunda Heurística (f) Candidato não promovido

Figura 3.5: Heurística para Obter Rede Sobreposta Conectada

Entretanto, esta não é a única configuração possível para promoção de alguns nós a nós diretórios. Por exemplo, a configuração na Figura 3.5(c) também permite que todos os nós da rede tenham conhecimento da existência de nós diretórios, mas não permite o estabelecimento de conectividade na rede sobreposta.

A primeira heurística proposta para reduzir este problema é a adoção de um limiar de promoção superior a 0, em especial, um limiar que permita que um nó que esteja a H saltos de um nó diretório entre em processo de promoção. Desta forma, a configuração da Figura 3.5(c) iria provocar que os dois nós centrais entrassem em processo de promoção, produzindo a configuração exibida na Figura 3.5(d).

O que observamos com esta heurística é que a rede sobreposta permanece particionada, entretanto passamos a ter uma nova situação quanto aos nós comuns. Na Figura 3.5(c), os nós comuns só possuíam conhecimento sobre um nó diretório, e portanto nenhum nó possuía capacidade de avaliar se havia a possibilidade da rede sobreposta estar particionada ou não. Na Figura 3.5(d), os nós comuns 2, 4 e 5 possuem visões distintas sobre a rede sobreposta. Vejamos:

- O nó 2 conhece os nós diretórios 1 e 3, e sabe que ambos estão à distância de 1 salto dele, e portanto a uma distância máxima de 2 entre si, portanto estão conectados. Na visão de 2, não há particionamento na rede sobreposta, ou não há particionamento que ele possa ajudar a resolver, logo o nó 2 não é candidato à ponte.

- O nó 4 conhece os nós diretórios 3 e 6, e sabe que o nó 3 está a 1 salto e que 6 está a dois saltos dele. Desta forma podem estar a uma distância máxima de 3 saltos entre si e podem representar um particionamento na rede sobreposta. Então, para solucionar este particionamento, o nó 4 se torna um candidato à ponte entre estas duas possíveis partições.

- O nó 5 é similar ao 4, estando, entretanto, a 2 saltos de 3, e 1 salto de 6. Ele também se torna um candidato à ponte entre as possíveis partições.

Desta forma, os nós 4 e 5 serão candidatos à promoção para permitir a junção das redes sobrepostas, entretanto seguindo mais uma heurística: eles somente serão candidatos caso suas promoções não gerem obrigatoriamente uma instabilidade na rede. Isto é verificado observando o valor de contribuição no momento, e testando se após sua promoção ele não ultrapassou o limiar de despromoção, pois nesse caso, no próximo ciclo dos processos, este nó entraria em processo de despromoção, podendo criar uma oscilação.

Uma vez que 4 e 5 são candidatos, eles montam uma lista de nós diretórios que conhecem, e enviam esta lista na mensagem nó candidato a ponte. Esta mensagem é encaminhada de forma similar à mensagem de consulta, entretanto carrega uma lista de DNs conhecidos. Portanto, esta mensagem segue o gradiente e alcança um nó diretório, que realiza a seqüência de tarefas apresentada no **Algoritmo de Conectividade da Segunda Camada**.

Algoritmo de Conectividade da Segunda Camada

```
1: para cada DN ∈ lista de DNs da mensagem nó candidato a ponte faça
2:     se DN ∈ Tabela de Vizinhos Virtuais faça
3:         Remova DN da lista da mensagem;
4:     caso contrário
5:         Envia mensagem busca DN no anel;
6:     fim se
7: fim para cada
8: se lista está vazia faça
9:     Envia mensagem DN encontrado;
10: fim se
```

Este procedimento de busca é realizado para evitar promoções desnecessárias, que podem aparecer quando um nó é candidato por estar a uma distância grande de 2 ou mais DNs, mas os mesmos estarem próximos ou terem conectividade entre si, como podemos ver no exemplo da Figura 3.5(f). Neste exemplo, o nó 4 é candidato, pois os DNs 1 e 3, que ele conhece, podem estar a uma distância de até 4 saltos entre si. Mas como eles estão na realidade a 2 saltos, recebem mutuamente as mensagens de anúncio e se conhecem na rede sobreposta. Quando o nó 4 enviar a mensagem nó candidato a ponte, esta mensagem será imediatamente respondida, com uma mensagem DN encontrado, pelo DN que a receber, uma vez que ele conhece todos os DNs que constam da lista.

Capítulo 4

Modelo Analítico

Neste capítulo é apresentada uma proposta de modelo analítico para o número de DNs e a sobrecarga de mensagens de controle na construção e manutenção da rede sobreposta de nossa arquitetura.

4.1 Objetivos e Nomenclatura

A principal motivação para a construção de um modelo analítico que descreva o funcionamento da arquitetura é a obtenção de um entendimento mais detalhado do comportamento desta mesma arquitetura frente a modificações nos diversos parâmetros de configuração e possibilidades de cenário a que ela possa ser submetida.

Portanto, apesar de considerarmos que o modelo proposto possui precisão limitada, ele permite alcançar uma avaliação da arquitetura sem perda de generalidade.

O foco deste nosso modelo é estimar as mensagens de controle geradas na rede, inicialmente na primeira camada, posteriormente no estabelecimento da conectividade da segunda camada, e comprovar a escalabilidade de nossa arquitetura em função do número de nós na rede. Além disso, buscamos estimar a influência do ttl da mensagem de anúncio no funcionamento da arquitetura, além de outros parâmetros, comparando estes resultados com aqueles obtidos por simulação, de forma a ajudar na configuração.

Para descrever analiticamente o modelo, que propomos inicialmente em [29], utilizamos as variáveis definidas na Tabela 4.1 que parametrizam a rede.

Tabela 4.1: Notação Adotada para Distribuição de Nós Diretórios

N	Número total de nós
R	Raio da rede
$A = \pi R^2$	Área da rede
$\delta = \frac{N}{A}$	Densidade de nós
r	Raio de alcance de transmissão
H	Ttl da mensagem de anúncio
Nv_H	Número de vizinhos de H saltos
N_{DN}	Número total de nós diretórios
$\rho(\delta, H, r)$	Raio médio da vizinhança de H saltos de um nó
Tx_A	Taxa(msg/s) de envio de anúncios por um DN
Tx_H	Taxa(msg/s) de envio de *hellos*
$MsgAnuncio/(s.N)$	Mensagens de anúncio por nó por segundo na rede
$MsgControle/(s.N)$	Mensagens de controle por nó por segundo na rede

Com o modelo, desejamos poder estimar o número médio de nós diretórios (N_{DN}), que serão promovidos, em função do número de nós (N), do ttl da mensagem de anúncio (H), do alcance de transmissão (r) e da área da rede (A) ou do raio da rede (R).

Também desejamos, através do modelo, estimar o número médio de mensagens de controle por segundo normalizado pelo número de nós da rede ($MsgControle/(s.N)$), em função dos mesmos parâmetros. A vantagem de utilizar esta normalização na contabilização das mensagens de controle é a facilidade de comparação entre simulações com diferentes tempos de execução e número de nós.

Para o desenvolvimento do modelo, utilizamos algumas premissas: a rede possui uma área circular; a rede é conectada, ou seja, não possui partições ou nós isolados; o raio da rede é significativamente maior que o raio do alcance de transmissão dos nós ($R >> r$); e os nós são dispostos de forma aleatória através de uma distribuição uniforme e são estáticos.

Algumas destas premissas são bastante restritivas e limitantes, como por exemplo, não considerar aspectos de mobilidade. Por este motivo, posteriormente iremos relaxar a premissa de área circular e considerar áreas com diferentes formatos, e avaliar, sem determinar um modelo específico, a influência da última premissa, ou seja, considerar uma distribuição dos nós não uniforme. Por último, avaliamos os impactos da mobilidade no comportamento da arquitetura.

4.2 Desenvolvimento do Modelo

Para alcançar os objetivos descritos anteriormente, o desenvolvimento do modelo foi realizado em duas etapas: a primeira, simplificada, considerando todas as premissas e aproximações como válidas; e a segunda, onde são detalhados alguns aspectos práticos de funcionamento da arquitetura, sendo o principal deles o *efeito de borda*.

4.2.1 Modelo Simplificado

Na elaboração do modelo, inicialmente vamos assumir que a distribuição de DNs na rede também é uniforme. Esta premissa inicial leva em conta a distribuição aleatória uniforme para os nós, e também o mecanismo utilizado no processo de promoção, baseado em janelas aleatórias de tempo. Uma vez que assumimos que a distribuição dos dois tipos de nós é uniforme, e que um nó pode ser DN ou nó comum, vamos considerar que a quantidade de DNs em uma amostra de nós da rede pode ser definida através de uma distribuição binomial, expressa por (4.1).

$$P(x = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{4.1}$$

Onde p é a probabilidade de um nó ser um nó diretório, que é igual a $\frac{N_{DN}}{N}$, k é a quantidade de DNs na amostra e n a quantidade de nós na amostra.

Utilizando a quantidade média de nós existentes em uma área de H saltos, Nv_H, como a amostra a ser aplicada a distribuição binomial, e fazendo $k = 0$ em (4.1), temos (4.2), que é a probabilidade de não existir nenhum DN na amostra, neste caso equivalente a uma área de H saltos de vizinhança de um nó.

$$P(x = 0) = (1 - \frac{N_{DN}}{N})^{Nv_H} \tag{4.2}$$

Entretanto, caso um nó não possua um DN vizinho de H saltos, ele deverá entrar no processo de promoção. Supondo que o mecanismo converge, após um tempo inicial, não deveríamos ter mais nós sendo promovidos. Se admitirmos uma nova premissa, de que as promoções de nós são processos aleatórios independentes, podemos expressar o número

médio de nós que não possuem DN vizinho de H saltos como $P(x = 0) \times N$. Então, quando esta expressão for igual a 1, este último nó será promovido e o processo estará estável. Com isto, podemos construir a igualdade apresentada em (4.3).

$$(1 - \frac{N_{DN}}{N})^{Nv_H} \times N = 1 \tag{4.3}$$

Resolvendo esta equação em N_{DN}, obtemos (4.4).

$$N_{DN} = N(1 - (\frac{1}{N})^{\frac{1}{Nv_H}}) \tag{4.4}$$

Com as hipóteses assumidas, este deve ser o número médio de DNs que produz em média somente um nó na rede sem vizinhos de H saltos. Após este último nó se promover, não deverão ocorrer mais promoções, e ocorrerá a estabilização do processo de promoção. Então, podemos considerar a expressão como uma aproximação para o número médio de DNs na rede. Como cada DN envia Tx_A mensagens de anúncio por segundo, o número de mensagens originadas pelos DNs por segundo, será (4.5).

$$MsgEnviadas/s = (N(1 - (\frac{1}{N})^{\frac{1}{Nv_H}}) \times Tx_A \tag{4.5}$$

Entretanto, cada mensagem enviada por um DN é retransmitida por cada vizinho de até $H - 1$ saltos dele, uma vez que nos vizinhos de H saltos, o ttl da mensagem chega a zero e ela não é retransmitida. Então, o número de mensagens de anúncio na rede por segundo será dado pela equação (4.6).

$$MsgAnuncio/s = (N(1 - (\frac{1}{N})^{\frac{1}{Nv_H}}) \times Tx_A \times Nv_{H-1} \tag{4.6}$$

Para calcular Nv_H utilizamos a definição da densidade de nós δ, que é igual a $\frac{N}{A}$, e a definição de $\rho(\delta, H, r)$, que é o raio médio da vizinhança de H saltos de um nó, conforme pode ser observado na Figura 4.1.

Desta forma, podemos calcular o número médio de vizinhos de H saltos, Nv_H, como a densidade de nós na rede, δ, multiplicada pela área média de H saltos, que pode ser calculada como π multiplicado pelo quadrado do raio médio de H saltos, $\rho(\delta, H, r)$, expresso pela equação (4.7).

$$Nv_H = \pi \times \rho(\delta, H, r)^2 \times \delta = \pi \times \rho(\delta, H, r)^2 \times \frac{N}{\pi R^2} \tag{4.7}$$

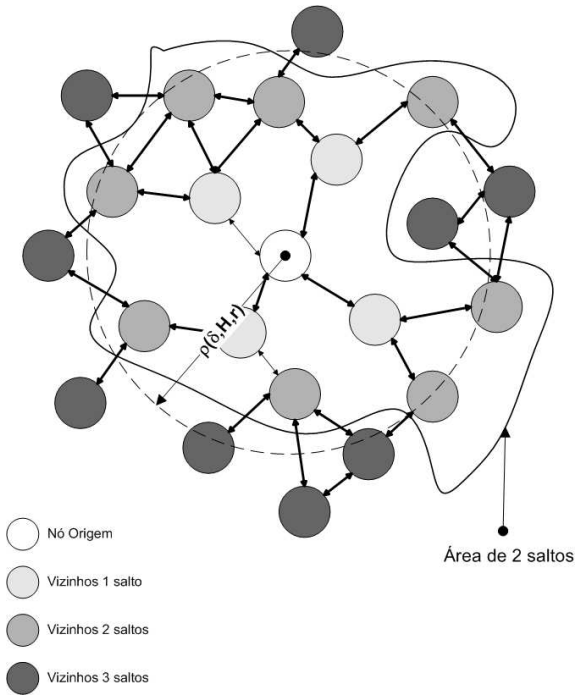


Figura 4.1: Exemplo da Obtenção de $\rho(\delta, H, r)$ para $H = 2$ Saltos

Então, substituindo δ e a equação 4.7 na equação 4.6, obtemos

$$MsgAnuncio/(s.N) = (1 - (\frac{1}{N})^{\frac{1}{\pi \times \rho(\delta,H,r)^2 \times \frac{N}{\pi R^2}}}) \times Tx_A \times \pi \times \rho(\delta, H-1, r)^2 \times \frac{N}{\pi R^2}. \tag{4.8}$$

Uma vez que podemos demonstrar que $N \times (1 - (\frac{1}{N})^{\frac{k}{N}})$ tende a $kLn(N)$ quando N tende a infinito, podemos fazer nova aproximação, e desconsiderar a influência de δ no valor de $\rho(\delta, H, r)$. Desta forma tomamos o raio médio da vizinhança de H saltos, $\rho(\delta, H, r)$, como uma constante em relação a N, e então obtemos $MsgAnuncio/(s.N) = O(log(N))$, expressa para grandes valores de N pela equação (4.9).

$$MsgAnuncio/(s.N) = \frac{\rho(\delta, H-1, r)^2}{\rho(\delta, H, r)^2} \times Ln(N) \times Tx_A \tag{4.9}$$

Adotando a mesma aproximação, o número de DNs é expresso por (4.10).

$$N_{DN} = \frac{R^2}{\rho(\delta, H, r)^2} \times Ln(N) \tag{4.10}$$

Por estas equações, vemos que precisamos estimar o valor de $\rho(\delta, H, r)$. Primeiro, o valor máximo deste raio ocorre quando sempre existem nós no limite do alcance de

transmissão r. Neste caso, que seria equivalente a δ muito elevado, ou N muito grande, teremos (4.11).

$$\rho_{max}(\delta, H, r) = r \times H \tag{4.11}$$

Por outro lado, podemos considerar como extremo inferior para $\rho(\delta, H, r)$ o caso onde os nós são colocados a $\frac{r}{2} + dr$ um do outro. Apesar deste não ser efetivamente o pior caso podemos, sem perda de generalidade, e considerando uma rede com distribuição aleatória e com densidade tal que não ocorra particionamento, considerar este como sendo o limite inferior de $\rho(\delta, H, r)$, e teremos (4.12).

$$\rho_{min}(\delta, H, r) = \frac{r}{2} \times H \tag{4.12}$$

Confirmamos, desta forma, que uma variação de δ de um valor baixo até infinito, produz uma variação limitada em $\rho(\delta, H, r)$, tornando razoável a aproximação realizada para obter (4.9).

