MÉTODOS DE AUTO-CONFIGURAÇÃO EM APLICAÇÕES MÓVEIS PAR-A-PAR NÃO ESTRUTURADAS

DIEGO NEVES DA HORA

MÉTODOS DE AUTO-CONFIGURAÇÃO EM APLICAÇÕES MÓVEIS PAR-A-PAR NÃO ESTRUTURADAS

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

Orientador: Daniel Fernandes Macedo
Co-orientador: José Marcos Silva Nogueira

Belo Horizonte
Maio de 2012

da Hora, Diego Neves
M1234x Métodos de Auto-configuração em Aplicações Móveis Par-a-Par Não Estruturadas / Diego Neves da Hora. — Belo Horizonte, 2012
xxii, 50 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Daniel Fernandes Macedo
Co-orientador: José Marcos Silva Nogueira

1. Par-a-Par. 2. MANET. 3. Aprendizado de Máquina. 4. redes adaptativas. I. Título.

CDU 100.0*01.10

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Métodos de auto-configuração em aplicações móveis par-a-par não estruturadas

DIEGO NEVES DA HORA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. DANIEL FERNANDES MACEDO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. JOSÉ MARCOS SILVA NOGUEIRA - Co-orientador
Departamento de Ciência da Computação - UFMG

PROF. ADRIANO ALONSO VELOSO
Departamento de Ciência da Computação - UFMG

PROF. ARTUR ZIVIANI
Laboratório Nacional de Computação Científica - CNPq

PROF. SÉRGIO DE OLIVEIRA
Departamento Multidisciplinar de Tecnologia, Ciências Humanas e Sociais - UFSJ

Belo Horizonte, 16 de maio de 2012.

Dedico este trabalho à memória de Mateus Neves.

Meu irmãozinho
que me ensinou tanto
sem precisar dizer nada.

Agradecimentos

A todos os amigos e colegas que tive o privilégio de conviver durante esta caminhada.

Aos colegas de laboratório (em especial Vinícius, Rone, Ewerton e Virgil), pois efetivamente fizeram parte do processo de desenvolvimento deste trabalho - tanto através das discussões, sugestões e comentários técnicos quanto pelos bons momentos de descontraçao nos *coffee-breaks*, encontros e sítios de final de semana.

Aos meus orientadores Daniel Macedo e José Marcos, pela confiança e motivação sempre presentes.

Agradeço aos meu pais, sobretudo por apostarem e investirem em mim desde cedo. Ao meu irmão pelo apoio e camaradagem.

Agradeço muito à minha esposa Irene, que esteve junto a mim durante meus momentos mais negros, sem jamais duvidar que dias melhores viriam. E eles vieram.

Agradeço a Deus: sem ele, nada disso seria possível.

"O mundo não tem lugar para covardes
Todos temos que estar dispostos a viver, lutar e morrer.
E teu esforço não é menor, por que nenhum
tambor rufa ao sair às suas lutas diárias.
E nenhuma multidão o aclama ao chegar,
vitorioso ou derrotado, de seus campos de batalhas diários"
(Autor Desconhecido)

Resumo

As aplicações Par-a-Par (P2P) devem ser configuradas de acordo com o ambiente em que estão sendo executadas a fim de obter melhor desempenho. A determinação da configuração ideal de parâmetros é custosa, necessitando de uma caracterização da rede em cada cenário. Assim, as redes P2P normalmente confiam em uma configuração genérica, que é aceitável em um grande número de cenários, mas que provê uma performance reduzida se comparado à melhor configuração manual de cada cenário.

Neste trabalho, investigamos maneiras de autoconfigurar aplicações P2P em ambientes móveis em tempo de execução. Utilizamos a arquitetura MAPE, proposta pela IBM para a criação de sistemas autonômicos, para propor duas soluções de autoconfiguração P2P. Nós propomos os controladores P-AIMD e P-ML, que configuram a aplicação em tempo de execução através das quatro etapas do controlador MAPE, de nome "monitoramento", "análise", "planejamento" e "execução". Na solução P-AIMD, utilizamos um perceptron na etapa de análise e planejamos a adaptação utilizando o algoritmo AIMD - "Aumento Aditivo, Decremento Multiplicativo", utilizado no controle de congestionamento do TCP. Na solução P-ML, utilizamos algoritmos clássicos de classificação na etapa de análise e utilizamos um simples algoritmo de planejamento de mudanças que utiliza a classificação realizada na etapa anterior para identificar a necessidade de mudança. Os controladores foram avaliados em uma rede móvel ad hoc empregando o protocolo Gnutella - no entanto essa solução pode ser aplicada sobre outros protocolos P2P não estruturados. Comparamos o desempenho das soluções proposta com o protocolo *Expanding Rings*, e os resultados de simulação mostram que o P-AIMD e o P-ML apresentaram, respectivamente, taxa de sucesso $5,18$ e $2,71$ pontos percentuais inferior à melhor configuração manual nos cenários simulados.

Palavras-chave: **Par-a-Par, MANET, Aprendizado de Máquina, redes adaptativas**

Abstract

Peer-to-Peer (P2P) applications must be configured according to the environment in which they are executed, in order to achieve the maximum performance. The identification of the ideal parameter configuration requires the characterization of the network in each deployment. Usually, P2P networks employ a generic default configuration, which is suitable to most scenarios, however its performance is worse than that of the best manual configuration. This work investigates methods to automatically configure the parameters of mobile P2P applications on runtime. We employ the MAPE architecture, proposed by IBM to create autonomic systems, in order to devise two solutions for P2P self-configuration. We propose the P-AIMD and P-ML controllers, which configure the application on run-time using the four phases of MAPE controllers, namely monitoring, analysis, planning and execution. In P-AIMD, the analysis and planning phase employ the Additive Increase and Multiplicative Decrease (AIMD) algorithm used in TCP for congestion control. In P-ML, we employ classic machine learning techniques in the analysis phase, and then employ a simple planning algorithm that uses the classification performed in the previous step to identify the need for configuration changes. The controllers were evaluated in mobile ad hoc networks running the Gnutella protocol, however the proposed solutions are applicable to any unstructured P2P protocol. We compared the performance of the proposed solutions against the Expanding Rings protocol, and the simulation results show that P-AIMD and P-ML present a success rate that is 5.18 and 2.71 percent superior to that of the best manual configuration, respectively.

Keywords: **Peer-to-Peer, MANET, Machine learning, Adaptive networks**

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Por serem independentes de infraestrutura prévia, as redes móveis ad hoc (MANETs) possibilitam a pronta execução de aplicações de apoio a resgate em situações de desastre e de troca de informações em campos de batalha, que de outra maneira encontrariam muitos obstáculos para sua execução [Borg, 2003]. Elas são compostas por dispositivos móveis que se comunicam diretamente usando rotas de múltiplos saltos (*multi-hop*) sem fio.

As redes Par-a-Par possuem grande sinergia com as redes móveis ad hoc. Elas buscam dividir a carga de trabalho entre os nós da redes - uma vez que, tipicamente, os nós de uma rede ad hoc possuem capacidades de bateria, conexão e processamento similarmente limitadas. Além disso, as redes Par-a-Par lidam bem com a perda ou ausência de conexões, uma vez que frequentemente são projetadas para lidar com a entrada e saída de pares na rede. No entanto, devido à dinamicidade e diversidade das redes ad hoc, torna-se difícil configurar os parâmetros das redes P2P.

Todas as redes par-a-par possuem parâmetros a serem ajustados. Em uma rede Gnutella [Adar & Huberman, 2000], por exemplo, tem-se que configurar o número de vizinhos e o TTL das mensagens de pesquisa. A escolha do valor de cada parâmetro impacta no desempenho da aplicação: o aumento do TTL garantirá maior abrangência da pesquisa, no entanto demandará mais recursos da rede. Este compromisso é modificado ainda por características da rede: se a rede em questão possuir uma carga alta taxa de utilização de recursos poderá resultar num crescente congestionamento de pacotes, degradando severamente o desempenho da aplicação.

Existem diversas formas de se abordar o problema da configuração de parâmetros. Pode-se utilizar uma configuração padrão, definida tipicamente pelo desenvolvedor da aplicação. Esta abordagem funciona bem se a rede em questão for conhecida ou de dimensões aproximáveis. É o caso da utilização da Gnutella sobre a internet, onde

diversos estudos apontam qual o melhor valor para vários parâmetros, em especial o TTL [Ripeanu, 2001] - cujo valor padrão estabeleceu-se em "7".

No entanto, as redes móveis ad hoc se apresentam em grande diversidade. Uma rede ad hoc pode possuir apenas alguns poucos elementos, como também dezenas ou centenas de participantes. Elas podem também no padrão de mobilidade, qualidade do enlace e taxa de utilização. De fato, as redes móveis ad hoc são especialmente sensíveis a colisões de pacotes, devido ao meio físico compartilhado [da Hora et al., 2009]. Como mostraremos adiante, a utilização de uma configuração padrão apresenta desempenho que poderia ser melhorado em diversos cenários.

Pode-se também realizar um estudo anterior à implantação de cada instância da rede, para configuração manual dos parâmetros. Essa abordagem enquanto possibilita resultados muito bons, pode ser muitas vezes impraticável. Essa abordagem requer um especialista no protocolo para realizar a configuração dos parâmetros e é propícia a erros humanos. Além disso, o tempo necessário para essa estimativa pode ser proibitivo na utilização dessa abordagem em algumas situações, como a utilização em redes de emergência.

Propomos que os parâmetros podem ser auto-configurados. Pode-se monitorar a rede e estimar, de forma algorítmica, o valor ideal para os parâmetros em cada instância da rede. Enquanto a proposta de auto-configuração de parâmetros não seja novidade [Ganek & Corbi, 2003], sua utilização e implementação em redes P2P sobre redes ad hoc não foram propostas na literatura. Mostraremos que a auto-configuração de parâmetros de redes P2P sobre redes ad hoc é viável, e apresenta desempenho comparável à melhor configuração manual de cada cenário.

1.1 Objetivos

O objetivo deste trabalho é propor algoritmos de auto-configuração de parâmetros otimizado para aplicações par-a-par não estruturadas sendo executadas sobre uma rede móvel ad hoc. Realizamos nossos estudos no Gnutella, uma rede par-a-par não estruturada, e propomos dois algoritmos de auto-configuração do tempo de vida das mensagens de pesquisa.

1.2 Contribuições

As contribuições desse trabalho são:

- **Auto-configuração de redes P2P com controlador MAPE:** Utilizando a definição de controlador MAPE de Corbi et al. [Ganek & Corbi, 2003], propomos resolver o problema da configuração automática de parâmetros utilizando um controlador MAPE, como pode ser visto na Seção 3.2.

- **Controlador MAPE AIMD:** Propomos um controlador MAPE baseado na conhecida técnica de controle de fluxo *Additive Increase, Multiplicative Decrease (AIMD)*. Esta técnica ajusta o parâmetro controlado a partir de estimativas de congestionamento, modeladas nesta proposta como um estimador perceptron de uma entrada. O perceptron por sua vez identifica congestionamento através da estimativa da ocupação média das filas de roteamento dos pares. A descrição da implementação deste controlador pode ser encontrada na Seção 3.3.

- **Controlador MAPE com técnicas de aprendizado de máquina:** Ainda dentro do arcabouço MAPE, propomos um controlador que faz uso de técnicas modernas de aprendizado de máquina para realização da configuração dos parâmetros. Modelamos a etapa de "Análise" do controlador MAPE como um problema de Classificação. Levantamos, através de simulações, uma base de dados para seu treinamento e comparamos o resultado de diversos algoritmos de classificação modernos. Propomos um algoritmo simples para modificação do parâmetro controlado utilizando o resultado da análise? A descrição de todas estas etapas pode ser encontrada na Seção 3.4.

1.3 Organização do Texto

O restante do texto se organiza da seguinte forma. No Capítulo 2, introduzimos os conceitos fundamentais sobre redes e Aprendizado de máquina. No Capítulo 3, abordamos o problema de configuração de parâmetros e propomos duas soluções, utilizando o arcabouço MAPE para auto-configuração de parâmetros. No Capítulo 4 descrevemos a metodologia de pesquisa utilizada, quais foram as métricas para avaliação e validação da solução e analisamos todos os algoritmos propostos em cenários variados. Apresentamos as conclusões da análise no Capítulo 5, onde também indicamos propostas de trabalhos futuros.

Capítulo 2

Conceitos de Redes e Aprendizado de Máquina

Este capítulo discute os principais conceitos utilizados neste trabalho. A Seção 2.1 define o que é uma rede móvel ad hoc, discute os possíveis cenários onde sua utilização é desejada e quais aplicações ela suporta. A Seção 2.2 apresenta os conceitos de rede P2P e discute as diferenças entre as principais redes Par-a-Par, mostrando suas vantagens e limitações. A Seção 2.3 apresenta os conceitos relacionados a Aprendizado de Máquina. Na Subseção 2.3.1 é definida a tarefa de classificação. Na Subseção 2.3.2 são apresentadas as vantagens e desvantagens do uso de Aprendizado de Máquina. A Seção 2.4 finaliza este capítulo com algumas considerações gerais.

2.1 Redes Móveis Ad Hoc

As redes móveis ad hoc (*MANETs*) são redes compostas por dispositivos móveis que se comunicam diretamente utilizando rotas de múltiplos saltos (*Multi-hop*). Elas possuem custo de instalação inferior às redes cabeadas e permitem aos seus usuários uma maior área de cobertura. As redes sem fio não estruturadas podem ser utilizadas onde não existe incentivo econômico para a instalação de infra-estrutura de rede. Nestas redes, os dispositivos sem fio repassam os dados entre si, encaminhando-a até que a informação encontre seu destinatário. Os nós que compõem a rede podem ser móveis como notebooks, tabletes e telefones celulares, ou fixos, como pontos de acesso.

O paradigma de redes ad hoc pode ser utilizado em diversas situações. Elas podem ser utilizadas na formação de redes de emergência, uma rede formada para prover comunicação entre dispositivos móveis a fim de auxiliar o trabalho de equipes de ajuda e resgate. Ela é adequada para esse cenário pois independe de infra-estrutura

prévia e possui baixo tempo de instalação. Elas também podem surgir espontaneamente em ambientes urbanos, onde dispositivos móveis interagem e colaboram para formação de uma rede a fim de possibilitar troca de arquivos e mensagens. Pode-se, ainda, utilizar redes ad hoc na formação de uma rede que interliga netbooks de baixo custo, como os computadores do projeto "UCA" [Federal, 2012], a fim de formar uma rede de troca de informações em locais com dificuldade de acesso a internet.

As rede móveis ad hoc são alvo de grande interesse de pesquisa, com diversos estudos acerca do desempenho da aplicação de protocolos típicos de redes estruturadas neste cenário, estudos de melhorias de protocolos já existentes e propostas de protocolos próprios para redes ad hoc. A pesquisa em redes ad hoc é variada, com estudos nas áreas de roteamento [Wang et al., 2009; Seys & Preneel, 2009], descoberta de conteúdo [Mian et al., 2009] e segurança [Sumathy & Kumar, 2010].

No entanto a definição do que, de fato, é uma rede ad hoc é um pouco tênue. Se frequentemente os nós participantes se desconectam da rede, podendo passar longos períodos sem rota entre um dado par de nó, essa rede se aproxima mais de uma rede tolerante a atrasos (DTN), como proposto por [Fall, 2003]. [Almeida, 2012] realizou um estudo sobre o desempenho de protocolos de roteamento Ad Hoc e DTN em cenários de emergência, comparando quando é melhor abordar esse tipo de rede como rede ad hoc ou como rede DTN. Verificou-se que se deve abordar essas redes como redes ad hoc quando existem, tipicamente, múltiplas rotas de conexão entre todos os participantes de rede.

2.2 Redes Par-a-Par

No paradigma cliente-servidor todos os clientes se comunicam com um único servidor. Esta característica torna este servidor um elemento com alta demanda de recursos, bem como suas vias de acesso tipicamente se tornam "gargalo" da rede. Além disso, frequentemente o servidor se torna um "ponto único de falha", uma vez que a falha do servidor resulta na indisponibilidade do recurso a todos os clientes.

O paradigma Par-a-Par difere do paradigma cliente-servidor em que todos os pares podem ser clientes e servidores simultaneamente. Em uma rede Par-a-Par de troca de arquivos, cada par serve arquivos para a rede bem como pode requisitar arquivos de outros pares por exemplo. Esta descentralização das responsabilidades traz alguns benefícios não encontrados no paradigma cliente-servidor. Não há "ponto único de falha", uma vez que a falha de um dos pares indisponibilizará apenas o conteúdo exclusivo daquele par. Além disso, como todo par pode assumir a tarefa de servidor,

ocorre de forma natural a descentralização da carga de trabalho. Isto permite que a tarefa de servir a rede seja feita com a demanda por recursos computacionais distribuída pela rede, não necessitando assim de equipamento especializado para servir a rede.

Uma das notórias aplicações de redes Par-a-Par é a descoberta de conteúdo. Cada par possui uma lista de conteúdos que possui e, possivelmente, dos conteúdos que outros pares possuem. Se um dado par necessita encontrar uma determinada informação, ele realiza uma pesquisa pelo conteúdo seguindo os mecanismos fornecidos pelo protocolo Par-a-Par. Existem duas principais abordagem para descoberta de conteúdo em redes P2P: abordagem *estruturada* e *não estruturada*.

Na abordagem *estruturada*, os pares constroem índices distribuídos e as pesquisas são implementadas como operações de busca nestes índices. No estado da arte, a busca ocorre em $O(log(N))$ mensagens. No protocolo Chord [Stoica et al., 2001], cada um dos N pares recebe um índice e constrói uma tabela com aproximadamente $log_2(N)$ entradas, contendo cada uma o índice e localização de outros pares. As pesquisas são feitas associando uma chave ao conteúdo desejado, normalmente por uma função *hash* como o SHA-1. O objetivo é descobrir qual par possui índice imediatamente superior à chave, pois ele é responsável por armazenar as informações de localização do conteúdo. Por fim, a pesquisa ocorre através de uma série de encaminhamentos, repassando a pesquisa a pares da tabela hash que possuam índice cada vez mais próximos ao da chave, sendo que o limite superior do número de encaminhamentos é $O(log(N))$.

Em redes par-a-par *não estruturadas*, a descoberta de conteúdo é realizada através da propagação da mensagem de pesquisa, frequentemente por mecanismo de inundação ou algum refinamento de inundação, como o random-walking([Servetto & Barrenechea, 2002]). Cada par possui uma lista de vizinhos e as mensagens de pesquisa são criadas com um "Tempo de vida" (TTL). As pesquisas são realizadas repassando as mensagens de pesquisas a todos os vizinhos e decrementando seu TTL, sendo que quando o TTL chega a 0 a mensagem não é mais repassada. Caso um par receba uma mensagem de pesquisa que possua o conteúdo procurado, ele deve enviar uma mensagem ao nó originário da pesquisa com a informação desejada. A grande vantagem da inundação é que ela é altamente resistente à perda de pacotes - uma realidade em redes móveis ad hoc. Por outro lado, o uso indiscriminado de inundação pode levar a uma sobrecarga dos nós da rede, levando a frequentes perdas e retransmissões de pacotes, eventualmente diminuindo o número de resposta por consulta, degradando a qualidade do serviço.