Utilizando valor máximo para ρ obtemos (4.13) e (4.14).

$$MsgAnuncio/(s.N) = \frac{(H-1)^2 r^2}{R^2} \times N \times (1 - (\frac{1}{N})^{\frac{R^2}{NH^2r^2}}) \times Tx_A \tag{4.13}$$

E obtemos, para $H > 1$:

$$MsgAnuncio/(s.N) = \frac{(H-1)^2}{H^2} \times Ln(N) \times Tx_A \tag{4.14}$$

Para obter a quantidade de mensagens de controle gerada no primeiro plano, devemos adicionar as mensagens de *hello*, que são geradas na taxa Tx_H por todos os nós. Então, podemos obter este total pela adição de Tx_H à equação 4.13, obtendo (4.15) e o gráfico na Figura 4.2 pode ser construído quando consideramos $Tx_A = Tx_H = 0.2Msg/s$.

$$MsgControle/(s.N) = \frac{(H-1)^2}{H^2} \times Ln(N) \times Tx_A + Tx_H \tag{4.15}$$

A estimativa de N_{DN} é mais dependente dos valores de $\rho(\delta, H, r)$, mas podemos aproximá-la para:

$$N_{DN} = \frac{R^2}{r^2 H^2} \times Ln(N) \tag{4.16}$$

Com esta equação, construímos o gráfico da Figura 4.3, utilizando $r = 200m$ e $R = 846.2844m$ que é o raio equivalente para obter uma área circular com a mesma área que um quadrado de lado $1500m$.

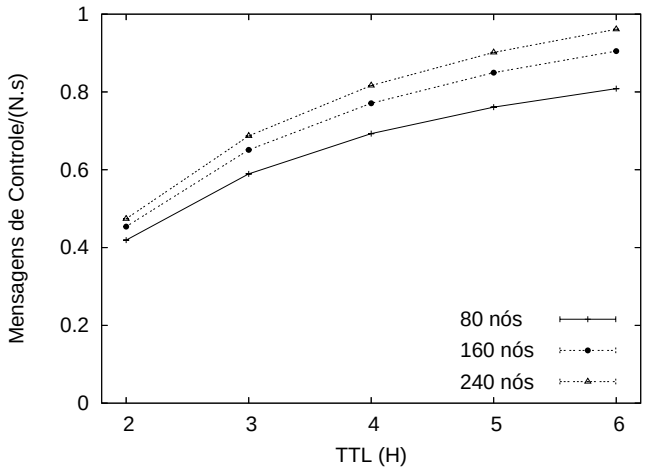


Figura 4.2: Mensagens de Controle pelo Modelo Analítico Simplificado

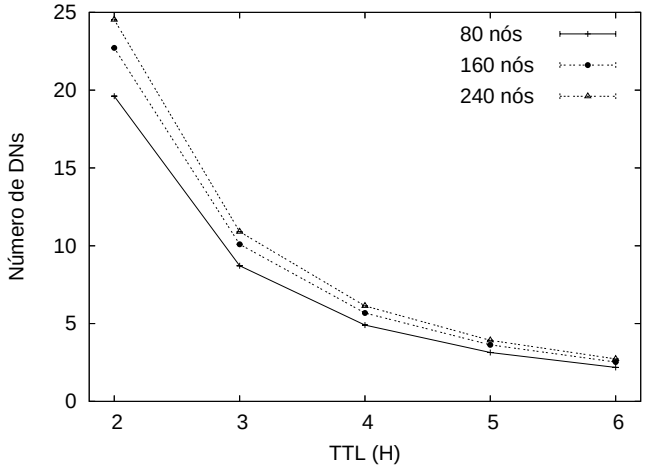


Figura 4.3: Número de DNs pelo Modelo Analítico Simplificado

Para a aplicação das equações 4.13 e 4.16, o produto Hr não pode ser muito maior que R, pois então teríamos uma inundação completa na rede. Além disto o modelo sofre com a existência de efeito de borda, que terá menor influência se o raio da rede for consideravelmente superior ao produto Hr.

4.2.2 Modelo Estendido - Efeito de Borda

Apesar deste modelo simplificado apresentar uma razoável aproximação para o número de mensagens da primeira camada da arquitetura, ele tem aplicação limitada para as demais camadas, pois não fornece uma boa aproximação para o número de DNs. Além disto, ele não modela nem o processo de despromoção e nem a adoção de limiar de promo-

ção não nulo. Outros problemas deste modelo anterior são o efeito de borda e a estimativa da área da vizinhança de H saltos.

Visando obter um modelo mais acurado e uma estimativa realista do número de nós diretórios, adaptamos os resultados do trabalho em [63], que calcula o número médio de vizinhos de um nó. Inicialmente, este trabalho distingue dois tipos de nós, onde os do primeiro tipo são aqueles cuja área de cobertura cai totalmente dentro da região da rede sendo estudada e os do segundo tipo são aqueles próximos da borda da rede, e portanto, somente parte da sua cobertura está dentro da área. Eles são chamados nós do tipo I e II, respectivamente.

Em seguida, demonstra que a distância média entre dois vizinhos, para nós do tipo I, pode ser obtida de (4.17).

$$\bar{r}_I = \frac{2}{3}r \tag{4.17}$$

E o número médio de nós vizinhos, que na notação utilizada pelos autores foi chamado de $\bar{k}$, e em nossa notação é Nv_1, se expressa por (4.18).

$$\bar{k} = \bar{k}_I \times (1 - \frac{r2p}{A}) + \bar{k}_{II} \times \frac{r2p}{A} \tag{4.18}$$

Onde 2p é o perímetro da rede e $\bar{k}_I$ e $\bar{k}_{II}$ são os números médios de vizinhos dos nós do tipo I e do tipo II, respectivamente, expressos por (4.19) e (4.20).

$$\bar{k}_I = (N-1)\frac{\pi r^2}{A} \tag{4.19}$$

$$\bar{k}_{II} = (1 - \frac{2}{3\pi})\bar{k}_I \tag{4.20}$$

Então, adaptamos estes resultados para vizinhos de 1 salto apresentados em [63], ao nosso modelo de H-saltos. Primeiro adaptamos (4.17) e aproximamos o raio médio da vizinhança de H-saltos, $\rho(\delta, H, r)$, usando (4.21).

$$\rho(\delta, H, r) = (1 + (H-1)\frac{2}{3})r \tag{4.21}$$

O conceito envolvido nesta equação é de que o alcance para um salto é equivalente a r, e para os demais saltos será dependente da distância média dos vizinhos, que por (4.17)

é de $\frac{2}{3}$ de r. Esta equação é somente uma aproximação, que não considera os efeitos da densidade δ no valor de ρ.

Então, substituindo (4.19) e (4.20) em (4.18), e r por ρ, podemos calcular Nv_H usando (4.22).

$$Nv_H = \frac{(N-1)\pi\rho^2}{A}[(1 - \frac{\rho \times 2p}{A}) + (1 - \frac{2}{3\pi})\frac{\rho \times 2p}{A}] \tag{4.22}$$

Para adaptar (4.16) para suportar despromoções, nós substituímos o último fator N de (4.3) pelos nós que efetivamente podem ser promovidos, ou seja, o número de nós na rede (N), menos o número de nós já promovidos (N_{DN}), e também menos o número de vizinhos destes nós que seriam despromovidos pelo Limiar de Despromoção. Esta nova equação é (4.23), onde D é o limiar de despromoção expresso em saltos.

$$(1 - \frac{N_{DN}}{N})^{Nv_H} \times (N - N_{DN}(1 + Nv_D)) = 1 \tag{4.23}$$

Para se obter D expresso em saltos, devemos conhecer os valores que os DNs utilizam para sua própria contribuição, ou quando $h = 0$, também chamada de Contribuição Infinita. Também são necessários o valor do limiar de despromoção e a expressão da equação de contribuição $Contrib_n$. Por exemplo, se a equação para contribuição é $Contrib_n = \frac{1}{h}$, então $Limiar\ de\ Despromoção - Contribuição\ Infinita = \frac{1}{D}$, ou, de forma genérica (4.24).

$$D = {Contrib_n}^{-1}(Limiar\ de\ Despromoção - Contribuição\ Infinita) \tag{4.24}$$

Portanto, sendo o Limiar de Despromoção igual a 101 e a Contribuição Infinita igual a 100, D será igual a 1.

Aplicando (4.22) em (4.23), podemos calcular N_{DN} usando métodos numéricos e construir os dois gráficos nas Figuras 4.4(a) e 4.4(b). Eles mostram o número de DNs e mensagens de controle por nó por segundo, usando os parâmetros descritos na tabela 4.2.

Utilizando cenários com estes mesmos parâmetros, executamos simulações no *ns-2* [61] com nossa arquitetura, sem a adoção do algoritmo de conectividade da segunda camada, uma vez que nosso objetivo é avaliar o número de DNs obtido somente pelos processos de promoção e despromoção. Com estas execuções, obtivemos os resultados

Tabela 4.2: Parâmetros Utilizados

N	80, 160 e 240
R	846.2844m
$A = \pi R^2$	2250000m^2
$2p$	6000m (quadrado de lado 1500m)
r	200m
H	2, 3, 4, 5 e 6
Tx_A	0.2 msg/s
Tx_H	0.2 msg/s, mas com supressão de *hellos*
D	1 salto

apresentados nas Figuras 4.5(a) e 4.5(b), as quais, comparadas com as Figuras 4.2 e 4.3 do modelo simplificado e com as Figuras 4.4(a) e 4.4(b) do modelo estendido, nos mostram melhoria na precisão das estimativas com a adoção do último modelo.

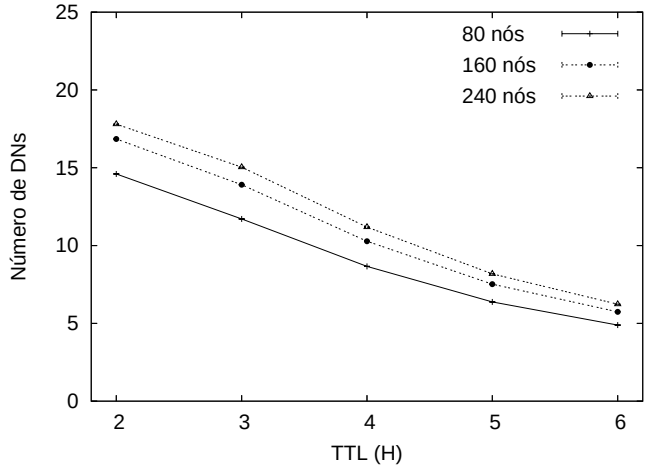
Entretanto, através da comparação desses resultados apresentados na tabela 4.3, que inclui margem de erro relativa calculada usando a equação (4.25), observamos que o modelo ainda não apresenta excelente acurácia, mas apresenta a mesma tendência e ordem de grandeza dos resultados obtidos por simulação.

$$\%erro = \frac{N_{DN}\ simulação - N_{DN}\ modelo}{N_{DN}\ simulação} \times 100\% \tag{4.25}$$

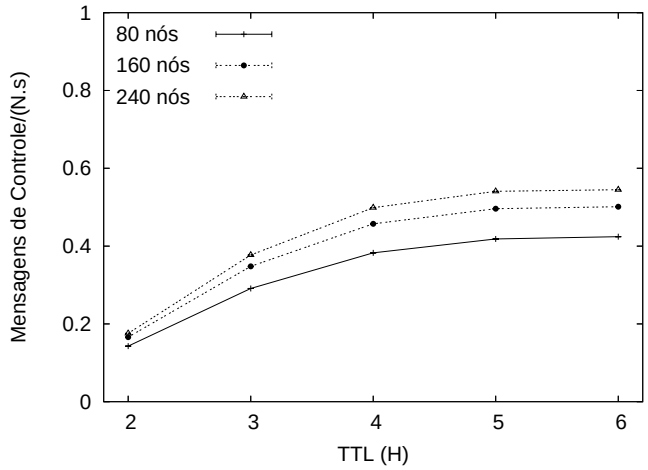
Para o número de DNs, um dos fatores que gera esta imprecisão é a estimativa utilizada para $\rho(\delta, H, r)$, que não considera a influência da densidade. Intuitivamente, podemos perceber que redes com menores densidades apresentam menor ρ e portanto maior número de DNs, o que faz com que o erro apresentado seja maior para menores densidades e maiores valores de H, o que está de acordo com os resultados obtidos.