O Gnutella é uma rede Par-a-Par não estruturada [Ripeanu, 2001]. Por ser uma das primeiras redes Par-a-Par não estruturadas propostas, seu funcionamento é bastante simples e por esse motivo foi escolhido para análise neste trabalho. O Gnutella possui dois parâmetros: número de vizinhos e TTL. O número de vizinhos indica o

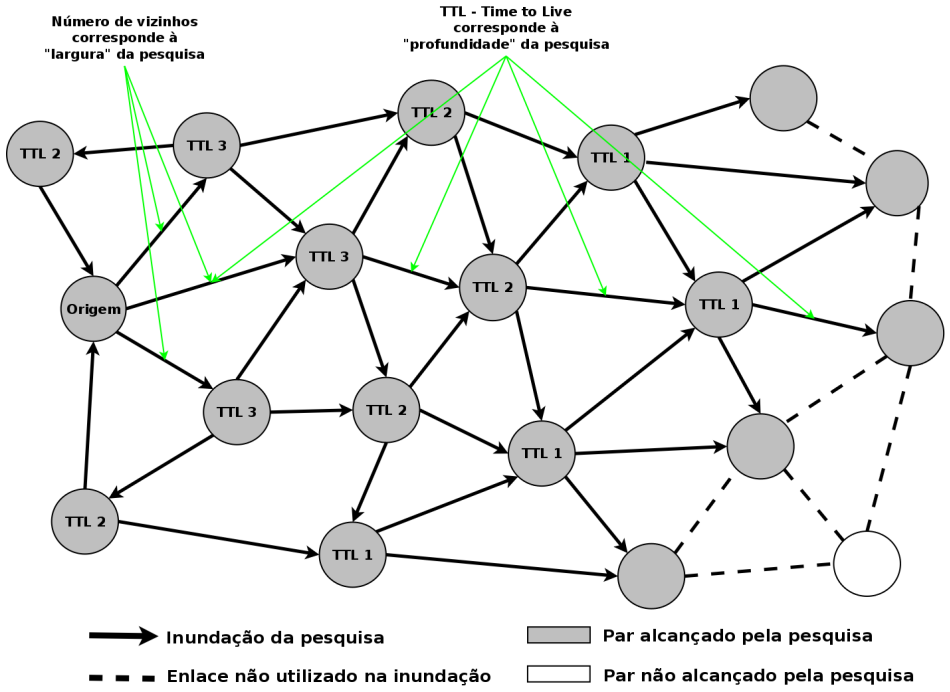

Figura 2.1. Exemplo de pesquisa no Gnutella. No exemplo, o par "Origem" realiza uma pesquisa com $TTL = 4$, e cada par possui 3 vizinhos

número de vizinhos da aplicação, utilizados no encaminhamento da pesquisa e o TTL é o valor inicial de TTL de cada mensagem de pesquisa. Temos um exemplo de pesquisa Gnutella na Figura 2.1. O par origem inicia uma pesquisa enviando uma requisição de conteúdo a todos os seus vizinhos com o seu valor de TTL. Cada par verifica se possui o conteúdo pesquisado quando recebe uma mensagem de pesquisa. Caso tenha o conteúdo, ele entra em contato com o par que iniciou a pesquisa a fim de informá-lo. Se o par não possui a informação desejada, ele encaminha a pesquisa aos seus vizinhos caso o TTL da pesquisa seja maior do que zero. Ao encaminhar, o TTL é decrementado em uma unidade.

Em [da Hora et al., 2009], fizemos um estudo do desempenho de ambas as abordagens no contexto de uma redes móvel ad hoc. Muito embora as redes P2P *estruturadas* são conhecidas, na literatura, por seu desempenho superior em redes cabeadas, nós descobrimos que redes P2P *não estruturadas* apresentam melhor desempenho na maioria dos ambientes estudados, devido à mobilidade dos nós e a pouca confiabilidade das conexões sem fio. Pesquisas em protocolos P2P *não estruturadas* usam frequentemente técnicas de inundação (flooding, mensagens de broadcast, *gossiping* e outros) a fim de alcançar diversos pares simultaneamente. Esta característica provê um alto grau de tolerância a falhas nas conexões e nos pares. No entanto, usar inundação reduz significativamente a escalabilidade destes protocolos, pois o congestionamento gerados

por muitas pesquisas é o principal motivo da perda de desempenho em grandes redes P2P em ambientes sem fio [da Hora et al., 2009].

A integração de MANETs e redes P2P é um tópico extensivamente pesquisado. Redes ad hoc são um ambiente inerentemente distribuído, de forma que o paradigma P2P se adequa melhor que o paradigma cliente-servidor às características das redes móveis sem fio. Oliveira et al. avaliaram o desempenho de uma rede não estruturada usando diversos protocolos de roteamento (DSR, AODV, DSDV) [Oliveira et al., 2005]. Franciscani et al. propuseram algoritmos de configuração que ajustam a topologia da rede P2P à topologia da rede ad hoc [Franciscani et al., 2005]. Conti et al. propuseram modificações ao Gnutella, utilizando uma abordagem *cross-layer* para reduzir o *overhead* do protocolo sobre MANETs [Conti et al., 2005].

Redes P2P adaptativas foram propostas para redes P2P sendo executadas sobre a Internet como uma forma de reduzir o tráfego da rede P2P. Em [Tsoumakos & Roussopoulos, 2003; Lv et al., 2002], os autores propõem mecanismos de pesquisa em vizinhanças de tamanho progressivo, que adaptam o número de saltos que uma mensagem percorre na rede. As consultas são feitas inicialmente em uma vizinhança reduzida, que aumenta progressivamente a cada passo caso a consulta não seja respondida. Jiang e Jim propuseram um mecanismo de duas fases, em que o TTL das pesquisas é ajustado de acordo com a popularidade da informação [Jiang & Jin, 2005]. O TTL ideal é calculado pela amostragem dos resultados obtidos em uma região reduzida da rede. Em seguida, a segunda fase emprega o TTL calculado nas consultas subsequentes.

Dentre os algoritmos adaptativos para redes P2P não estruturadas, uma das propostas com bom desempenho no contexto de redes estruturadas é o *Expanding Rings*, proposto por [Tsoumakos & Roussopoulos, 2003]. O protocolo *Expanding Rings* possui dois parâmetros, chamados aqui de τ e δ. Toda pesquisa é realizada por inundação, com TTL *tau*. Se a pesquisa falhar, repete-se a pesquisa com o TTL anterior acrescido de δ e repete-se o processo até que o TTL utilizado exceda certo valor limite. A adaptabilidade reside no fato que alguns conteúdos populares podem ser facilmente encontrados a poucos hops de distância, podendo com isso ser acessados utilizando um baixo TTL. No entanto, a abrangência da pesquisa aumenta caso a pesquisa não retorne resultados, a fim de pesquisar mais profundamente na rede. O *Expanding Rings* funciona de forma bastante satisfatória considerando uma rede Par-a-Par sendo executada sobre a internet, no entanto não existem estudos de seu funcionamento em redes móveis ad hoc.

2.3 Aprendizado de Máquina

O campo de Aprendizado de máquina estuda os processos computacionais que envolvem o aprendizado nas máquinas. Segundo [Langley, 1996]: *"O aprendizado consiste na melhoria do desempenho em algum ambiente através da aquisição de conhecimento como resultado da experiência neste ambiente."*.

Diversas tarefas podem ser automatizadas por uma máquina, desde as mais maçantes e triviais, como as repetitivas tarefas de uma linha de produção, até outras mais complexas e sofisticadas, como classificação de *spam* e reconhecimento facial. É especialmente nessas que se concentram os estudos de aprendizagem de máquina. Das diversas áreas focos de estudo de aprendizagem de máquina, a que nos é de interesse nos remete à tarefa de classificação.

Como o processo de aprendizagem envolve a melhoria do desempenho na tarefa, faz-se necessário que o algoritmo de aprendizagem tenha contato com o ambiente múltiplas vezes e que, a cada interação, armazene o aprendizado adquirido como fruto da interação. Esse processo pode ser chamado de "aprendizagem" ou ainda "treinamento". Se o treinamento ocorre durante o funcionamento em produção do sistema, damos a isso o nome de aprendizado *on-line*. Caso contrário, se o treinamento ocorre em um momento anterior ao funcionamento em produção do sistema, temos o aprendizado *off-line*.

2.3.1 Classificação

Podemos definir, também de forma bastante prática, a tarefa de classificação como a tarefa de associar um conjunto de dados de entrada (também chamados de *atributos*) a um dado conjunto de categorias (também chamadas de *classes*). A Tabela 2.1 ilustra esse relacionamento entre atributos e classe por meio de uma tabela de dados, frequentemente usada no aprendizado off-line.

$\mathbf{in_1}$	$\mathbf{in_2}$	...	$\mathbf{in_n}$	**classe**
$d_{1,1}$	$d_{1,2}$	...	$d_{1,n}$	c_1
$d_{2,1}$	$d_{2,2}$	...	$d_{2,n}$	c_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$d_{m,1}$	$d_{m,2}$	...	$d_{m,2}$	c_m

Tabela 2.1. Exemplo de conjunto de dados para treinamento *off-line* de um classificador. Em cada uma das m tuplas, os n atributos $\{in_1, in_2, \ldots, in_n\}$ devem ser mapeados em sua classe *class*.

Ao processo de ajustar os parâmetros internos do classificador para que ele "aprenda" a fazer as classificações desejadas dá-se o nome de treinamento. Existem duas formas de treinamento: *on-line* e *off-line*. No aprendizado *off-line*, apresenta-se um conjunto de dados ao classificador, similares aos presentes na Tabela 2.1. A esse primeiro conjunto de dados damos o nome de conjunto de treinamento. Com a utilização de algum algoritmo de aprendizagem o classificador irá, através da repetida apresentação dos atributos e classes associadas gerar uma regra de associação entre atributos e classe.

A fim de verificar se o classificador "aprendeu" a classificar os dados e "generalizou", e não apenas "decorou" os dados de treinamento, apresenta-se um segundo conjunto de dados, denominado "conjunto de testes". Nessa etapa, o classificador irá utilizar como entrada os atributos e irá classificá-los em uma classe. Comparando essa classificação com a verdadeira classe da tupla, descobrimos se ele acertou ou não em sua classificação. Tipicamente mede-se a qualidade de um classificador pelo percentual de classificações corretas.

Enquanto no aprendizado *off-line*, o classificador aprende ao ser apresentado a um conjunto de dados, o aprendizado *on-line* é um modelo que aprende uma instância por vez. É uma abordagem focada para que o aprendizado ocorra durante a utilização do classificador, onde um pouco de conhecimento é agregado a cada utilização. Fato importante no aprendizado on-line é que não se sabe a classe da instância antes que ela ocorra, muito embora muitas vezes seja possível saber após a utilização. Não utilizaremos esta abordagem em nossos estudos.

Pode-se realizar classificação através de inúmeras abordagens. Na abordagem via Árvores de Decisão, classificam-se os eventos a partir do caminhamento em uma árvore. Cada nó não terminal da árvore direciona o caminhamento baseado nos atributos do evento. Assim o evento caminha da raiz em direção a uma folha específica da árvore, esta indicando a qual classe o evento pertence. Diversos métodos existem para gerar a árvore de classificação a partir do conjunto de treinamento variando em complexidade computacional, dificuldade de implementação e generalidade da árvore de classificação [Quinlan, 1986]. Pode-se utilizar também classificação por Rede Neurais [Zhang, 2000], Algoritmos Genéticos [Ishibuchi et al., 1995] e através de métodos estatísticos [Androutsopoulos et al., 2000] – mas estes não serão foco de nossos estudos?

2.3.2 Vantagens e Limitações

Existem diversas situações onde a aplicação de algoritmos de classificação seja benéfica. Padrões complexos de correlacionamento entre atributos e classes, dificilmente

perceptíveis mesmo a um perito no assunto, podem ser explorados utilizando algoritmos modernos de classificação. Além disso, pode-se utilizar o grande poder computacional hoje disponível para se explorar níveis mais profundos de relacionamentos entre os atributos, a fim de obter um modelo mais complexo mas possivelmente mais preciso.

No entanto, aprendizagem de máquina possui limitações. Para que um algoritmo de classificação possa criar um modelo confiável relacionando atributos e classes, os dados repassados ao classificador devem apresentar essa relação - mesmo que não se saiba qual seja. Não se pode associar classes aleatórias às instâncias e esperar que o algoritmo "adivinhe" corretamente quais são as classes. De igual forma, não se pode esperar que o algoritmo de classificação encontrasse um modelo preciso se os atributos não possuem, de fato, correlação com as classes. Uma vez que a única tarefa dependente do homem é a de modelagem, esta deve ser realizada com bastante cuidado e cautela, para gerar um modelo válido e sem vícios.

2.4 Conclusão

As redes móveis ad hoc são importantes, e podem ser utilizadas para troca de informações em situações onde não há infra-estrutura de rede disponível. As redes Par-a-Par possuem grande sinergia com as redes ad hoc mas muitos problemas já resolvidos para redes P2P na internet devem ser revistos ao aplicados em redes móveis ad hoc, como a configuração de parâmetros.

Realizamos também uma revisão dos principais conceitos de aprendizagem de máquina e suas aplicações. Definimos a tarefa de classificação e quais as suas métricas de sucesso. Por fim, discutimos acerca dos benefícios e limitações do uso de aprendizagem de máquina, em especial nas tarefas de classificação. Com os conceitos aqui estabelecidos, seguimos no Capítulo 3 descrevendo a solução proposta para o problema em questão.

Capítulo 3

Controlador Autonômico

Neste Capítulo descrevemos as duas soluções que propomos para a auto-configuração do parâmetro TTL. A esses algoritmos de auto-configuração de parâmetros damos o nome de controlador P-AIMD e controlador P-ML, descritos nas Seções 3.3 e 3.4 respectivamente.

3.1 Definição do problema

O problema da configuração de parâmetros pode ser definido da seguinte forma: Seja um protocolo P com parâmetros $\Phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$. A utilização do protocolo P em um cenário C com parâmetros Φ nos resulta em uma métrica de desempenho m, ou seja: $P(\Phi, C) = m$.

Para cada cenário C_i, existe uma configuração Φ_i^* que apresenta desempenho m_i^* ótimo. Seja um algoritmo Λ que gere uma configuração de parâmetros Φ para cada cenário C, ou seja: $\Lambda(C) = \Phi$. Nosso objetivo é criar Λ tal que para todo $C_i \in \{C_1, C_2, \ldots, C_m\}$, a diferença entre $P(\Lambda(C_i), C_i) = m_i$ e m_i^* seja mínima.

3.2 Controlador MAPE

As soluções propostas baseiam na arquitetura de um controlador MAPE. Conforme a definição em [Computing, 2006], um controlador MAPE possui as seguintes etapas em seu laço de controle: Monitorar, Analisar, Planejar e Executar; de onde deriva seu nome. Podemos ver seu arcabouço de forma ilustrada na Figura 3.1.

Na etapa "Monitorar", o controlador MAPE irá coletar dados do sistema gerenciado utilizando alguma estratégia pré-definida. Isto se dará, tipicamente, utilizando

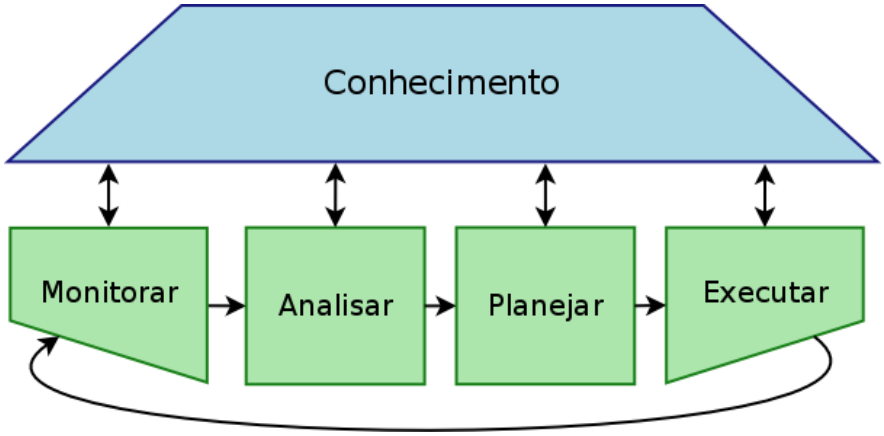

Figura 3.1. Laço de controle MAPE

um ou mais sensores. No caso específico de redes sem fio, diversas métricas podem ser monitoradas, como taxa de ocupação da fila de roteamento, atraso médio observado nas pesquisas e consumo de energia médio.

Na etapa "Analisar", o controlador MAPE irá utilizar os dados monitorados como entrada para decidir se existe alguma anormalidade e se esta deve ser gerenciada. É nessa etapa que o controlador identifica a necessidade de reação a alguma ameaça ou deficiência encontrada.

Na etapa "Planejar", o controlador MAPE irá utilizar dados da análise realizada, como o grau da necessidade de reação, para planejar uma modificação no sistema através dos recursos controlados. Nessa etapa é que se concentra toda a lógica de mensuração da intensidade da modificação nos recursos controlados a fim de se obter o resultado desejado. Caso as mudanças planejadas necessitem ser executadas em várias etapas, ele deverá "agendá-las".

Por fim, na etapa "executar" o controlador MAPE irá executar as modificações demandas pela etapa "Planejar", conforme o agendamento planejado.

A utilização de um controlador MAPE é adequada como arcabouço das soluções no problema de configuração de parâmetros. Sua estrutura de monitoração, análise e modificação pode ser utilizada para identificar situações onde perdas de desempenho estão presentes e modificações nos parâmetros são realizadas a fim de minimizar estas perdas. Esta característica de minimização das perdas de desempenho é muito desejável devido à modelagem do problema. Utilizando, portanto, as definições deste arcabouço, iremos agora descrever a arquitetura das duas soluções propostas.

3.3 Controlador P-AIMD

Este controlador se baseia no algoritmo "Aumento Aditivo, Decremento Multiplicativo", do inglês *Aditive Increase, Multiplicative Decrease*, de onde deriva sua sigla AIMD. Um dos exemplos mais notáveis do uso desse algoritmo é o controle de congestionamento do TCP, que ajusta o tamanho da Janela deslizante utilizando um algoritmo AIMD [Chiu & Jain, 1989]. Descreveremos o funcionamento do controlador P-AIMD em termos de um controlador MAPE.

3.3.1 Etapa "Monitorar"

Em [da Hora et al., 2009], vemos que o desempenho das redes Par-a-Par não estruturadas é severamente limitado caso ocorra congestionamento de pacotes. O algoritmo de inundação, presente em todas as abordagens Par-a-Par não estruturadas, envia um elevado número de mensagens, sobretudo quando possui um TTL acima do necessário. Assim, configurar o TTL de forma adequada em uma rede Par-a-Par não estruturada pode se traduzir em encontrar um TTL que seja alto o suficiente para garantir uma boa cobertura da rede, mas que de maneira alguma gere um congestionamento exacerbado na rede.

Em redes 802.11, a vazão de um nó depende do número de pacotes enviados pela sua vizinhança [Malone et al., 2007; Tanenbaum, 2002]. O método de acesso ao meio IEEE 802.11 CSMA/CA utiliza um algoritmo de *backoff* exponencial a fim de mitigar a contenção e resolver as colisões. Em redes com alta demanda, este algoritmo tenderá a aumentar o atraso na entrega dos pacotes, pois frequentemente irá encontrar o meio ocupado, aumentando o tamanho da fila na camada de roteamento. Sobre cargas amenas, a rede tenderá a diminuir sua fila de pacotes, uma vez que frequentemente irá encontrar o canal ocioso. Com isso, temos duas possíveis métricas indicadoras de congestionamento: a latência média dos encaminhamentos de pacotes e o tamanho médio da fila na camada de roteamento.

Optamos nesta solução por utilizar apenas a ocupação na fila da camada de roteamento como métrica indicadora de congestionamento. Em uma primeira abordagem, assumimos que cada par poderia monitorar a própria fila de roteamento, e repassar os dados para a etapa de análise. No entanto, simulações preliminares mostraram que na rede foco do estudo, os nós que ocupam as posições centrais da rede tendem a fazer parte de muitas rotas, aumentando assim grandemente o tamanho de sua fila, enquanto nós periféricos podem ter poucos ou nenhum pacote em suas filas.