Por outro lado, uma justificativa para a divergência obtida no número de mensagens de controle no modelo estendido com o efeito de borda, é a diferença entre a estimativa do número de vizinhos de $H - 1$ saltos, obtida através do modelo, e o mesmo número de vizinhos obtido por simulação. Através da simulação, podemos obter este número de vizinhos calculando a razão entre o número de mensagens de anúncio encaminhadas e o número de mensagens de anúncio enviadas.

Por exemplo, para $ttl = 4$, temos os valores apresentados na tabela 4.4, tanto para



(a) Número de DNs



(b) Mensagens de Controle

Figura 4.4: Modelo Analítico Estendido

simulação quanto para o modelo analítico estendido. A tabela mostra que o número médio de vizinhos de 3 saltos obtidos por simulação é significativamente menor que o sugerido pelo modelo analítico.

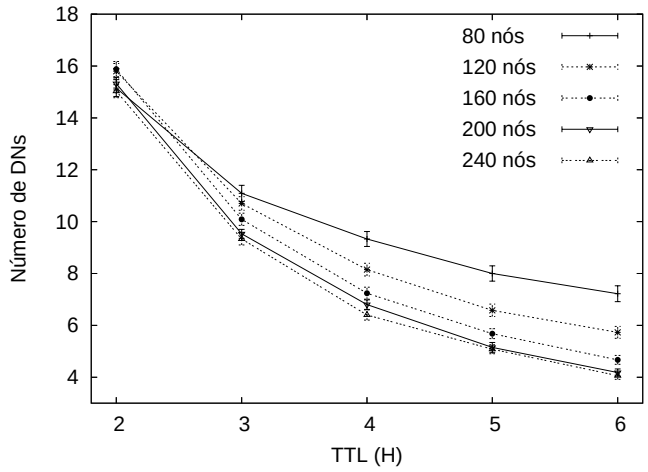
Esta diferença também é devida ao efeito de borda, entretanto afetando outra premissa de nosso modelo: a de que a distribuição de DNs na rede é uniforme. Esta premissa fundamentou-se no fato de que o processo de promoção baseia-se na escolha de tempos aleatórios, de forma que numa rede simples, como por exemplo a rede com 4 nós totalmente conectados, apresentada na Figura 4.6(a), todos possuem a mesma probabilidade de se promoverem a DN, ou seja, cada um possui probabilidade de promoção de 25%. Entretanto, em outras redes, como a da Figura 4.6(b), o efeito de borda implica em que os

Tabela 4.3: N_{DN} por Simulação e pelos Modelos Analíticos Simplificado(1) e Estendido(2)

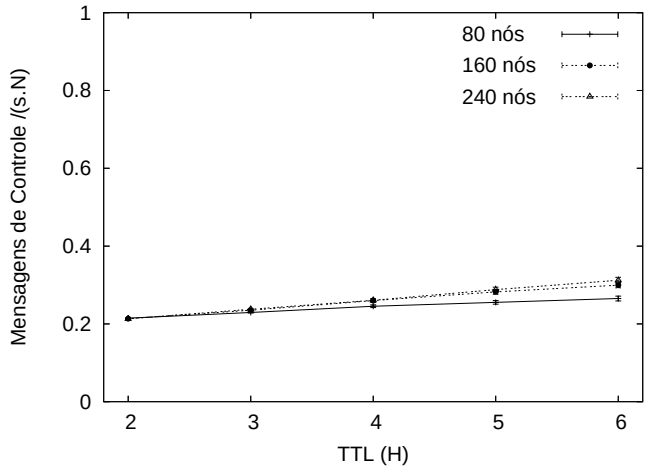
Nós	H	Simulação	Modelo 1	% erro	Modelo 2	% erro
80	2	15,15	19,61	-29,47	14,6	3,63
	3	11,09	8,72	21,39	11,71	-5,59
	4	9,33	4,9	47,44	8,66	7,18
	5	8,00	3,14	60,77	6,38	20,25
	6	7,22	2,18	69,81	4,89	32,27
160	2	15,88	22,72	-43,06	16,85	-6,11
	3	10,09	10,1	-0,07	13,91	-37,86
	4	7,24	5,68	21,56	10,28	-41,99
	5	5,68	3,63	36,01	7,52	-32,39
	6	4,67	2,52	45,95	5,74	-22,91
240	2	15,03	24,53	-63,22	17,8	-18,43
	3	9,33	10,9	-16,86	15,04	-61,2
	4	6,4	6,13	4,17	11,19	-74,84
	5	5,08	3,93	22,73	8,18	-61,02
	6	4,06	2,73	32,86	6,23	-53,45

Tabela 4.4: Nv_3 Obtidos por Simulação e pelo Modelo Analítico Estendido

Nós	Anúncios Enviados ($AEnv$)	Anúncios Encaminhados ($AEnc$)	Nv_3 simulação ($\frac{AEnc}{AEnv}$)	Nv_3 analítico
80	916,44	7617,69	8,312	17,68
160	707,28	17621,06	24,914	35,58
240	622,72	26945,62	43,271	53,48



(a) Número de DNs



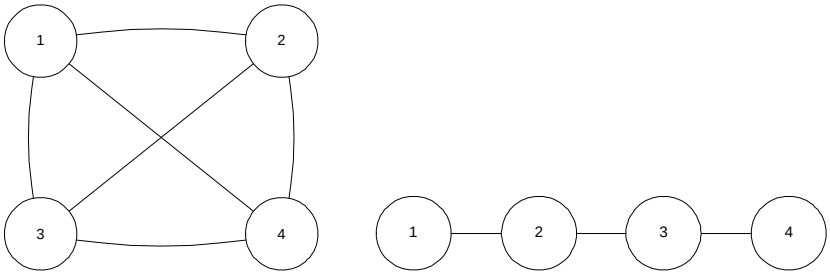
(b) Mensagens de Controle

Figura 4.5: Simulação

nós próximos a borda tenham uma maior probabilidade de se promoverem, e portanto os DNs possuem um número menor de vizinhos que o estimado.

Para explicar isto, vamos considerar o exemplo da Figura 4.6(b) com a arquitetura funcionando com $ttl = 2$, e também considerar limiar de promoção nulo e que uma vez que um nó se promova, todos os seus vizinhos de 2 saltos tenham conhecimento imediato disto, e portanto não se promovam.

Com estas considerações, o primeiro nó a se promover será aquele que escolher a menor janela. Como todos os nós são iguais, a probabilidade de que um nó seja o primeiro a se promover, que chamaremos de P_1, é a mesma para todos, e então podemos escrever a equação (4.26).



(a) Grafo totalmente conectado com 4 nós

(b) Grafo em linha com 4 nós

Figura 4.6: Efeito de Borda na Promoção

$$P_1(1) = P_1(2) = P_1(3) = P_1(4) = 0,25 \tag{4.26}$$

Entretanto, existe a probabilidade de um segundo nó se promover, P_2, e ela é condicionada pelo evento relativo a P_1, conforme equações (4.27).

$$\begin{aligned}
&P_2(2|1\ promovido) = P_2(3|1\ promovido) = 0,\ P_2(4|1\ promovido) = 1 \\
&P_2(1|2\ promovido) = P_2(3|2\ promovido) = P_2(4|2\ promovido) = 0 \\
&P_2(1|3\ promovido) = P_2(3|3\ promovido) = P_2(4|3\ promovido) = 0 \\
&P_2(2|4\ promovido) = P_2(3|4\ promovido) = 0,\ P_2(1|4\ promovido) = 1
\end{aligned} \tag{4.27}$$

Ou seja, haverá maior probabilidade de que nós com menor número de vizinhos sejam promovidos.

Para comprovar isto utilizamos a mesma simulação, e procuramos obter a distribuição de DNs na área. Para isto, a área da rede, um quadrado de $1500m \times 1500m$, foi dividida em *quadrados concêntricos*, obtida pela união de quadrados com lado de $100m$ cada que possuem igual distância da borda da rede, como na Figura 4.7. Para cada área concêntrica, numerada da borda para o centro de 0 a 7, foi medido o número médio de DNs para cada $10000m^2$, ou quadrados de $100m \times 100m$. Estes valores são apresentados na Figura 4.8 e nela podemos constatar que a distribuição de DNs depende da distância para a borda, que chamamos de d.

Com estes gráficos, podemos constatar que a premissa de que a distribuição de DNs é uniforme não é válida. Outra premissa igualmente não válida é a de que os processos de promoção são independentes.

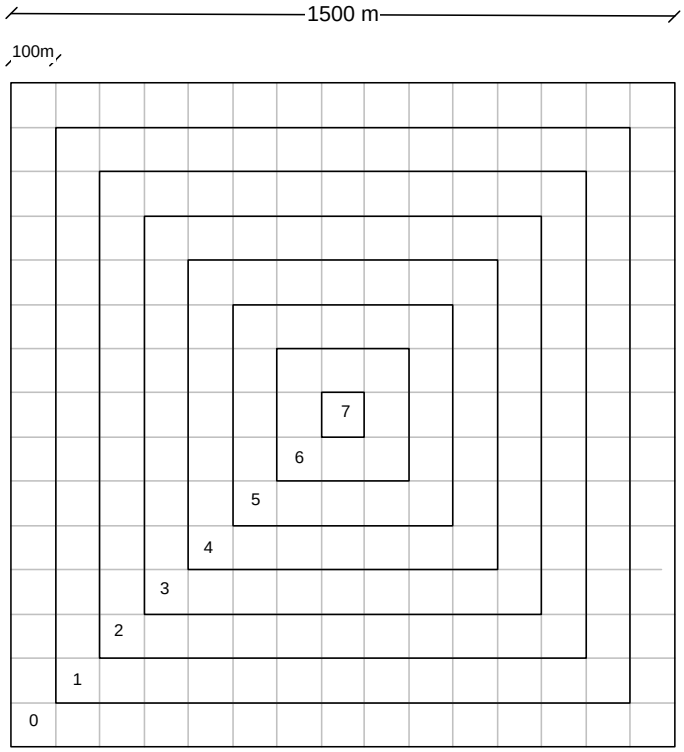


Figura 4.7: Quadrados Concêntricos para Medição da Densidade de DNs

Tabela 4.5: N_{DN} Obtidos por Simulação e pelo Modelo Estendido - $ttl = 4$ e área de $3000m \times 750m$

Nós	Simulação	Modelo
80	7,72	6,91
160	7,70	8,17
240	7,01	8,88

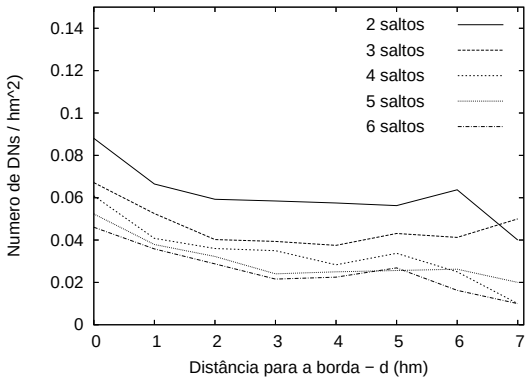
Entretanto, o modelo estendido apresenta um número de mensagens de controle superior ao obtido por simulação, e portanto pode ser adotado como uma estimativa conservadora.

Outra vantagem do modelo estendido é a de apresentar resultados para redes com diversas formas, além da circular, somente dependendo da relação entre perímetro e área. Na Tabela 4.5 apresentamos o número de DNs estimado e simulado com ttl da mensagem de anúncio igual a 4 e quando alteramos a rede para o formato de $3000m \times 750m$, ou seja, alterando o perímetro sem modificar a área. Estes valores são bem distintos daqueles apresentados pelo modelo simplificado, que não permite capturar a influência do formato da área.

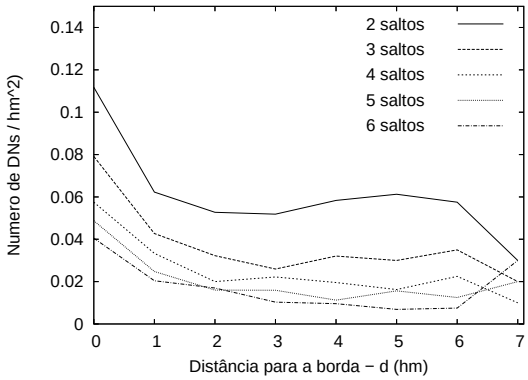
Finalmente, apresentamos algumas considerações sobre o algoritmo de conectividade da segunda camada e a mobilidade.

Caso a heurística adotada para o algoritmo seja bastante eficiente, devemos esperar que o número final de DNs, com a ativação deste processo, seja no máximo duas vezes o obtido nestes modelos. Esta afirmação se explica pelo fato de que dois DNs que estejam particionados necessitarão no máximo de que um nó seja promovido para se conectarem, portanto com N_{DN} particionados, necessitaríamos de um máximo de $N_{DN} - 1$ promoções para obter uma rede conectada.

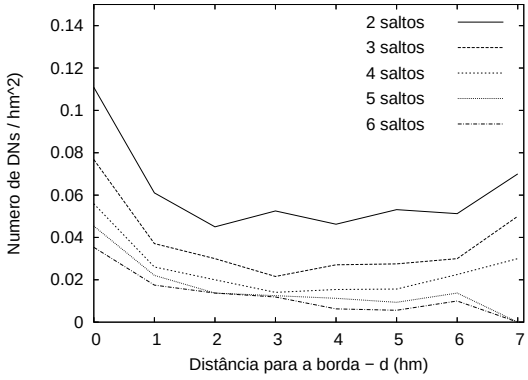
Ainda sobre o algoritmo de conectividade e também sobre mobilidade, uma importante observação é que ambos podem ajudar na revalidação da premissa de distribuição uniforme dos nós diretórios na rede. Primeiro o algoritmo de conectividade, por efetuar promoções nas regiões com menor número de nós diretórios, segundo a mobilidade, por redistribuir os nós na rede, inclusive os nós diretórios. Estes fatores explicam porque ambos modelos podem ser aplicados, considerando-se as imprecisões já observadas, também em redes com mobilidade.



(a) 80 nós



(b) 160 nós



(c) 240 nós

Figura 4.8: Densidade de DNs em Função da Distância à Borda da Rede

Capítulo 5

Avaliação de Desempenho

Este capítulo descreve a implementação da arquitetura em ambiente de simulação, assim como os resultados obtidos. Para isto ele está dividido em quatro seções: a primeira relativa aos detalhes da implementação; a segunda à geração de cenários utilizados para avaliação; a terceira sobre as métricas utilizadas; e a quarta e última sobre os resultados obtidos.

5.1 Implementação

Para avaliar a proposta, utilizamos o simulador *ns-2* [61], versão 2.29. Foi realizada a implementação da arquitetura como um agente do *ns-2*, similar aos agentes de roteamento, associado ao nó móvel conforme descrito na documentação do *ns-2* [64] e representado na Figura 5.1, extraída da referida documentação.

Um fator importante nesta implementação foi permitir que a arquitetura enviasse mensagens diretamente através da camada de enlace, para ser roteada por mecanismo próprio, como no caso das mensagens enviadas através do gradiente, e que também enviasse mensagens através do protocolo de roteamento, como no caso de mensagens em *unicast*. Desta forma o agente criado, representado na Figura 5.2 como SDA, pode enviar mensagens tanto ao nó (*Node entry*), como para a camada de enlace (*Link*).

Durante a implementação de nossa arquitetura, verificamos problemas no funcionamento dos protocolos AODV e DSDV no *ns-2*, que não realizavam corretamente o en-

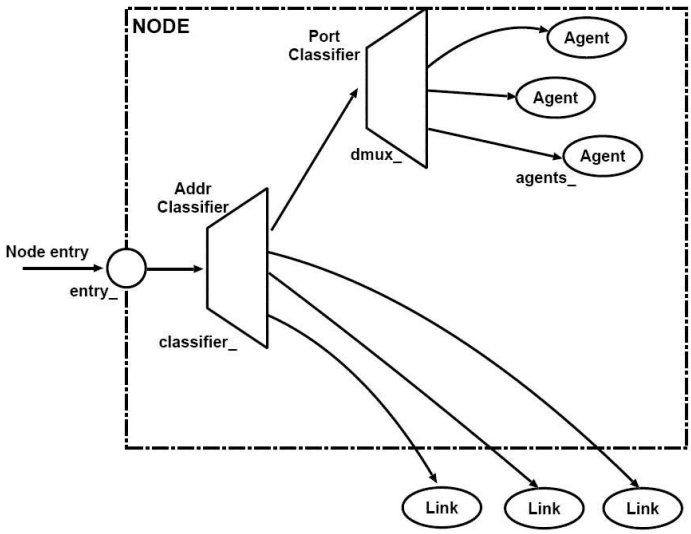


Figura 5.1: Diagrama do nó obtido da documentação do ns-2

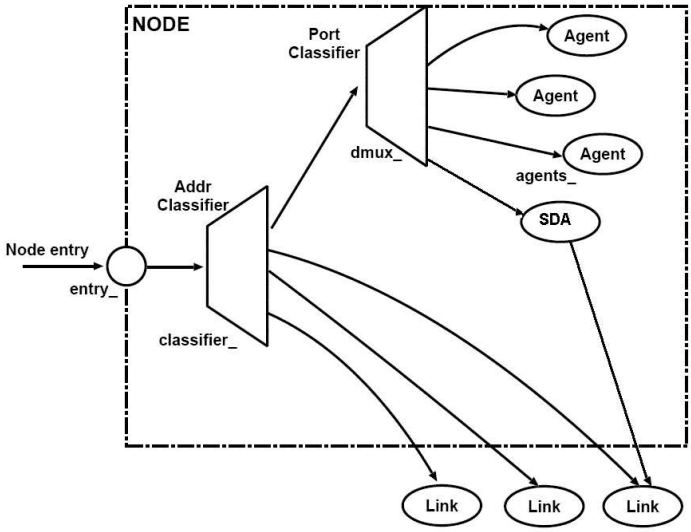


Figura 5.2: Diagrama do nó com agente para descoberta de serviço (SDA)

caminhamento de mensagens em *broadcast* por vários saltos, tais como a mensagem de anúncio de nossa arquitetura. O problema foi corrigido no código do AODV, e este protocolo de roteamento foi escolhido para ser usado na rede *ad hoc*.

Os pacotes recebidos por um nó são encaminhados ao respectivo agente através do *dmux_*, que utiliza a informação de porta para esta finalidade. Os protocolos de roteamento utilizam porta de número 255, então, para que nosso agente funcione em conjunto com estes protocolos, ele deve utilizar uma porta diferente, portanto, foi adotado o número 253.

Uma vez ativado o agente da arquitetura, cada nó pode ser configurado para funcionar em 4 modos: modo 0, estático, ou seja, processo de promoção e despromoção desabilitados; modo 1, promoção; modo 2, despromoção; e o modo 3, dinâmico, onde atuam simultaneamente os processo de promoção e despromoção. Destes modos, o que corresponde ao funcionamento normal da arquitetura é o modo 3, dinâmico.

No modo 0, estático, o nó permanecerá no estado nó comum ou nó diretório, que lhe foi atribuído no início da simulação, durante toda a execução. O modo 1, promoção, permite que nós inicialmente configurados como comuns promovam-se a nós diretórios. Já o modo 2, despromoção, atua de forma inversa ao modo 1, permitindo que nós inicialmente configurados como diretórios despromovam-se a nós comuns.

A finalidade destes modos diferentes do modo 3 é poder avaliar cada processo, de promoção e de despromoção, de forma independente, e avaliar seus comportamentos em termos de convergência.

Outra opção de implementação que fizemos, para agilizar o desenvolvimento da arquitetura, foi não desenvolver a terceira camada, uma vez que os principais mecanismos que desejamos verificar são os das duas primeiras camadas. Assim, as mensagens da terceira camada não foram criadas, e para substituí-las, adotamos a inundação na segunda camada.

Desta forma, quando um nó provedor envia sua mensagem de cadastro e ela alcança um nó diretório, a mensagem sobreposta de cadastro é difundida em toda rede sobreposta, permitindo que todos os DNs tenham o serviço cadastrado. Esta solução permite avaliar se a rede sobreposta criada é realmente conectada, e gera uma sobrecarga da ordem do número de arestas da rede sobreposta, ou no pior caso, de $O({N_{DN}}^2)$. Estas arestas na rede sobreposta correspondem na rede real em caminhos de múltiplos saltos, entretanto estes caminhos tem no máximo H saltos, e portanto o número de mensagens geradas na rede será $H \times O({N_{DN}}^2)$, que é também $O({N_{DN}}^2)$.

Como o cadastro se propaga para todos os DNs, a mensagem sobreposta de consulta também não foi utilizada, pois se o DN não possui o serviço cadastrado, espera-se que os demais nós da rede sobreposta também não tenham.

Por último, a não implementação da terceira camada implicou em alteração no algoritmo de conectividade da segunda camada. Os passos implementados são os seguintes:

- passo 1 – verifica a lista de DN na mensagem. Para cada DN que ele já tenha conhecimento, remove da lista da mensagem.
- passo 2 – se após a remoção dos DNs conhecidos, a lista se tornou vazia, ele envia a mensagem DNs encontrados ao candidato e encerra o procedimento. Se a lista não ficou vazia, ou seja, ainda existem DNs não conhecidos, segue ao passo 3.
- passo 3 – como a lista não está vazia, enviar, em *unicast*, a lista restante a todos os DNs que conhece.

Este terceiro passo deveria ser uma busca pelos elementos da lista no anel da terceira camada, enviando a mensagem busca por DN. Entretanto, como não fizemos esta implementação, é realizada uma inundação na segunda camada, e então o DN envia a lista a todos os DNs que conhece. Por exemplo, na Figura 5.3, que reproduz a Figura 3.5(d) onde o alcance da mensagem de anúncio é de 2 saltos, o DN 3 receberia uma lista inicial do nó 4 com a lista dos possíveis DNs particionados 3 e 6. O DN 3 removeria seu próprio endereço, e encaminharia a lista com o endereço 6 para o DN 1. Como o DN 1 também não conhece o DN 6, o nó 4 não recebe a mensagem DNs encontrados e se promove, permitindo a conectividade da rede sobreposta.

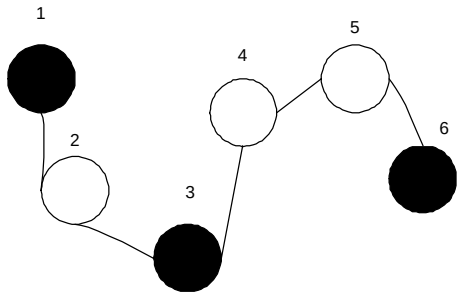


Figura 5.3: Exemplo de execução do algoritmo de conectividade da segunda camada

Um DN ao receber a lista vinda de outro DN, realiza o mesmo procedimento, removendo os endereços desconhecidos e encaminhando a lista aos demais DNs vizinhos. Esta inundação na rede sobreposta, evidentemente, é controlada, utilizando identificação das

mensagens e fazendo com que a mesma seja encaminhada somente na primeira vez que é recebida. Após este procedimento, uma vez que o nó candidato, após uma janela de promoção, não tenha recebido a mensagem de DNs encontrados, ele se promove, permitindo assim a conectividade entre as partições.

Para configuração da arquitetura também foram criadas variáveis, de modo a permitir flexibilidade. Estas variáveis, com uma explicação sucinta e valores padrões, configurados no arquivo *ns-default.tcl* do *ns-2*, são apresentados na Tabela 5.1.

Estes valores indicados na tabela foram os utilizados nas diversas simulações, a menos daqueles explicitamente modificados, e informados a cada caso.

Os valores de *jitter* foram adotados para propiciar maior aleatoriedade aos mecanismos e impedir a sincronização dos nós, e são expressos em variação percentual em relação ao valor médio. Por exemplo, os *hellos* possuem *jitter* para reduzir a probabilidade que sincronizem e colidam. No caso do *hello*, os valores padrões correspondem a intervalos de 5 segundos e *jitter* de 50 %, ou seja, mais ou menos 25 %, de forma que o intervalo entre dois *hellos* consecutivos pode ir de 3,75 segundos até 6,25 segundos.

As chaves para ativar/desativar os *hellos* disparados e verificação de conectividade da segunda camada foram criadas para testar e avaliar o funcionamento dos respectivos mecanismos, permitindo que eles sejam desligados ou religados facilmente.

Os valores de 5 segundos para intervalo entre *hellos* e anúncios são iguais aos propostos em [17] e também utilizados em nossos trabalhos em [29] e [30]. Os tempos escolhidos para as janelas de promoção e de despromoção, de 9 segundos, basearam-se na idéia de que o processo de promoção deve atuar mais lentamente que os anúncios. Mesmo conceito foi utilizado para adoção do tempo de 19 segundos para verificação de conectividade de segunda camada, que deve ocorrer em função das promoções já ocorridas através do processo de promoção.

Intervalos maiores podem ser adotados em redes que apresentem baixa mobilidade, de forma que a sobrecarga de mensagens seja menor. Entretanto, é importante observar que tal medida, além da redução de robustez em relação a mobilidade, também provoca um aumento no tempo de estabilização ou convergência da rede.

Tabela 5.1: Parâmetros da Arquitetura e Valores Padrões

Nome da Variável	Descrição	valor
announc_ttl_	TTL da mensagem de anúncio	4
DN_	Estado inicial do nó. true=DN, false=Comum	false
HelloInterval_	Tempo médio entre Hellos (s)	5
HelloJitterPerc_	Variação aleatória imposta entre Hellos (%/100)	0.5
AnnouncInterval_	Tempo médio entre Anúncios (s)	5
AnnouncJitterPerc_	Variação aleatória imposta entre Anúncios (%/100)	0.5
HelloExpirationFactor_	Número de *Hellos* perdidos para remoção do vizinhos	2
ContribExpirationFactor_	Número de Anúncios perdidos para remoção do DN	2
InfiniteContrib_	Valor correspondente a Contribuição infinita	100
PromotionInterval_	Tempo médio entre mudanças de estado no processo de promoção (s)	9
PromotionThreshold_	Limiar de promoção	0
PromotionJitterPerc_	Variação aleatória imposta no tempo do processo de promoção (%100)	0.5
DemotionInterval_	Tempo médio entre mudanças de estado no processo de despromoção (s)	9
DemotionThreshold_	Limiar de despromoção	101
DemotionJitterPerc_	Variação aleatória imposta no tempo do processo de despromoção (%/100)	0.5
mode_	Modo de funcionamento do nó (modo 0, 1, 2 ou 3)	0
TriggeredHello_	Hellos disparados ativados (true ou false)	true
SubscribeInterval_	Tempo médio entre Cadastros (s)	20
SubscribeJitterPerc_	Variação aleatória imposta entre Cadastros (%/100)	0.2
SubscribeExpirationFactor_	Número de Cadastros perdidos para remoção do provedor	3
Connect2ndLevelInterval_	Tempo médio entre verificações de conectividade da segunda camada (s)	19
Connect2ndLevelJitterPerc_	Variação aleatória imposta na verificação de conectividade (%/100)	0.2
Connect2ndLevelSwitch_	Verificação de conectividade da segunda camada ativada (true ou false)	true
DNSearchTimeout_	Tempo de espera pela resposta DNs Encontrados pelo nó candidato (s)	3

Para as mensagens de cadastro, adotamos um intervalo de 20 segundos entre mensagens, para que não resultasse em um número grande de mensagens, uma vez que foi utilizada a inundação na segunda camada. Além disso, este valor pode ser controlado pela própria aplicação que disponibiliza o serviço, por exemplo, a mensagem de cadastro pode carregar uma informação de tempo de duração do serviço, e utilizá-lo como valor periódico de renovação do cadastro.