Devido a essa heterogeneidade, adotamos a seguinte estratégia: cada par irá

aproximar a ocupação média das filas de roteamento de sua vizinhança. Dessa forma os pares irão encontrar estimativas mais similares, produzindo análises da rede mais unânimes. A seguir, propomos um algoritmo escalável para disseminação da ocupação média das filas de roteamento.

3.3.1.1 Algoritmo de disseminação da ocupação média

A ocupação de um par j, considerando uma vizinhança de até i saltos, é calculada na Equação 3.2. A fim de derivar este valor definimos $n_{i,j}$ como o número de vizinhos de j em uma vizinhança de até i saltos, como pode ser visto na Equação 3.1. Estas equações assumem que $o_{0,j}$ é a ocupação local do par j e que V_j é o conjunto de pares vizinhos do par j.

$$n_{i,j} = \begin{cases} 1 & i = 0 \\ |V_j| & i = 1 \\ \sum_{p \in V_j} n_{i-1,p} & i > 1 \end{cases} \quad (3.1)$$

$$o_{i,j} = \frac{\sum_{p \in V_j} (o_{i-1,p} \times n_{i-1,p})}{n_{i,j}}, i > 1 \quad (3.2)$$

Esta aproximação pode ser facilmente implementada em uma rede Par-a-Par. Todo par j envia aos seus vizinhos, periodicamente, sua ocupação $o_{0,j}$. Dessa forma, todo par terá em mãos a ocupação individual dos vizinhos e poderá calcular a ocupação média dos vizinhos $o_{1,j}$. Realizado o cálculo de $o_{1,j}$, ele deverá repassar esse valor aos seus vizinhos e receber a deles, podendo calcular $o_{2,j}$ e assim por diante. De forma prática, basta cada par j enviar um vetor O e N de tamanhos i a todos seus vizinhos V. Cada par armazena os vetores O e N recebidos dos vizinhos, a fim de possuir dados para computar e atualizar os vetores O e N locais.

Na nossa implementação utilizamos $i = 3$ e implementamos a troca de vetores O e N inserindo-os no corpo das mensagens *ping* e *pong*, nativas do Gnutella. Fizemos isso pois o tamanho, em bytes, das informações trocadas é fixo - e em nossa implementação não houve a necessidade de aumentar o número de pacotes enviados, não gerando com isso *overhead* para a realização deste cálculo. A preocupação com a não geração de tráfego adicional é consonante com o objetivo das duas soluções de evitar congestionamentos por acúmulo de pacotes e por princípio não gostaríamos de gerar nenhum tráfego adicional. Não analisamos o *overhead* gerado pela implementação de um algoritmo sem utilização das mensagens *ping* e *pong*, ou ainda com comunicação direta entre todos os pares para troca dessas informações. Para uma análise mais

acurada do *overhead* gerado por algoritmos similares de propagação de informações, confira Macedo et al. [2009].

De fato, o valor $o_{i,j}$ é uma aproximação da ocupação média dos vizinhos de j de até i saltos. Percebe-se que o algoritmo implementa uma média ponderada entre a ocupação média calculada até então e o número de medições agregadas. Considere uma topologia em que a e b são vizinhos, e ambos possuem vizinhança com c. Os valores $o_{1,j}$ são calculados através da média dos vizinhos, logo c é computado em $o_{1,a}$ e $o_{1,b}$. No entanto, ao se computar $o_{2,a}$ por exemplo, a receberá $o_{1,j}$ de cada um de seus vizinhos (bem como o seu próprio) e calculará a média ponderada. No entanto, o valor c contribuirá para $o_{2,a}$ mais de uma vez, já que ele foi computado tanto em $o_{1,a}$ quanto em $o_{1,b}$.

Outro motivo que atesta que $o_{i,j}$ é apenas uma aproximação reside no fato que $o_{i,j}$ depende do cálculo de $o_{i-1,j}$. De fato, $o_{i,j}$ só pode ser calculado após i iterações, e possuirá um valor relativo a i iterações no passado. Considere uma rede Par-a-Par que troca informações de ocupação com periodicidade de 5 segundos. Todo par j calcula $O_{0,j}$ referente ao momento atual da rede, $O_{1,j}$ referente a 5 segundos atrás e $O_{i,j}$ referente a $5 \times i$ segundos atrás. Por isso, embora considerar níveis maiores de vizinhança seja benéfico em termos da estabilidade e homogeneidade das medições, esses fatores expostos nos limitam ao considerarmos profundidades de vizinhança cada vez maiores.

3.3.2 Etapa "Analisar"

O objetivo dessa etapa é analisar, utilizando os dados da etapa de monitoramento, se há necessidade de intervenção no sistema. O desafio então é identificar, utilizando a ocupação média da vizinhança definida na função anterior, a existência ou não de indícios de congestionamento.

Nessa proposta, a etapa de análise será realizada de maneira bastante simplória e intuitiva. Seja um valor limite t, tal que se verifique experimentalmente que caso a ocupação média esteja menor que t provavelmente não há congestionamento e caso a ocupação média esteja maior que t provavelmente há congestionamento. Precisamos então encontrar esse valor limite t bem como indicar o congestionamento para a etapa de planejamento, caso ele seja identificado.

3.3.2.1 Valor limite t

Com relação ao cálculo do valor limite de congestionamento t, realizamos alguns experimentos preliminares a fim de conseguir uma maneira de obtê-lo. Sabemos que a

taxa de sucesso é diretamente afetada pelo congestionamento em redes Par-a-Par não estruturadas [da Hora et al., 2009]. Tomamos por cenário base o descrito na Subseção 4.1.1 e variamos o número de pesquisas por segundo, a fim de verificar o impacto do congestionamento no desempenho da aplicação.

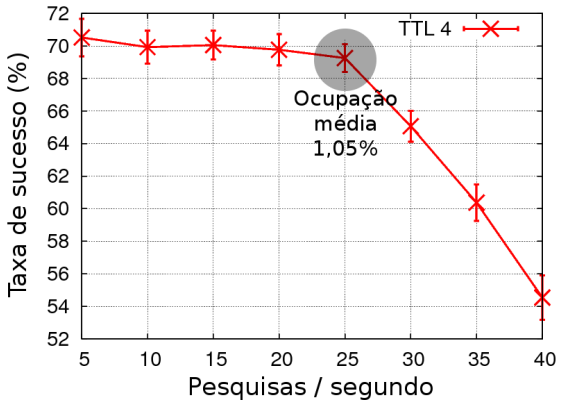

Figura 3.2. Valor aproximado de t, para $TTL = 4$

Pode-se ver na Figura 3.2 que, para TTL = 4, a taxa de sucesso das pesquisas é similar nos cenários com menos do que 25 pesquisas por segundo. No entanto, a taxa de sucesso cai consideravelmente para mais do que 25 pesquisas por segundo, indicando que a rede está operando próxima do limite de sua capacidade. Este ponto de operação imediatamente antes do congestionamento é o que possui maior utilização dos recursos sem observar a perda de desempenho típica dos congestionamentos. Assim, nós consideramos o valor médio da ocupação encontrado neste cenário como um bom candidato para t, cujo valor é $1,05$ %. Fizemos o mesmo com os valores de TTL de 3 a 7, cujos resultados estão resumidos na Tabela 3.1.

Tabela 3.1. Possíveis valores para t

TTL	Pesquisas/Seg	Ocupação Média observada
3	50	1,84 %
4	25	1,05 %
5	20	2,06 %
6	15	1,67 %
7	15	2,80 %

Com os dados retirados da Tabela 3.1, calculamos a média e o intervalo de confiança usando a distribuição T-student. Com 95 % de confiança, encontramos a média $1,884$ e intervalo de confiança $\pm 0,88149$. Uma vez que $1,884$ está muito próximo de $0,0$, utilizamos o valor da média + IC = $2,76549$ como o valor limite t a fim de que,

como veremos posteriormente, ele admita que redes com ocupação média até t sejam consideradas não congestionadas.

3.3.2.2 A função $F(o)$

O objetivo da etapa de análise é fornecer conclusões à etapa de planejamento, para que esta agende mudanças no parâmetro controlado. Este controlador se baseia na ideia de melhoria da taxa de sucesso por evitar a formar de congestionamentos.

Possuindo uma estimativa da média da ocupação o e o valor limite de congestionamento t, queremos comparar o e t e inferir se há ou não congestionamento. Isso pode ser feito de uma forma muito simplista:

$$F(o) = \begin{cases} -1, & o \leq t \\ +1, & o > t \end{cases} \tag{3.3}$$

Utilizando a Função 3.3, teríamos um classificador linear, no entanto a descontinuidade dessa função nos traria problemas: a função apresentaria previsões completamente confiantes de 1 ou -1, mesmo para valores próximos a t, nosso limite. Assim, gostaríamos que $F(o)$ seja derivável e que, de certa forma, o módulo de $F(o)$ reflita a certeza da classificação, como sugerido por [Russell & Norvig, 2010] como solução ao problema do classificador binário com limites rígidos. A Função 3.4 descreve a forma desejada de $F(o)$.

$$y \mapsto F(o) \begin{cases} -1 \leq y < 0, & \text{se a rede não está congestionada} \\ 0 < y \leq +1, & \text{se a rede está congestionada} \\ |y| & \text{indica a certeza da avaliação} \end{cases} \tag{3.4}$$

Percebemos que a tarefa de $F(o)$ se parece muito com a de um *perceptron* com apenas uma entrada, que pode ser visto na Figura 3.3. Seguindo as definições matemáticas de [Russell & Norvig, 2010], um *perceptron* possui uma função de entrada e uma função de ativação. A função de entrada serve para combinar e atribuir pesos às múltiplas entradas. A função de ativação tipicamente é uma função não linear que fará de fato a distinção entre os casos desejados. Realizamos, assim, o seguinte esquema para geração da função $F(o)$. Propomos a seguinte função de entrada:

$$E(o) = \beta \times (o - \gamma) \tag{3.5}$$

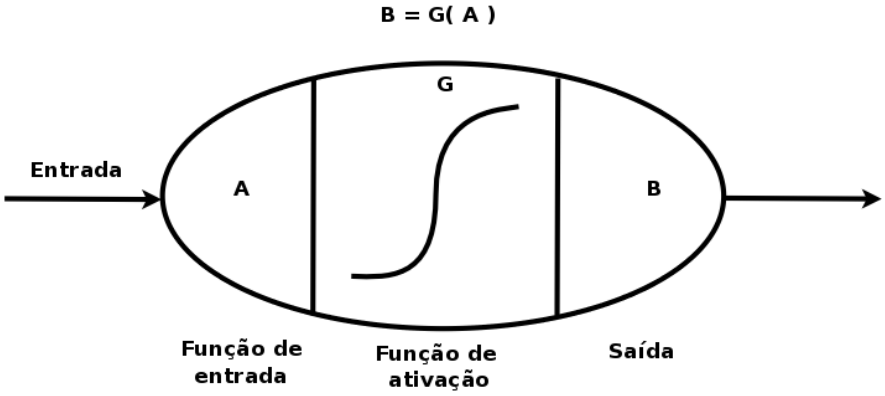

Figura 3.3. Modelo de um perceptron de uma entrada.