5.2 Cenários

Para avaliar nossa implementação, criamos alguns cenários baseados em disposição aleatória dos nós, entretanto procurando controlar alguns parâmetros dos mesmos.

Inicialmente, utilizamos a ferramenta *setdest* do *ns-2* e geramos cenários aleatórios, onde os nós são distribuídos uniformemente em uma área de 1500 por 1500 metros, variando o número de nós de 80 a 240, em intervalos de 20 nós. Como não há sentido em avaliar a conectividade da rede sobreposta se esta conectividade já não existe na rede *ad hoc*, a primeira camada, selecionamos somente os cenários onde a rede gerada era conectada, considerando alcance de transmissão constante de 200 metros. Foram armazenados 100 cenários para cada quantidade de nós, totalizando 800 cenários, com esta configuração.

Posteriormente, através de programa próprio, geramos outros cenários, com a mesma área e mesmo número de nós, entretanto limitando a conectividade, ou seja, o grau máximo ou número máximo de vizinhos, de cada nó. Este programa insere um nó na área, e em seguida vai inserindo aleatoriamente outros nós, mas sempre garantindo que cada novo nó tenha pelo menos um outro nó à distância de 200 metros, ou seja, não está isolado, e que também esteja a mais do que uma distância mínima de 25 metros de cada um dos outros. Além destas restrições de distância, o grau máximo de cada nó é limitado, tendo sido utilizado o valor igual a 4, para rede com 80 nós, e incrementando este grau de 1 para cada mais 20 nós na rede. Desta forma, foram então armazenados outros 800 cenários.

Além disto, geramos, com a ferramenta *setdest*, cenários com 80, 160 e 240 nós,

na mesma área de 1500 por 1500 metros, com mobilidade do tipo *Random Waypoint* e velocidades aleatórias variando uniformemente até 0, 3, 5, 10 e $20m/s$. Para cada par $(nó, velocidade\ máxima)$, foram gerados 100 cenários. Não foi utilizada pausa entre movimentações, uma vez que o objetivo destes cenários é avaliar a robustez da arquitetura frente a situações de intensa mobilidade, justificando desta forma a utilização de velocidades tão grandes quanto $20m/s$ em uma área limitada de menos de pouco mais de $2km^2$. Neste caso foram gerados mais 1500 cenários.

Cada simulação foi então executada, para cada configuração, em 100 rodadas, uma para cada cenário específico. Portanto, os gráficos gerados possuem barras de erro, em torno das médias das amostras, correspondentes a 100 amostras, utilizando intervalo de confiança de 95%.

5.3 Métricas

Duas métricas principais de desempenho foram utilizadas para avaliação da arquitetura: taxa de sucesso de descoberta; e sobrecarga de mensagens de controle.

Estas métricas principais se dividem em outras da forma a seguir. A taxa de sucesso de descoberta divide-se em:

- taxa efetiva de descoberta de serviços: calculada como a razão entre o número de mensagens de resposta a consultas recebidas e o número de mensagens de consulta enviadas pelos clientes.

- taxa de consultas que chegam aos DNs: calculada como a fração entre o número de mensagens de consulta que alcançam um DN e o número de mensagens de consulta geradas pelos clientes. Esta foi a métrica utilizada em nossos trabalhos prévios em [29] e [30]. Ela mede a eficiência do encaminhamento das mensagens através do gradiente até alcançar a rede sobreposta.

- taxa de consultas respondidas pelos DNs: calculada como a fração entre o número de mensagens de consulta respondidas pelos DNs e o número de mensagens de controle geradas pelos clientes. Este métrica corresponde ao funcionamento do

mecanismo de cadastro e de conectividade da segunda camada. Os valores obtidos nesta métrica devem ser menores ou iguais ao da taxa de consultas que chegam aos DNs e maiores ou iguais a taxa efetiva. Se esta métrica apresentar valor muito próximo ou igual ao da chegada de consultas aos DNs, teremos uma indicação de bom funcionamento do cadastro e da conectividade. Por outro lado, a diferença entre esta taxa e a taxa efetiva indica que o protocolo de roteamento não está sendo eficiente na entrega das mensagens de resposta que são enviadas em *unicast*.

Já a sobrecarga de mensagens de controle foi considerada como o número de mensagens de controle enviadas e encaminhadas por segundo, normalizado pelo número de nós. Esta normalização foi adotada para se comparar a quantidade de mensagens geradas em redes com tamanhos e números de nós diferentes ou tempos de simulação distintos, além disto, este valor significa o número médio de mensagens que cada nó envia por unidade de tempo para o funcionamento da arquitetura. Esta métrica divide-se em:

- sobrecarga da primeira camada: considerando somente as mensagens de *hello* e mensagens de anúncio enviadas e encaminhadas. É a mesma métrica apresentada em nossos trabalhos iniciais em [29] e [30].

- sobrecarga da segunda camada: computando todas as mensagens geradas ou encaminhadas para o funcionamento das duas primeiras camadas da arquitetura, ou seja, *hellos*, anúncios, cadastro, candidato à ponte e DN encontrado. As mensagens de cadastro na segunda camada e busca por DN não foram computadas, uma vez que na nossa implementação elas são inundadas na segunda camada, portanto geram um número de mensagens bem superior ao necessário na implementação completa. Também não incluímos as mensagens de consulta, pois estas, na prática, devem ser geradas de acordo com a necessidade dos clientes, e nas nossas simulações foram geradas periodicamente de forma intensiva.

- sobrecarga total: computando todas as mensagens da arquitetura, incluindo consultas e cadastro na segunda camada e busca por DN.

- sobrecarga detalhada: nesta métrica, o número de mensagens não é normalizado,

sendo apresentado o total ao final da simulação, discriminado por cada tipo de mensagem.

Além destas métricas, avaliamos alguns valores indicativos do funcionamento da arquitetura. Foi medida a convergência da arquitetura, apresentando a evolução do número de DNs na rede ao longo do tempo de simulação. Também medimos o número de promoções por tipo de algoritmo utilizado na promoção, ou seja: pelo processo de promoção, apresentado na Seção 3.3; para conectividade da segunda camada, apresentado na Seção 3.5; ou para eliminação de máximo local, apresentado na Seção 3.4.2.

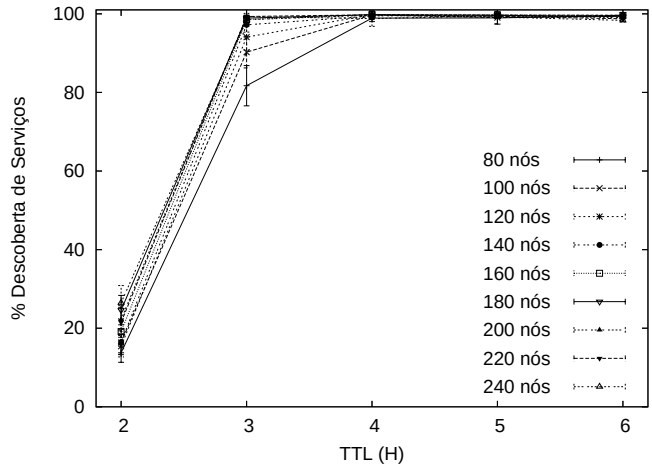
5.4 Resultados Obtidos

Em nossas simulações, foram utilizados 10 clientes na rede, fazendo consultas periódicas em intervalos fixos de 3 segundos, todos buscando o mesmo serviço, o único cadastrado na rede por um único provedor. É importante observar que como o processo de promoção é aleatório, alguns clientes podem se tornar DNs e o próprio provedor também pode ser um DN, o que é totalmente natural e transparente para a arquitetura.

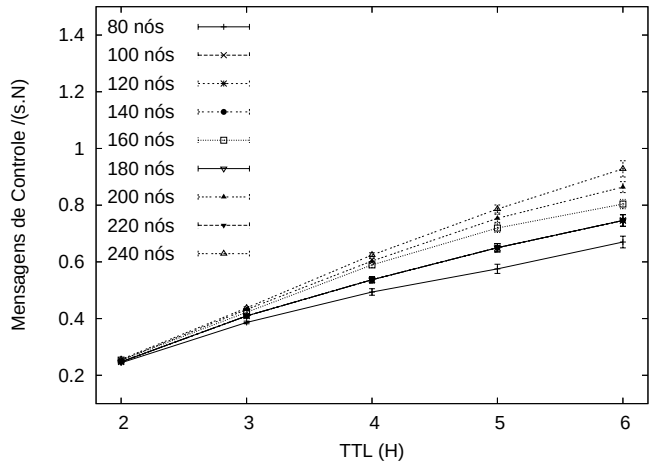
No primeiro conjunto de simulações objetivamos obter qual o valor de ttl da mensagem de anúncio é mais adequado para o funcionamento da arquitetura. Para isto, adotando os parâmetros da Tabela 5.2, e utilizando o primeiro conjunto de cenários, variamos o ttl dos anúncios de 2 a 6. Desta forma podemos avaliar também o algoritmo de conectividade da segunda camada, ao comparar os resultados com aqueles obtidos no Capítulo 4, e os resultados são apresentados nas Figuras 5.4(a) e 5.4(b).

As Figuras 5.4(a) e 5.4(b) mostram taxa de descoberta de serviços efetiva e sobrecarga de mensagem de controle da primeira camada, respectivamente, em função do ttl utilizado. Podemos ver que com os valores de limiares de promoção e despromoção utilizados, a taxa de descoberta apresenta melhores resultados com ttl entre 4 e 6.

No caso da adoção de ttl da mensagem de anúncio baixo, a eficiência foi reduzida pela dificuldade no estabelecimento da conectividade da segunda camada, uma vez que o uso de limiar de despromoção igual a 101, que corresponde a no máximo 1 DN vizinho



(a) % de Descoberta de Serviços



(b) Mensagens de controle

Figura 5.4: Simulação com primeiro conjunto de cenários

de 1 salto, limita a promoção de DNs muito próximos. A adoção de outros limiares de despromoção poderia aumentar a eficiência de descoberta mesmo com $ttls$ baixos, mas adicionando o custo de um maior número de DNs na segunda camada.

Como menores $ttls$ provocam um menor número de mensagens de controle, adotamos o valor de ttl das mensagens de anúncio igual a 4 nas demais verificações de funcionamento da arquitetura e avaliações de desempenho. Por exemplo, a Figura 5.5 mostra o tempo de convergência de nossa arquitetura, onde podemos ver o número médio de DNs ao longo do tempo de simulação quando considerado o ttl igual a 4, e podemos notar que em menos de 100 segundos há um número estável de DNs na rede.

Esta estabilidade no número de DNs é importante para o bom funcionamento da ar-

Tabela 5.2: Valores Usados nas Simulações

Alcance de transmissão	$200m$	Área da rede	$1500 \times 1500m^2$
Início dos cadastros	$50s$	Início das consultas	$70s$
Perdas de *hellos* até expiração	4	*Jitter* da conectividade	0.5
Tempo de Simulação	$500s$	Taxa de dados	$11Mbps$

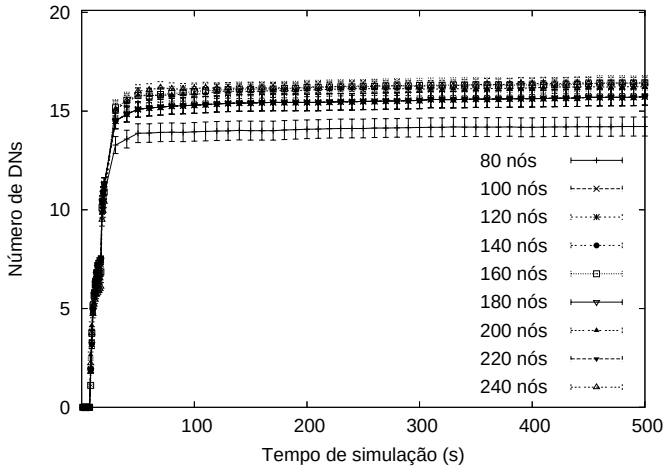


Figura 5.5: Convergência do número de DNs - TTL = 4

quitetura, pois quando um DN é despromovido e outro promovido em seu lugar, ou deve ser realizada uma transferência da tabela de serviços entre eles, ou os serviços precisam se cadastrar novamente, criando um intervalo de inconsistência durante o qual consultas não serão respondidas. Pela Figura 5.5 podemos concluir que os valores adotados permitem que esta estabilidade seja alcançada.

Na Figura 5.6, vemos o número de promoções ocorridas para cada um dos 3 tipos de algoritmo: pelo processo de promoção por limiar; para evitar máximo local; e para obter conectividade da segunda camada. Estes números indicam o número bruto final de promoções por cada tipo, portanto a soma deles pode ser maior que o número final de DNs, uma vez que além das promoções, ocorrem despromoções.