O parâmetro γ nos indica o valor limite da função de ativação, que separa os dois estados: congestionado e não congestionado. O valor *beta* nos indica quão rapidamente nos afastamos do valor limite γ, tornando a curva de resposta do *percepetron* mais ou menos "suave". Para a função de ativação, testamos as funções logística (Equação 3.6) e arcotangente (Equação 3.7), frequentemente utilizadas nas funções de ativação em redes neurais. Elas estão normalizadas no intervalo $[+1 : -1]$, de forma a cumprir as especificações da Equação 3.4.

$$G(x) = \frac{2}{1 + e^{-x}} - 1 \tag{3.6}$$

$$G(x) = \frac{arctan(x)}{\pi/2} \tag{3.7}$$

O *perceptron* fica sendo então a função $F(o)$:

$$F(o) = G(E(o)) \tag{3.8}$$

Como $E(\gamma) = 0$ e $G(0) = 0$, temos que $F(\gamma) = G(E(\gamma)) = 0$. De fato, caso $o = \gamma$ teremos o caso limite entre o congestionamento e não congestionamento. À medida que o se afasta de γ, o valor da função se aproxima de -1 ou $+1$ nos casos de não congestionamento e congestionamento respectivamente. Segundo análise descrita na Subseção 3.3.2.1, temos que $\gamma = 2,76549$, restando apenas definir o valor de β. Fizemos alguns experimentos para encontrar um valor de β razoável, bem como descobrir qual a melhor função de ativação. Os resultados mostraram que a função de ativação logística utilizando $\beta = 1$ apresentou os melhores resultados, sendo portanto utilizada.

3.3.3 Etapa "Planejar"

Nesta etapa, temos por objetivo planejar ações de acordo com a análise realizada, a fim de combater os problemas ou explorar as oportunidades encontradas. No caso específico deste controlador, duas ações distintas poderão ser tomadas: caso seja detectado um congestionamento, o controlador irá reduzir o TTL a fim de mitigar o congestionamento e caso não seja detectado congestionamento, ele aumentará o TTL a fim de explorar melhor os recursos disponíveis.

Para isso, utilizamos um mecanismo de controle de crescimento aditivo e decrescimento multiplicativo, ou AIMD. O AIMD reduz rapidamente o congestionamento devido ao decréscimo multiplicativo, enquanto o acréscimo aditivo permite um aumento conservador do parâmetro controlado. Aplicamos o algoritmo AIMD ao tempo de vida das mensagens de pesquisa (TTL), utilizando como entrada o resultado da função $f(o)$, descrita na Subseção 3.3.1.

A alteração planejada do TTL se dará da seguinte forma. Seja $TTL(i)$ o TTL em um período i. O novo TTL será modificado por um fator $\Delta(o,i)$, tal que:

$$TTL(i+1) = TTL(i) + \Delta(o,i) \tag{3.9}$$

$$\Delta(o,i) = |F(o)| \times \begin{cases} \delta, & \text{se } F(o) \leq 0 \\ (\alpha - 1) \times TTL(i), & \text{se } F(o) > 0 \end{cases} \tag{3.10}$$

Por sua vez $\Delta(o,i)$, definido na Equação 3.10, faz uso da Função $F(o)$ de duas formas. Primeiro, ele verifica o sinal de $F(o)$ para definir se irá aplicar o crescimento aditivo ao TTL ($F(o) \leq 0$) ou se irá aplicar o decréscimo multiplicativo ($F(o) > 0$). Segundo, ele utiliza o módulo de $F(o)$ como modificador na intensidade da alteração, uma vez que o módulo de $F(o)$ indica a certeza da análise, como visto na Equação 3.4.

Por fim, resta definir a função dos parâmetros δ e α. O parâmetro δ indica em quanto o TTL deve ser aumentado, caso seja detectado que não há congestionamento. O parâmetro α indica por qual número $\alpha < 1$ o TTL será multiplicado, caso seja detectado congestionamento. Note que, das Equações 3.9 e 3.10, pode-se perceber que o TTL será ajustado da seguinte forma, caso $F(o) = 1$:

$$TTL(i+1) = TTL(i) \times \alpha \tag{3.11}$$

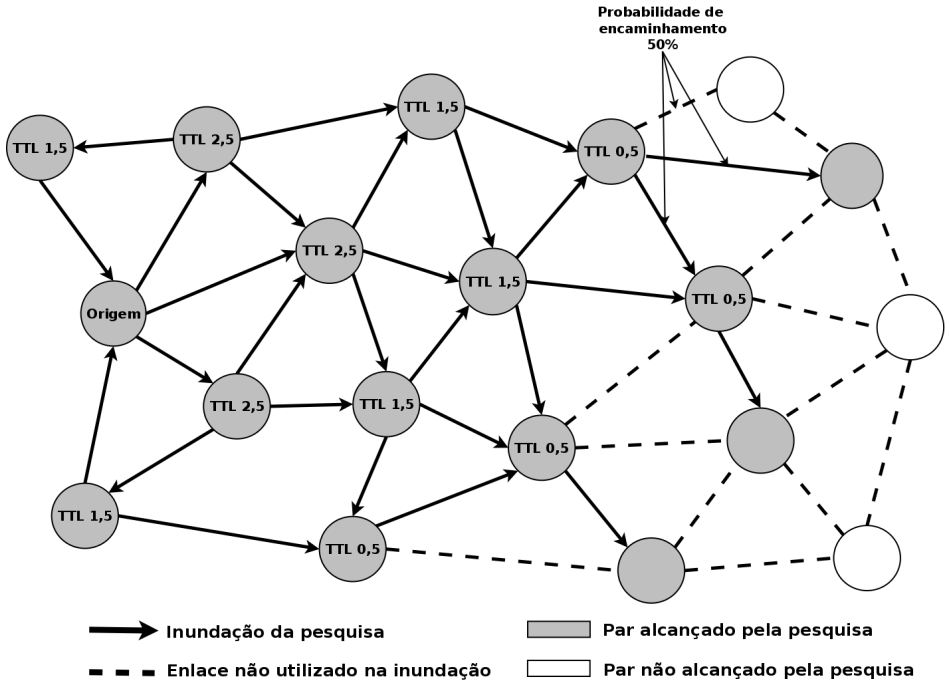

Figura 3.4. Funcionamento do TTL fracionário. No exemplo, o par "Origem" realiza uma pesquisa com $TTL = 3,5$. O processo de encaminhamento de TTL ocorre de forma usual para TTL>=1. Os pares com TTL=0,5 encaminham a pesquisa aos seus vizinhos com probabilidade de 50%

3.3.4 Etapa "Executar"

A etapa de execução é responsável pela execução das modificações planejadas na etapa anterior. Uma vez que o controlador é executado dentro de cada par, a execução implica na simples modificação do TTL. No entanto, note que pelas definições matemáticas acima encontraremos, frequentemente, valores de TTL fracionários como $3,5$. Para dar suporte a esses valores, de forma a termos uma melhor granularidade do parâmetro TTL, criamos a noção de TTL fracionário, que funciona da seguinte forma: cada nó avalia, ao receber a mensagem de pesquisa, se o TTL é maior do que 1. Se for, ele decrementa o TTL em uma unidade e repassa a pesquisa aos seus vizinhos. Quando a mensagem de pesquisa possuir TTL $t, 0 < t < 1$, ele repassa a mensagem aos seus vizinhos com probabilidade t. Uma ilustração deste processo pode ser vista na Figura 3.4

O controlador P-AIMD utiliza as quatro etapas descritas para autoconfigurar o parâmetro TTL da aplicação. Este laço de controle é executado com periodicidade p. Isto foi feito dessa forma pois uma vez que a etapa de monitoramento é executada de forma passiva através das mensagens *ping-pong*, tem-se sempre dados atuais para a análise e adaptação do TTL.

3.4 Controlador P-ML

Muitas vezes sabe-se descrever, em termos de métricas, uma situação desejável para a operação da rede, como baixa latência, filas pequenas e alta taxa de sucesso das pesquisa. Isso significa que se pode classificar diversos momentos da rede como situações congestionadas ou não congestionadas, em especial supondo conhecimento global da rede. No entanto, mapear métricas locais dos nós em uma dessas duas classes: congestionado / não congestionado, é uma tarefa um pouco mais desafiadora, e é dela que trata a etapa de análise.

Ao invés de abordar este problema diretamente, nesse controlador utilizamos técnicas de aprendizado de máquina para criar essa relação entre métricas locais e estado de congestionamento, tendo portanto um foco muito maior na etapa de análise. Implementamos também uma simples etapa de planejamento de mudanças a fim de ressaltar a eficácia da análise realizada.

3.4.1 Etapa "Monitorar"

A etapa de análise deste controlador utiliza diversas métricas locais. Em especial, as métricas de ocupação local e da vizinhança utilizadas no P-AIMD são utilizadas (Subseção 3.3.1). Ainda dentro das métricas locais, utilizamos a latência média das pesquisas realizadas. Para cada pesquisa realizada, medimos o período de tempo entre o início da pesquisa e o recebimento da primeira resposta. Armazenamos essas informações e tiramos a média das pesquisas realizadas nos últimos 30 segundos, a fim de evitar o uso de dados "antigos".

Outras métricas locais podem ser incorporadas e disponibilizadas para a etapa de análise, a fim de prover um modelo de aprendizagem mais robusto.

3.4.2 Etapa "Analisar"

O processo de análise dessa etapa é feito utilizando um classificador. Para utilizarmos um classificador, precisamos modelar a análise de congestionamento como um problema de classificação e levantar uma base de dados para seu treinamento. Uma vez construído, o classificador classifica os estados em "não congestionado" e "congestionado", numericamente representados pelos números -1 e $+1$, respectivamente.

3.4.2.1 Modelagem

A ideia de utilizar um classificador para a análise vem do fato de que se pode, com facilidade, classificar *a posteriori* um cenário como congestionado - principalmente supondo conhecimento global. Pode-se afirmar que há correlação entre dados locais acessíveis a cada par, como latência média das pesquisas e ocupação das filas dos vizinhos, e o congestionamento da rede; muito embora seja extremamente difícil especificar qual seja esta relação. Dado este cenário, técnicas modernas de classificação são muito úteis, visto que irão abstrair do homem a complexidade do relacionamento entre métricas.

Como utilizaremos um classificador *off-line* precisamos de um conjunto de tuplas que mapeiam os atributos em uma classe, de forma similar à Tabela 2.1. Levantamos algumas métricas relevantes disponíveis e chegamos ao seguinte conjunto de atributos.

- Ocupação local
- Ocupação média da vizinhança imediata
- Ocupação média da vizinhança de nível 2
- Latência média fim-a-fim
- TTL utilizado

Cabe ressaltar que estes são todos atributos locais, fornecidos pela etapa de monitoramento. Desejamos mapear um dado contexto da rede, indicado pelas métricas supracitadas, em uma classe que indique se há ou não congestionamento. Muito embora seja complexo identificar um congestionamento a partir de métricas locais, essa tarefa se torna um pouco mais fácil considerando métricas globais. Sabe-se, por exemplo, que quando há indícios de congestionamento, percebemos de forma generalizada um aumento no tempo das filas de roteamento (como explicado na Subseção 3.3.1). Utilizamos, então, a latência média "global" das pesquisas fim-a-fim como métrica que sugere a ocorrência de congestionamento.

Seguimos o seguinte critério: se a latência fim-a-fim média da rede for superior a t segundos (ida e volta da pesquisa), então há um congestionamento - caso contrário a rede não está congestionada. Este valor t é o que irá separar um estado de congestionamento e não congestionamento. Quando a rede está operando abaixo de sua capacidade ela tipicamente opera com latência bem abaixo desse limite, como de 0.1 a 0.3 ms, e quando há formação de congestionamento ela passa para patamares elevados, como de 7 a 9 s. Precisamos encontrar um t que explore bem a capacidade da rede mas que, de forma alguma, permita a formação de congestionamentos. Utilizamos

os valores $t \in \{1, 2, 3\}$ e avaliamos os resultados do uso de diferentes parâmetros na Subseção 4.3.2.

3.4.2.2 Base de dados

Devido à novidade da proposta, não encontramos base de dados compatível com nossa proposta. Optamos, então, por gerar uma base de dados através de medições durante simulações. Utilizamos o simulador NS-2 para simular a operação de diferentes redes, a fim de obter tuplas conforme as descritas na Sub-subseção 3.4.2.1. Todas as simulações utilizaram o cenário padrão, definido na Subseção 4.1.1, variando os seguintes parâmetros:

- Carga da rede: $\{9, 13.5, 18, 22.5, 27, 31.5, 36\}$ pesquisas por segundo
- Dinamicidade: $\{0\%, 50\%\}$ dos nós participantes entram e saem da rede em momentos aleatórios.
- TTL utilizado pelo par medido: $\{2, 3, 4, 5, 6, 7\}$
- TTL utilizado pelos demais pares: $\{2, 3, 4, 5, 6, 7\}$

Em cada simulação, um par foi escolhido para realizar as medições. Seu TTL era diferente configurado à parte dos demais, uma vez que na execução não haverá garantias de que todos usarão TTL similar. A geração das tuplas foi gerada por esse par de 5 em 5 segundos, do instante 100 s ao instante 450 s. Em cada simulação, escrevemos em cada linha de um arquivo os atributos da Sub-subseção 3.4.2.1 observados pelo par e a latência média fim a fim da rede. Posteriormente, utilizamos um script simples para transformar a latência média fim a fim da rede nas classes "congestionado" e "não-congestionado", além formatar o arquivo para ser utilizado no treinamento do classificador. Após filtrar dados em multiplicidade, nossa base de dados apresentou quase 1 milhão e 200 mil tuplas.

3.4.2.3 Classificador

Devido aos inúmeros classificadores disponíveis na literatura, optamos por uma maneira prática de avaliarmos vários desses de forma sistemática. O pacote de Software WEKA ([Hall et al., 2009]) possui uma série de algoritmos de aprendizagem de máquinas implementados com uma interface única, em especial algoritmos de classificação. Através dele, conseguimos testar a capacidade de generalização de diversos classificadores da nossa base de dados - bem como a viabilidade de sua aplicação em uma base de dados tão extensa.

O classificador que apresentou bons resultados, conforme métrica de avaliação descrita na Seção 2.3.1, foi o REPTree [Quinlan, 1992]. Utilizando a técnica de *cross-validation* com 10 dobras, obtivemos um classificador com acurácia das classificações de $87,4155\%$. O tempo médio levado para construir a árvore de decisão foi de 140.7 s, mostrando que além de produzir bons resultados, estes são gerados em tempo hábil.

Para a utilização do classificador nas simulações, encontramos algumas dificuldades técnicas. O principal objetivo de utilizar o WEKA é abstrair a complexidade de implementação dos diversos classificadores disponíveis. No entanto, o WEKA é um pacote implementado em Java enquanto o simulador de redes NS-2 é programado em C++ e TCL. A fim de evitar a custosa implementação dos algoritmos de classificação do WEKA em C++, implementamos um modelo simples de invocação remota de métodos no Java.

Implementamos um programa servidor java que "escuta" em uma porta específica. Sempre que uma máquina se conecta nessa porta, ele cria uma *thread* com um método que recebe por socket os atributos de classificação e retorna o resultado da classificação. Uma vez que o servidor e suas *threads* são implementados em Java, pode-se facilmente acessar a interface programável do WEKA, utilizado para efetuar as classificações. Parte da inicialização do servidor é o longo treinamento da base de dados, a fim de conseguir responder rapidamente as requisições recebidas após o término da inicialização. Implementamos também uma classe em C++ responsável pelo gerenciamento da comunicação via socket e realização das classificações requisitadas pelo controlador implementado no NS-2.

3.4.3 Etapa "Planejar"

A etapa de planejamento deste algoritmo ocorre de forma bastante simplificada. Embasados na observação de que o pico de desempenho da rede ocorre próximo do congestionamento da rede, propusemos três possíveis ações:

Com base nessas duas indicações, implementamos um mecanismo de modificação que segue os seguintes critérios

- `Manter` o TTL inalterado. Este caso deve ocorrer quando a rede está operando em regime de capacidade máxima, próximo de um estado de congestionamento.

- `Aumentar` o TTL. Esta ação deve ser ativada quando a rede estiver operando abaixo do limite de congestionamento e quando houver indícios que aumentar o TTL também não provocará congestionamento.

- `Diminuir` o TTL. Esta ação deve ser tomada quando o controlador identifica que a rede está congestionada, provocando perda de desempenho.

Utilizando o classificador da etapa de análise, desenvolvemos um mecanismo que ativa estas ações. Seja o Classificador C descrito na Sub-subseção 3.4.2.3 uma função $C(p, \tau)$ com parâmetros de contexto p (métricas de rede) e o TTL τ. Sejam também as classe "não congestionado" e "congestionado" os valores -1 e 1, respectivamente. Assim, definimos a função do Classificador como sendo:

$$C(p, \tau) \mapsto \{-1, 1\}$$

Seja $\delta \in 0, 1$ um parâmetro do controlador, τ o TTL atual e p os parâmetros atuais. Utilizamos o classificador para computar o estado de congestionamento $C(p, \tau)$ e $C(p, \tau + \delta)$. Para cada um dos quatro possíveis resultados, uma das três ações supracitadas é tomada, como pode ser visto na Tabela 3.2.

Tabela 3.2. Ações planejadas com base nas saídas do classificador

		$C(p, \tau)$	
		-1	1
$C(p, \tau + \delta)$	-1	`Aumentar`	`Diminuir`
	1	`Manter`	`Diminuir`

A construção desta tabela segue a seguinte intuição: Se a rede atual $C(p, \tau)$ for classificada como não congestionada, ela pode estar abaixo de sua capacidade de operação ou no limite. Caso aumentar o TTL, modelado por $C(p, \tau + \delta)$, nos dê indícios de congestionamento então a rede parece estar operando próximo ao limite de congestionamento, sendo que `Manter` o TTL parece ser a melhor ação. Caso o aumento do TTL $C(p, \tau + \delta)$ não dê indícios de congestionamento, então a rede está operando abaixo de sua capacidade e `Aumentar` é a ação ideal. Por fim, se identificarmos que a rede atual está congestionada, sempre a melhor ação é `Diminuir`, a fim de mitigar o congestionamento presente[1].

3.4.4 Etapa "Executar"

Foi utilizada a mesma estratégia de execução do algoritmo AIMD, como descrito na Subseção 3.3.4.

[1] O estado $C(p, \tau) = 1, C(p, \tau + \delta) = -1$ intuitivamente não deve ocorrer na prática, no entanto configuramos sua ação para `Diminuir`.

3.5 Conclusão

Nesse capítulos, apresentamos os detalhes dos dois protocolos propostos para a auto-configuração de parâmetros. Inicialmente, mostramos o framework MAPE, base para ambos e depois detalhamos como cada etapa dos dois protocolos foi implementada. A seguir, descrevemos a metodologia para a validação da proposta, as simulações realizadas e é feita uma análise dos resultados.