Conforme esperado e citado na Seção 3.4.2, a ocorrência de promoções por existência de máximo local é quase nula. Entretanto esta técnica é importante, pois nos raros casos onde ocorra um máximo local, a arquitetura sofreria danos graves em seu funcionamento, uma vez que este máximo local atuaria como um ”buraco negro”, atraindo as consultas e os cadastros e reduzindo a taxa de descoberta. Como a escala do gráfico não permite

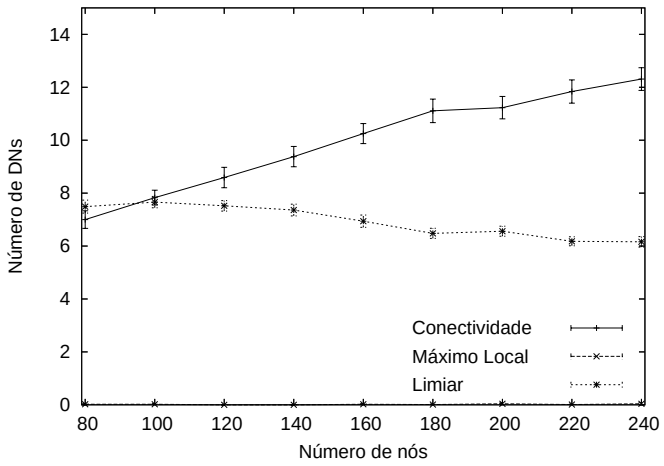


Figura 5.6: Tipos de promoção

Tabela 5.3: Numero Médio de Promoções por Máximo Local - $ttl = 4$

Número de nós	média de promoções
80	0.02000000
100	0.02000000
120	0.00000000
140	0.00000000
160	0.02000000
180	0.01000000
200	0.04000000
220	0.01000000
240	0.02000000

uma visualização detalhada deste tipo de promoção, por ser bem menor que os outros dois tipos, também a apresentamos na Tabela 5.3.

Por outro lado, diferente do sugerido ao final do Capítulo 4, o número de promoções pelo algoritmo de conectividade superou o número de promoções por limiar, para redes com mais de 100 nós. Um dos motivos é que ambos os processos atuam simultaneamente, e apesar de mais lentas, as promoções obtidas pelo algoritmo de conectividade acabam por inibir as promoções por limiar. Outro motivo, é que o algoritmo de conectividade de segunda camada permite, em alguns casos, que ocorram duas ou mais promoções quando somente uma seria suficiente para a obtenção da conectividade. Esta situação pode ser entendida através da Figura 5.7, onde bastaria a promoção do nó A ou do nó B, mas ambos podem ser promovidos para conectar as duas partições da rede, uma vez que os nós A e B

estão a uma distância superior a H saltos entre si, e a mensagem de nó candidato a ponte de ambos pode ser enviada e tratada antes da promoção de cada um.

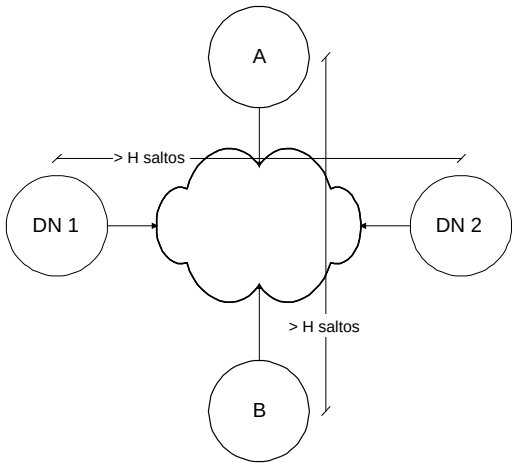


Figura 5.7: Dupla promoção pelo algoritmo de conectividade da segunda camada

O funcionamento do mecanismo de supressão de *hellos*, apresentado na Seção 3.4.1, pode ser observado na Figura 5.8. Sem este mecanismo, cada nó enviaria uma mensagem de *hello* a cada período configurado, no caso de $5s$. Desta forma, a taxa de envio de mensagens de *hello* por nó, por segundo, seria de $0.2\ mensagens hellos/(s.N)$. Conforme podemos observar, com o mecanismo esta taxa é inferior a 0.02, e para ttl do anúncio igual a 4, esta taxa se torna inferior a 0.01 para todas as redes, de 80 a 240 nós, usadas nas simulações.

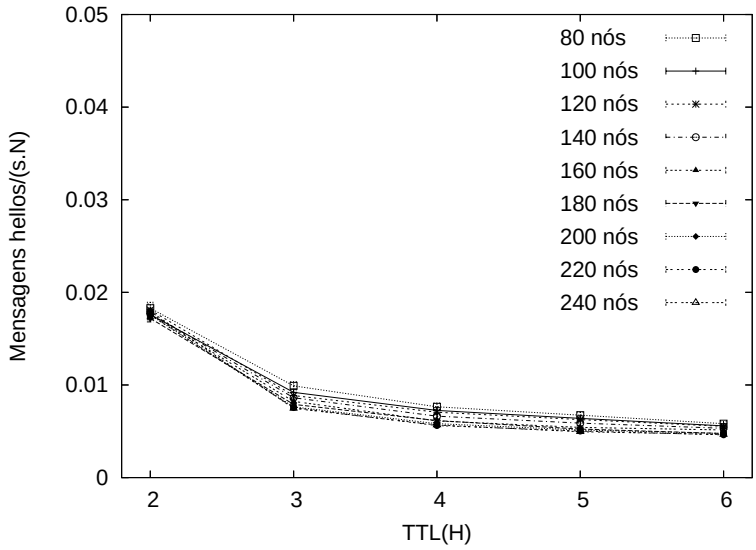


Figura 5.8: Taxa de *hellos* com a supressão ativada

Na Figura 5.9, podemos ver as 3 taxas referentes à descoberta de serviço quando o ttl das mensagens de anúncio é igual a 4. Primeiro, a chegada de consultas a DNs, mostrando a grande eficiência da primeira camada, com taxas muito próximas a 100%. Vemos também a taxa de consultas respondidas, com valores também bem próximos a 100%, o que nos mostra que o processo de cadastramento do serviço e a distribuição na segunda camada são eficientes, ou seja, o algoritmo de conectividade da segunda camada é efetivo. Por último vemos a taxa efetiva de descoberta. Esta última métrica incorpora a utilização das mensagens de respostas a consultas, enviadas em *unicast* pelos DNs aos nós clientes.

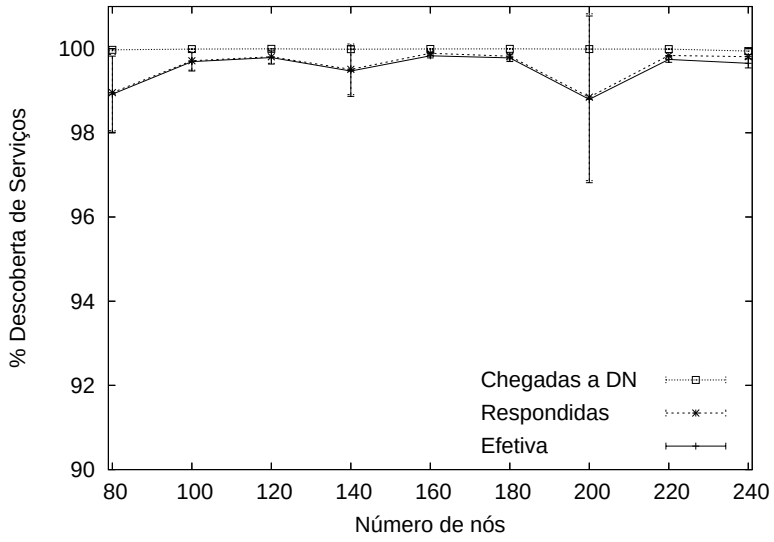


Figura 5.9: Taxas de descoberta de serviço (%)

Estas mensagens por serem em *unicast* utilizam o protocolo de roteamento da rede *ad hoc*, tendo sido adotado o AODV. Uma diferença entre os resultados desta métrica e as demais indicaria que apesar da arquitetura funcionar bem, o protocolo de roteamento não consegue encaminhar as respostas com eficiência. Entretanto, na Figura 5.9 observamos que a taxa efetiva é quase a mesma que a taxa de consultas respondidas, só podendo ser observada na rede com 240 nós uma pequena diferença entre ambas. Esta diferença pode ser um indicativo de alguma dificuldade da implementação do AODV existente no *ns-2* para fazer a entrega de mensagens em redes com esta elevada densidade e grande número de nós, entretanto não de forma conclusiva, devido à diferença ser muito pequena.

Outro fator relevante observável na Figura 5.9 é uma alteração, tanto na diminuição

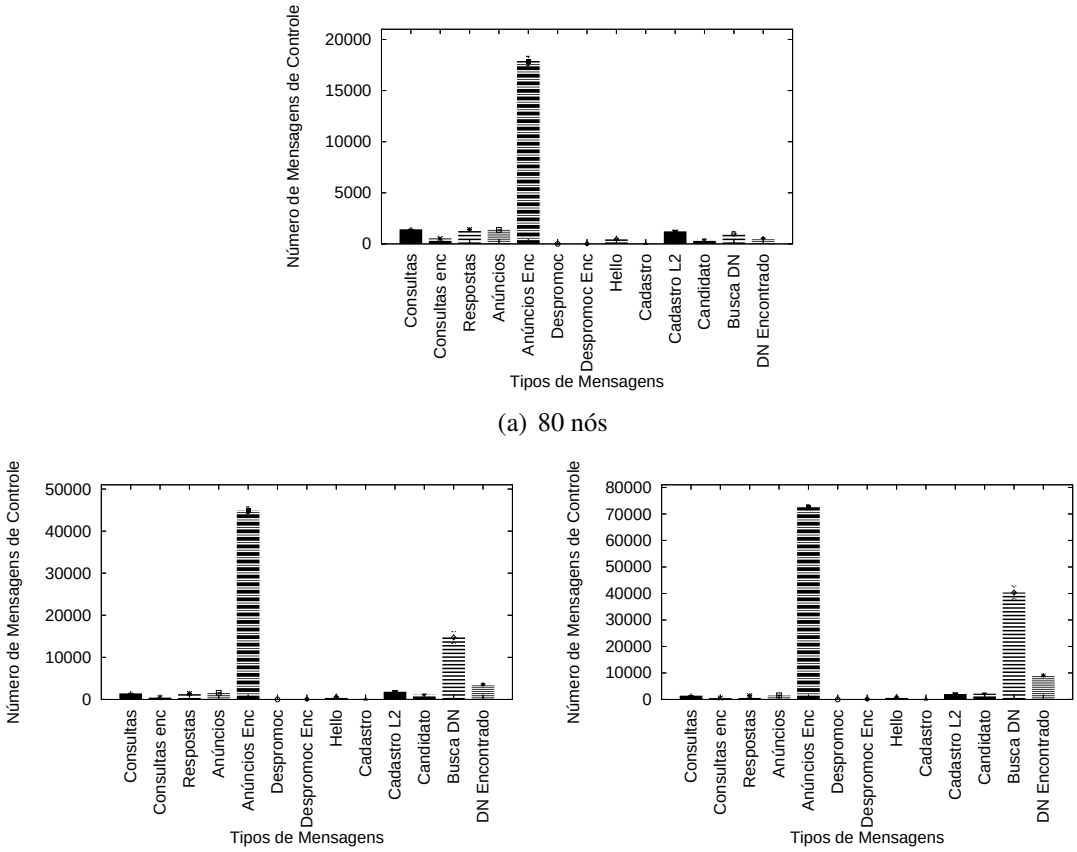
da média quanto no aumento do intervalo de confiança, nos resultados para 80, 140, e principalmente para 200 nós. Avaliando o resultado de cada execução para 200 nós, é possível concluir qual o problema apresentado. Nestas 100 execuções, em 29 delas foi obtida eficiência total de descoberta, com 100% de respostas. Em 68 execuções, o percentual de descoberta foi superior ou igual a 99%, e diferente de 100%. Em duas execuções, o percentual de descoberta foi superior ou igual a 98% e inferior a 99%. Finalmente, em uma execução, o percentual de descoberta foi de 0%.

Avaliando esta execução específica onde não ocorreram descobertas, concluímos que o algoritmo de conectividade da segunda camada não conseguiu conectar duas partições da rede sobreposta, e que, coincidentemente, todos os clientes ficaram em partição diferente daquela do provedor, impedindo que a descoberta fosse realizada.

Este resultado indica que o algoritmo de conectividade, apesar de eficiente na maioria dos casos, ainda deve ser mais estudado, para uma melhor escolha dos parâmetros adotados. Por exemplo, conforme apresentado na Seção 3.5, a escolha de limiares de promoção maiores, conforme indicado pela primeira heurística do algoritmo de conectividade da segunda camada, pode melhorar esta eficiência, evidentemente ao custo de um aumento do número de anúncios encaminhados.

As Figuras 5.10(a), 5.10(b) e 5.10(c) mostram o número de mensagens de controle, com $ttl = 4$ saltos, de forma qualitativa, para redes com 80, 160 e 240 nós, respectivamente. Nelas, podemos observar que as mensagens com maior quantidade correspondem aos anúncios encaminhados e as mensagens de busca por DN. Estas últimas podem ser consideravelmente reduzidas através da implementação da terceira camada com pesquisa estruturada conforme proposto, uma vez que a inundação na segunda camada provoca um número de mensagens em $O({N_{DN}}^2)$, enquanto uma busca em um anel simples gera $O(N_{DN})$ e uma busca com Symphony gera $O(logN_{DN})$.

Estas figuras também mostram que melhorias na arquitetura proposta, com o objetivo de aumentar a escalabilidade, devem se concentrar na questão do encaminhamento dos anúncios, e na estrutura de busca na rede sobreposta, para reduzir tanto o número de mensagens do tipo busca por DN, como também reduzir as mensagens do tipo cadastro na segunda camada (Cadastro L2), pois apesar desta última mensagem não apresentar



(a) 80 nós

(b) 160 nós

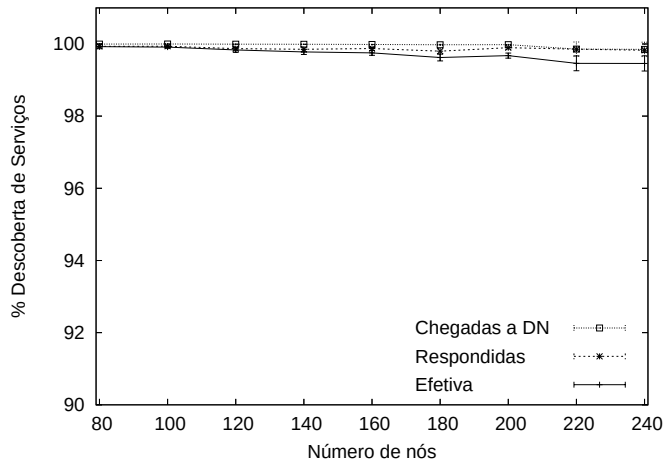
(c) 240 nós

Figura 5.10: Quantidade de mensagens por tipo

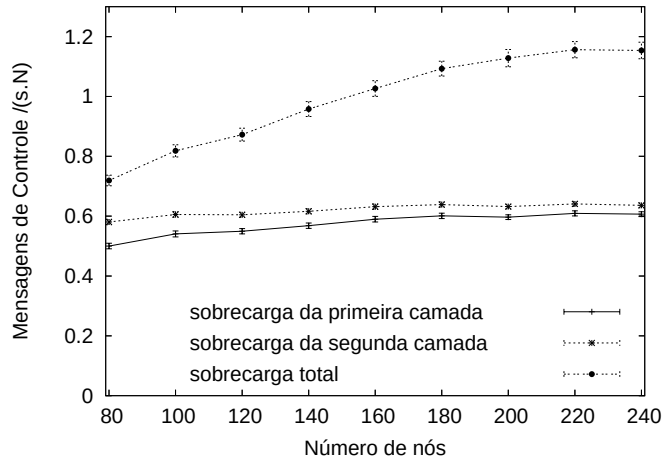
uma quantidade elevada nestas figuras, é importante observar que o resultado obtido foi proveniente do cadastro de um único serviço, e teoricamente ele deve crescer linearmente com o aumento do número de serviços.

Em seguida, realizamos simulações com o segundo conjunto de cenários, também estático, mas com grau de conectividade controlado. Neste caso, desejamos verificar a diferença nos resultados quando não há mais uma distribuição uniforme dos nós, mas sim uma rede onde a conectividade é mais uniforme.

As Figuras 5.11(a) e 5.11(b) mostram as taxas de descoberta e as mensagens de controle com este conjunto de cenários, respectivamente, com ttl da mensagem de anúncio igual a 4. Podemos constatar que os resultados são bastante similares aos obtidos com o primeiro conjunto de cenários, apresentados nas Figuras 5.4(a) e 5.4(b), indicando que a arquitetura proposta se adapta a outras distribuições de nós na rede além da aleatória uniforme.



(a) Taxa de descoberta de serviço %



(b) Mensagens de Controle

Figura 5.11: Simulação com conectividade controlada

Nestas simulações medimos também os tipos de descarte de mensagens de consulta pela primeira camada. Novamente, como os valores são muito reduzidos, apresentamos estes valores não em gráfico, mas na Tabela 5.4. Através dela, pode-se constatar que o descarte de consultas é extremamente reduzido.

Em seguida, foram realizadas simulações usando o terceiro conjunto de cenários e ttl igual a 4. Nestes cenários, onde os nós podem apresentar mobilidade, a rede pode iniciar particionada, e isto implica em uma redução na taxa de descoberta de serviços na rede sem mobilidade, ou seja, onde a velocidade é de até 0 m/s. Portanto, as redes com menor número de nós, e conseqüentemente menor conectividade, apresentam taxas de descobertas reduzidas, quando os nós são estáticos.

Tabela 5.4: Numero Médio de Descartes de Consulta por tipo - $ttl = 4$

Número de nós	loop	ttl	falta de vizinho
80	0.00000000	0.00000000	0.00000000
100	0.00000000	0.00000000	0.00000000
120	0.01000000	0.00000000	0.00000000
140	0.00000000	0.00000000	0.00000000
160	0.01000000	0.00000000	0.00000000
180	0.00000000	0.04000000	0.00000000
200	0.00000000	0.00000000	0.00000000
220	0.01000000	0.10000000	0.00000000
240	0.01000000	0.01000000	0.00000000

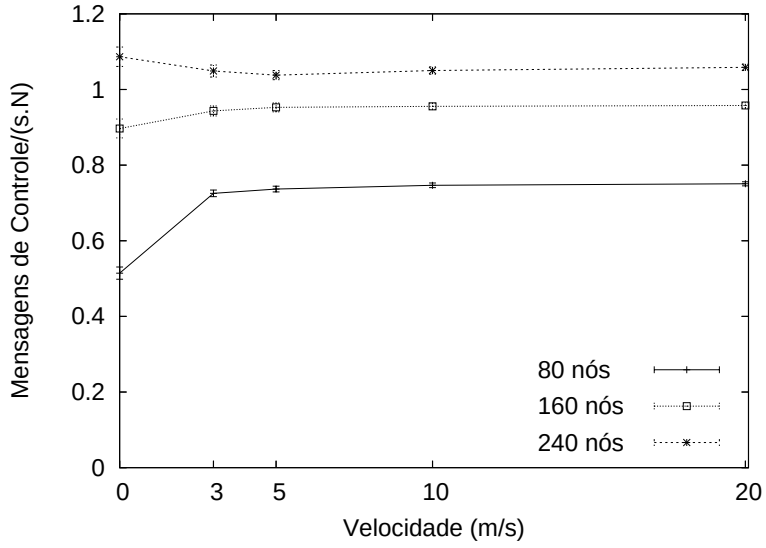


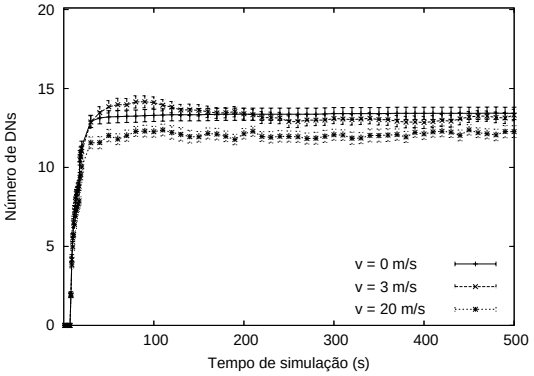
Figura 5.12: Mensagens de Controle com mobilidade

Este resultado não é uma deficiência da arquitetura, mas somente conseqüência do particionamento da rede, o que causa que somente os clientes que estejam na mesma partição do serviço consigam alcançá-lo. Adotamos esta abordagem de não garantir a conectividade neste tipo de cenário porque a mobilidade impediria a avaliação do tipo de particionamento da rede a cada instante. Desta forma, podemos comparar os resultados com mobilidade, que podem particionar a rede, com aqueles obtidos em redes estáticas também particionadas.

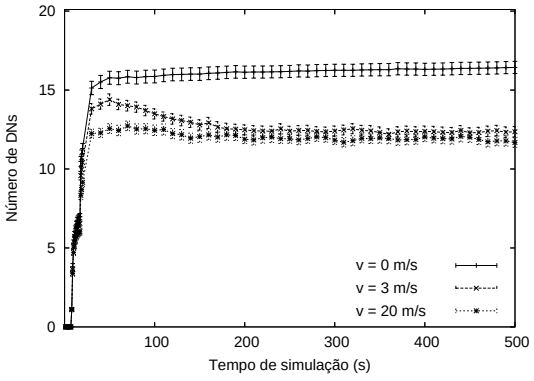
Podemos ver na Figura 5.12 que ocorre somente um pequeno aumento na sobrecarga de mensagens de controle em comparação com os cenários sem mobilidade. Este aumento é ocasionado pelas mudanças de vizinhança, que provocam novas promoções de DNs.

Entretanto, o comportamento geral é similar, demonstrando a efetividade da arquitetura.

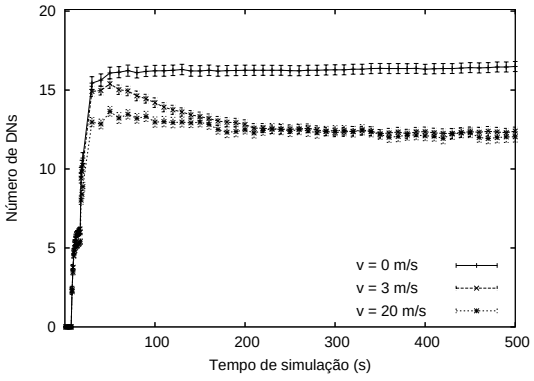
Ainda na Figura 5.12 podemos notar que há até um decréscimo nas mensagens de controle para a rede com 240 nós e mobilidade. Esta redução ocorre devido a redução do número de DNs com a mobilidade, pois tal mobilidade proporciona uma distribuição mais uniforme dos DNs, conforme citamos ao final do Capítulo 4. Podemos observar esta redução na Figura 5.13, que apresenta o número de DNs ao longo do tempo para as redes com 80, 160 e 240 nós, com velocidades de até 0, 3 e 20 m/s. Nestas figuras também podemos notar que o número de DNs ao longo do tempo é estável mesmo com grande mobilidade.



(a) 80 nós



(b) 160 nós



(c) 240 nós

Figura 5.13: Número de DNs Durante a Simulação

Para estes cenários com mobilidade também foi medida a taxa de descoberta de serviço, apresentada na Figura 5.14. Este resultado mostra uma grande robustez frente à mobilidade, uma vez que há pouca mudança nos resultados entre as diversas velocidades. Em relação ao cenário estático, observa-se até um aumento da taxa de descoberta, expli-

cado pelo efeito da disseminação, provocado por um DN que se desloque de uma partição para outra, carregando as informações de cadastro previamente obtidas, e também pela união de partições que pode ser provocada pela mobilidade.

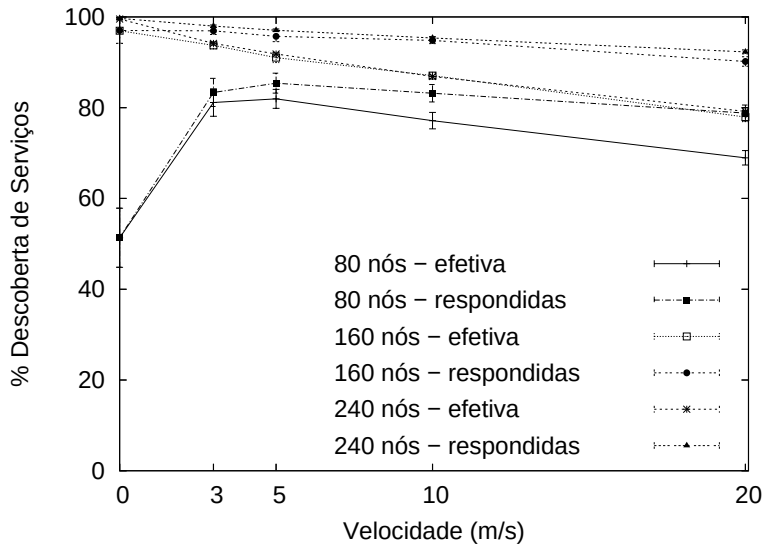


Figura 5.14: Taxa de descoberta de serviço % com mobilidade

Capítulo 6

Conclusões

6.1 Contribuições

Esta dissertação propõe uma arquitetura para descoberta de serviços em redes *ad hoc* baseada no uso de nós diretórios. Ela se divide em três camadas, sendo que a primeira possui a tarefa de encaminhar as consultas e os cadastros até os nós diretórios, a segunda possui a tarefa de criar uma rede sobreposta conectada com os nós diretórios, e a terceira possui a tarefa de criar uma estrutura de distribuição dos cadastros e encaminhamento das consultas na rede sobreposta.

Nossa implementação da arquitetura utiliza o conceito de campo, similar ao proposto em [17] para criar um mecanismo de roteamento *anycast* para mensagens de consulta e cadastro de serviços até estes nós diretórios. Os mecanismos de anúncio, e de campo, dos nós diretórios, são utilizados para realizar a seleção dinâmica destes mesmos nós diretórios. Para obter a conectividade da rede sobreposta, utilizamos uma heurística, realizando a promoção de outros nós de forma a obter a conectividade. Na terceira camada, nossa proposta é utilizar uma simplificação do Symphony [14], entretanto nossa implementação utilizou inundação na rede sobreposta, para reduzir o tempo de implementação da arquitetura.

A contribuição principal desta dissertação é a apresentação da arquitetura em si, indicando detalhadamente seu comportamento e forma de funcionamento. Dentro dos detalhes da arquitetura, uma das principais contribuições é a proposta de um mecanismo

de promoção e de despromoção dos nós a nó diretório, de forma a cobrir toda a rede, utilizando somente decisão local, o que é muito desejável em redes *ad hoc*.

Outra importante contribuição é a heurística para conectividade de segunda camada, que permite o estabelecimento de uma rede sobreposta conectada, somente através do uso das informações da primeira camada da arquitetura, e um pequeno conjunto de mensagens.

Também apresentamos um modelo analítico que permite estimar número de mensagens de controle e de nós diretórios na arquitetura, indicando boa escalabilidade da mesma. Apesar do modelo não apresentar excelente acurácia, é possível através dele estimar o funcionamento da arquitetura em diferentes cenários, e avaliar o impacto de alguns parâmetros de configuração. Além disto, o estudo do modelo permitiu o entendimento de alguns detalhes da arquitetura, e validar seu funcionamento.

Finalmente, uma importante contribuição é a implementação da arquitetura em ambiente de simulação no *ns-2* e a avaliação dos diversos mecanismos e resultados. Com esta implementação foi possível estabelecer outros importantes mecanismos tais como *hellos* disparados, supressão de *hellos* e eliminação de máximo local. Pelos resultados de simulação, também foi possível estabelecer os pontos frágeis da arquitetura, que podem ser modificados para incrementar a escalabilidade, como por exemplo, o encaminhamento de mensagens de anúncio.

6.2 Trabalhos Futuros

A implementação da arquitetura em ambiente de simulação abre um grande conjunto de possibilidades de trabalhos futuros. Como principal, indicamos a incorporação de um mecanismo de busca estruturada na terceira camada, podendo ser o Symphony, conforme proposto.

Uma outra possibilidade de trabalho futuro é uma avaliação da arquitetura quando associada a outros protocolos de roteamento além do AODV. Além disto, existe a possibilidade de estabelecer mecanismos de otimização da arquitetura quando associada aos diversos protocolos de roteamento, através da otimização entre camadas.

Outros trabalhos podem ser indicados como continuidade do trabalho desenvolvido, podendo ser obtidos através do estabelecimento de novos paradigmas tais como: adotar a arquitetura como mecanismo de roteamento, fazendo com que cada nó se cadastre nos nós diretórios e que o roteamento de mensagens seja um serviço; associação da arquitetura com mecanismos de roteamento geográfico, e utilizar a arquitetura como estrutura para armazenamento posicional; e implementar serviços de compartilhamento de arquivos ou de mecanismos de segurança sobre a arquitetura.

6.3 Considerações finais

Conforme modelo analítico e resultados de simulações, a arquitetura proposta apresenta desempenho satisfatório, com taxas de descoberta muito próximas a 100% e uma boa escalabilidade ao não ter um aumento significativo do número de mensagens com o aumento do número de nós. Desta forma consideramos que os objetivos propostos na Seção 1.2, de obter escalabilidade e robustez, foram alcançados. Além disto, acreditamos que a arquitetura proposta possa ser adotada com sucesso no estabelecimento de redes *ad hoc*, ou outros ambiente cooperativos, onde não seja possível realizar configuração prévia.

Conquanto não tenha sido efetivamente comparada com outras propostas, os resultados mostram escalabilidade superior a daquelas propostas baseadas em inundação completa na rede. Em relação às demais propostas, que também adotam o uso de nós diretórios, os resultados indicam bom desempenho, com elevadas taxas de descoberta mesmo frente a grande mobilidade. Isto, apesar da dificuldade de estabelecimento de métricas unificadas e de cenários equivalentes que permitam uma comparação precisa entre todas as propostas existentes.

Referências Bibliográficas

[1] ZHU, F., MUTKA, M. W., E NI, L. M. Service discovery in pervasive computing environments. *Pervasive Computing, IEEE 4*, 4 (2005), 81–90.

[2] WU, J., E STOJMENOVIC, I. Ad hoc networks. *Computer, IEEE 37*, 2 (2004), 29–31.

[3] PERKINS, C. E., E BHAGWAT, P. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications* (New York, NY, USA, 1994), ACM Press, pp. 234–244.

[4] JINI. Jini Network Technology, 2005. `http://www.sun.com/software/jini/ - 13/12/2005`.

[5] *Jini Network Technology - DataSheet*, 2001.

[6] LUO, H., E BARBEAU, M. Performance evaluation of service discovery strategies in ad hoc networks. In *Second Annual Conference on Communication Networks and Services Research* (2004), pp. 61–68.

[7] ZEILENGA, K. RFC 4510 - lightweight directory access protocol (ldap): Technical specification road map. *IETF Request for Comments* (junho de 2006).

[8] RFC2608. Service Location Protocol, Version 2, junho de 1999. `ftp://ftp.rfc-editor.org/in-notes/rfc2608.txt - 13/12/2005`.

[9] UPNP. UPnP Forum, 2005. `http://www.upnp.org/ - 13/12/2005`.

[10] ROWSTRON, A., E DRUSCHEL, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science 2218* (2001), 329.

[11] ZHAO, B. Y., KUBIATOWICZ, J. D., E JOSEPH, A. D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Relatório Técnico UCB/CSD-01-1141, UC Berkeley, abril de 2001.

[12] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., E SHENKER, S. A scalable content addressable network. Relatório Técnico TR-00-010, University of California at Berkeley, Berkeley, CA, 2000.

[13] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, F., E BALAKRISHNAN, H. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference* (2001), pp. 149–160.

[14] MANKU, G., BAWA, M., E RAGHAVAN, P. Symphony: Distributed hashing in a small world. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS 2003)* (março de 2003).

[15] GNUTELLA. Gnutella Protocol Development, 2005. `http://rfc-gnutella.sourceforge.net/ - 13/12/2005`.

[16] JXTA. JXTA technology, 2005. `http://www.jxta.org/ - 13/12/2005`.

[17] LENDERS, V., MAY, M., E PLATTNER, B. Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (junho de 2005), pp. 120–130.

[18] KOODLI, R., E PERKINS, C. E. *Service Discovery in Ad Hoc Networks - draft-koodli-manet-servicediscovery-00.txt*, outubro de 2002. `http://www.watersprings.org/pub/id/index-k.html - 13/12/2005`.

[19] GARCIA-MACIAS, J. A., E TORRES, D. A. Service discovery in mobile ad-hoc networks: better at the network layer? In *ICPP 2005 Workshops. International Conference Workshops on Parallel Processing* (junho de 2005), pp. 452–457.

[20] VARA, M. I., CABERO, J. M., JODRÁ, J. L., E FAJARDO, J. O. Service Discovery Protocol in Proactive Mobile Ad Hoc Networks. In *Fourth Annual Mediterranean Ad Hoc Networking Workshop - MedHoc* (junho de 2005).

[21] SAILHAN, F., E ISSARNY, V. Scalable Service Discovery for MANET. In *3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'2005)* (março de 2005), pp. 235–244.

[22] KOZAT, U. C., E TASSIULAS, L. Network layer support for service discovery in mobile ad hoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies.* (2003), vol. 3, pp. 1965–1975.

[23] KREUTZER, M., E KÄHMER, M. Ubiquitous computing technology in infrastructureless environments. In *IEEE International Conference on Systems, Man and Cybernetics* (outubro de 2004), vol. 6, pp. 5639 – 5644.

[24] KOUBAA, H., E FLEURY, E. Service location protocol overhead in the random graph model for ad hoc networks. In *ISCC 2002 - Seventh International Symposium on Computers and Communications* (2002), pp. 49–54.

[25] KOZAT, U. C., E TASSIULAS, L. Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks 2* (janeiro de 2004), 23–44.

[26] CONTI, M., GREGORI, E., E TURI, G. A Crosslayer Optimization of Gnutella for Mobile Ad Hoc Networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing* (maio de 2005), pp. 343–354.

[27] PUCHA, H., DAS, S. M., E HU, Y. C. Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks. In *WMCSA 2004. Sixth IEEE Workshop on Mobile Computing Systems and Applications.* (dezembro de 2004), pp. 163–173.

[28] YANG, B., E GARCIA-MOLINA, H. Designing a super-peer network. In *19th International Conference on Data Engineering, 2003.* (março de 2003), pp. 49–60.

[29] AUGUSTO, C. H. P., E DE REZENDE, J. F. Uma abordagem adaptativa e escalável para descoberta de serviços em redes ad hoc. In *XXIV Simpósio Brasileiro de Redes de Computadores, 2006* (maio de 2006), pp. 945–960.

[30] AUGUSTO, C. H. P., E DE REZENDE, J. F. An adaptive approach to service discovery in ad hoc networks. In *8th IFIP IEEE International Conference on Mobile and Wireless Communications Network, 2006.* (agosto de 2006), pp. 61–75.

[31] KAZAA. Kazaa, 2006.
`http://www.kazaa.com - 01/12/2006`.

[32] NAPSTER. Napster, 2006.
`http://www.napster.com/ - 12/12/2006`.

[33] KREUTZER, M., KÄHMER, M., E FALK, H. Service discovery with higher order services in mobile hospitals. In *17th IEEE Symposium on Computer-Based Medical Systems, 2004. CBMS 2004.* (junho de 2004), pp. 460 – 465.

[34] ADJIE-WINOTO, W., SCHWARTZ, E., BALAKRISHNAN, H., E LILLEY, J. The design and implementation of an intentional naming system. In *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles* (New York, NY, USA, 1999), ACM Press, pp. 186–201.

[35] HODES, T. D., CZERWINSKI, S. E., ZHAO, B. Y., JOSEPH, A. D., E KATZ, R. H. An architecture for secure wide-area service discovery. *Wirel. Netw. 8*, 2/3 (2002), 213–230.

[36] NIDD, M. Service discovery in DEAPspace. *Personal Communications, IEEE 8*, 4 (agosto de 2001), 39–45.

[37] BONJOUR. Bonjour - zero-configuration networking, 2006. http://developer.apple.com/networking/bonjour/index.html - 01/12/2006.

[38] BLUETOOTH PROFILE SPECIFICATIONS. Service Discovery Application Profile, 2006. http://bluetooth.com/Bluetooth/Learn/Technology/Specifications/ - 01/12/2006.

[39] VANTHOURNOUT, K., DECONINCK, G., E BELMANS, R. A taxonomy for resource discovery. *Personal and Ubiquitous Computing 9*, 2 (março de 2005), 81–89.

[40] PERKINS, C., BELDING-ROYER, E., E DAS, S. RFC 3561 - Ad hoc On-demand Distance Vector (AODV) routing. *IETF Request for Comments* (julho de 2003).

[41] JOHNSON, D. B., MALTZ, D. A., E HU, Y.-C. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, julho de 2004.

[42] CLAUSEN, T., E JACQUET, P. RFC 3626 - Optimized Link State Routing protocol (OLSR). *IETF Request for Comments* (outubro de 2003).

[43] MALKIN, G. RFC 2453 - Routing Information Protocol version 2. *IETF Request for Comments* (nov de 1998).

[44] GOLAND, Y. Y., CAI, T., LEACH, P., E GU, Y. *Simple Service Discovery Protocol/1.0 Operating without an Arbiter*, outubro de 1999.

[45] LV, Q., CAO, P., COHEN, E., LI, K., E SHENKER, S. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing* (New York, NY, USA, junho de 2002), ACM Press, pp. 84–95.

[46] FREENET. Freenet, 2006. http://freenetproject.org - 12/12/2006.

[47] KLEINBERG, J. The small-world phenomenon: An algorithmic perspective. In *STOC '00: Proceedings of the 32nd ACM Symposium on Theory of Computing* (2000), pp. 163–170.

[48] TRAVERS, J., E MILGRAM, S. An experimental study of the small world problem. *Sociometry 32*, 4 (1969), 425–443.

[49] DOVAL, D., E O'MAHONY, D. Nom: Resource Location and Discovery for Ad Hoc Mobile Networks. In *1st Annual Mediterranean Ad Hoc Networking Workshop, Medhoc -Net* (setembro de 2002).

[50] HELAL, S., DESAI, N., VERMA, V., E LEE, C. Konark - a service discovery and delivery protocol for ad-hoc networks. In *Third IEEE Conference on Wireless Communications and Networking (WCNC)* (março de 2003), vol. 3, pp. 2107–2113.

[51] XML. XML, 2006. `http://www.xml.org - 18/12/2006`.

[52] KOUBAA, H., E FLEURY, E. A performance study of a service covering protocol in ad hoc networks. In *Ninth IEEE International Conference on Networks, 2001.* (outubro de 2001), pp. 87–92.

[53] KOUBAA, H., E FLEURY, E. A fully distributed mediator based service location protocol in ad hoc networks. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE* (novembro de 2001), vol. 5, pp. 2949–2953.

[54] KOZAT, U. C., TASSIULAS, L., KONDYLIS, G., RYU, B., E MARINA, M. K. Virtual dynamic backbone for mobile ad hoc networks. In *IEEE International Conference on Communications, 2001. ICC 2001.* (junho de 2001), vol. 1, pp. 250–255.

[55] HELMY, A., GARG, S., PAMU, P., E NAHATA, N. Contact-Based Architecture for Resource Discovery (CARD) in Large Scale MANets. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing* (abril de 2003), p. 219.1.

[56] HELMY, A., GARG, S., NAHATA, N., E PAMU, P. CARD: a contact-based architecture for resource discovery in wireless ad hoc networks. *Mob. Netw. Appl. 10*, 1-2 (2005), 99–113.

[57] CONTI, M., MASELLI, G., E TURI, G. Design and evaluation of a flexible cross-layer interface for ad hoc networks. In *Fourth Annual Mediterranean Ad Hoc Networking Workshop - MedHoc* (junho de 2005).

[58] SETHOM, K., E AFIFI, H. A new service discovery architecture for sensor networks. In *Wireless Telecommunications Symposium, 2005* (abril de 2005), pp. 190–196.

[59] GRUBER, I., SCHOLLMEIER, R., E NIETHAMMER, F. Protocol for peer-to-peer networking in mobile environments. In *The 12th International Conference on Computer Communications and Networks. ICCCN 2003.* (outubro de 2003), pp. 121–127.

[60] PARK, V. D., E MACKER, J. P. Anycast routing for mobile networking. In *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE* (outubro de 1999), vol. 1, pp. 1–5.

[61] NS-2. The network simulator - ns-2, 2005.
http://www.isi.edu/nsnam/ns/ - último acesso em 08/04/2007.

[62] CRAMER, C., E FUHRMANN, T. Proximity neighbor selection for a DHT in wireless multi-hop networks. In *Fifth IEEE International Conference on Peer-to-Peer Computing, 2005. P2P 2005.* (agosto de 2005), pp. 3–10.

[63] LI, H., E YU, D. A statistical study of neighbor node properties in ad hoc network. In *Parallel Processing Workshops, 2002. Proceedings. International Conference on* (agosto de 2002), pp. 103–108.

[64] NS 2, T. N. S. The ns manual(formerly ns notes and documentation), 2007.
http://www.isi.edu/nsnam/ns/ns-documentation.html - último acesso em 08/04/2007.