Capítulo 4

Avaliação de Desempenho

Neste Capítulo, trataremos da avaliação do desempenho dos algoritmos propostos. Mostraremos a metodologia de validação dos resultados utilizada, uma descrição das métricas de avaliação seguido da análise dos resultados. Por fim, apresentamos breves conclusões dos resultados obtidos.

4.1 Metodologia

Utilizamos simulações a fim de avaliar a validade das soluções propostas. O NS-2 (*Network Simulator - 2* [McCanne et al., 1997]) é um simulador de redes extensivamente utilizado pela comunidade acadêmica. Ele simula de forma bastante abrangente os diversos eventos que uma rede móvel ad hoc apresenta como mobilidade dos nós, atenuação do sinal e colisão de pacotes. Descrevemos abaixo o cenário base utilizado por todas as simulações deste trabalho, seguido das métricas de desempenho consideradas.

4.1.1 Cenário padrão

Cada cenário foi avaliado 30 vezes e apresentamos a média com 95 % de Intervalo de confiança. A carga de trabalho simula um compartilhamento de arquivos urbano, onde dispositivos Wi-Fi formam uma rede P2P sobre uma MANET. Os protocolos Par-a-Par são implementados sobre o protocolo UDP, uma vez que o TCP não apresenta bom desempenho neste tipo de cenário, como apontado por [Sundaresan et al., 2005; Katabi et al., 2002].

Para roteamento, escolhemos o protocolo OLSR [Jacquet et al., 1998], um dos diversos padrões possíveis para redes *mesh* [ols, 2010]. Haerri et al., realizaram uma

Tabela 4.1. Configuração do cenário base de simulação

Parâmetro	**Valor**
Protocolo MAC	IEE 802.11b
Número de nós	50
Área	1 km^2
Percentual de nós Permanentemente on-line	50%
Número de Arquivos	250 arquivos diferentes
Número de réplicas por aquivoarquivo	5
Erros do canal de comunicação	0%

comparação de desempenho entre AODV e OLSR num contexto de VANETs e o protocolo de roteamento OLSR apresentou melhores resultados [Haerri et al., 2006].

Os nós usam radiosrádios IEEE 802.11b, modelados a partir da especificação da placa Cisco Aironet 350 [Cisco Systems, 2005], com potência de transmissão de 10dBm. Nesta configuração, o rádio consume 1,6887W no modo de transmissão, 0,6699W em modo ocioso e 1,0791W em modo recepção. A fila na cama de roteamento é de 30 pacotes.

No cenário padrão, assumimos uma rede de 50 pares em uma área de $1200m \times 1200m$ seguindo uma distribuição uniforme. Nosso modelo de mobilidade é o *restricted random walk*, onde os pares caminham aleatóriamentealeatoriamente sobre um grafo que modela as ruas de uma parte da cidade de Houston [Saha & Johnson, 2004]. Os pares caminham apenas nas ruas e a velocidade dos pares segue uma distribuição uniforme, dentro do limite de velocidade de cada rua. Utilizamos a implementação de Le Boudec et al do Random Walk sobre grafos, uma vez que provou-sese provou que provê uma convergência rápida a um estado estacionário [Boudec & Vojnovic, 2006]. Os dois parâmetros desta implementação – o tempo médio de pausa e seu desvio padrão – foram definidos como 10 s e 2 s, respectivamente.

Além disso, 50 % dos nós sempre fazem parte da rede Par-a-Par, enquanto os demais ingressam à rede em algum momento e saem após algum tempo. Os tempos de entrada e saída na rede seguem uma distribuição uniforme. Cada par provê 5 arquivos diferentes, havendo por isso 250 arquivos diferentes na rede. Cada arquivo, em média, possui 5 réplicas aleatoriamente distribuídas entre os pares. Cada simulação possui 550 segundos de duração, e as pesquisas P2P ocorrem seguindo uma distribuição uniforme entre os segundos 50 e 500. O arquivo pesquisado é escolhido aleatóriamente-aleatoriamente dentre os 250 disponíveis. O número padrão de vizinhos em uma rede Gnutella foi ajustado para 3. No cenário padrão, não consideramos perdas devido a erros no canal. As principais configurações do cenário base aparecem sumarizadas na Tabela 4.1.

4.1.2 Métricas

Definimos aqui as principais métricas de desempenho para avaliar a validade dos algoritmos propostos. Uma das propostas é a utilização de um classificador para realizar a etapa de análise do controlador P-ML. Testamos diversos classificadores, conforme descrito na Subseção 3.4.2.3 e a métrica de desempenho utilizada para escolher o melhor dentre estes é o percentual de instâncias corretamente classificada. Utilizamos a metodologia de validação cruzada, com 10 dobras, a fim de avaliar quão bem o classificador generaliza para um conjunto independente de testes.

A técnica de validação cruzada (*cross-validation*) funciona da seguinte forma: Divida a base de dados D em n partes disjuntas e de tamanho similar $D_1, D_2, \ldots, D_n$. Crie n classificadores $C_1, C_2, \ldots, C_n$, tal que o classificador C_i utilize o conjunto D_i para testes, e $D - D_i$ para treinamento. Seja o $N(D)$ o número de instâncias em um conjunto D, e $C(D)$ o número de instâncias corretamente classificadas do classificador C no conjunto de testes D. O percentual de instâncias corretamente classificadas é calculado da seguinte forma:

$$\frac{\sum_{i=1}^{n} C_i(D_i)}{\sum_{i=1}^{n} N(D_i)}$$

Para avaliação dos controladores, analisaremos em especial três métricas. A "taxa de sucesso" das pesquisas corresponde ao percentual das pesquisas P2P que obtiveram sucesso na descoberta do conteúdo. Note que, para fins de simulação, uma pesquisa só é realizada se pelo menos um par da rede possuir, no momento da pesquisa, uma cópia do arquivo pesquisado. Um alto valor da "taxa de sucesso" é muito desejado, uma vez que esta métrica reflete o sucesso na aquisição de informações possivelmente vitais ao usuário.

Outra métrica relevante é a "latência" da aplicação. Esta métrica é calculada comparando o tempo da requisição da pesquisa pelo usuário com o tempo de recebimento do primeiro resultado positivo. De forma objetiva, ela serve como métrica de qualidade para o usuário, que prefere receber os resultados rapidamente. Por outro lado, essa métrica reflete o congestionamento da rede - uma vez que atrasos nas requisições são originadas pelo conflito no acesso ao meio sem fio, bem como por longas filas de roteamento.

Por fim, a métrica de energia é sempre considerada no contexto de aplicações móveis. Ela não é crítica como no caso de redes de sensores sem fio, visto que os dispositivos considerados tipicamente possuem forma de serem recarregados. No entanto, alto consumo de energia é indesejável por abreviar o tempo contínuo de uso da rede.

4.2 Configurações Manuais

Nossas simulações modelam um cenário de troca de arquivo em ambiente urbano. Avaliamos o desempenho de diferentes configurações de TTL fixo, a fim de validarmos a necessidade de adaptação. Na Figura 4.1 vemos a comparação da taxa de sucesso da utilização de TTLs fixos. Perceba que nos cenários com baixo número de pesquisas/s, as configurações com maior TTL apresentam melhor taxa de sucesso, pois exploram melhor a capacidade da rede. Isto ocorre porque o uso de um TTL alto evita que pesquisas falhem por erros aleatórios. Havendo baixo número de pesquisas por segundo, não ocorre ainda formação de congestionamento, evidenciado pela baixa latência vista na Figura 4.2. No entanto, para uma demanda maior as configurações de alto TTL rapidamente perdem desempenho, tornando-se opções com resultados piores em termos de taxa de sucesso e latência.

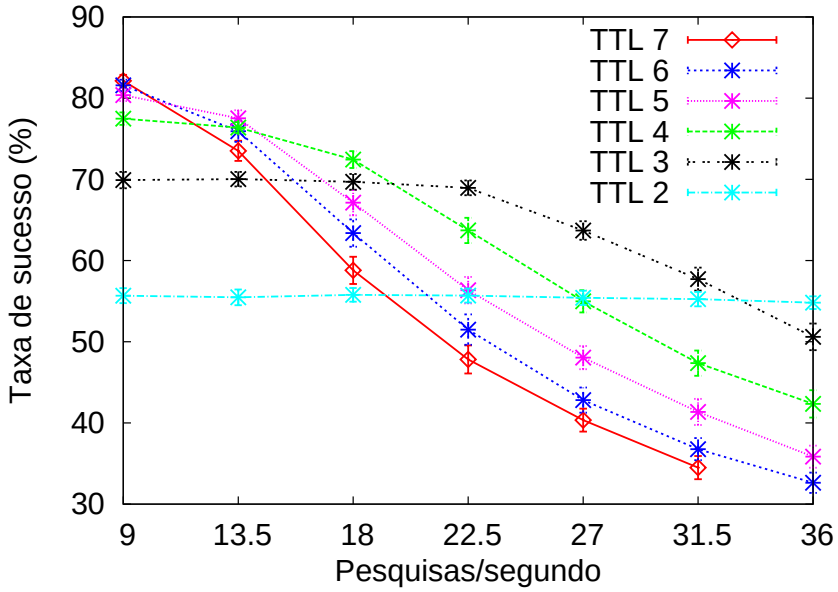

Figura 4.1. Comportamento da taxa de sucesso de diferentes configurações de TTL sobre demanda variada

No cenário de 9 pesquisas/s, a configuração TTL 7 apresenta taxa de sucesso de 82,7 %, sendo a configuração com melhor desempenho nesse cenário. No entanto, para 13.5 pesquisas/s sua taxa de sucesso é apenas de 73,51 %, inferior ao desempenho do TTL 5, melhor taxa de sucesso deste cenário 77,51 %. Essa inversão na melhor configuração ocorre pela formação de congestionamentos, evidenciada pelo notável aumento da latência do TTL 7, de 167,8 ms para 974 ms nos cenários de 9 e 13,5 pesquisas/s, respectivamente.

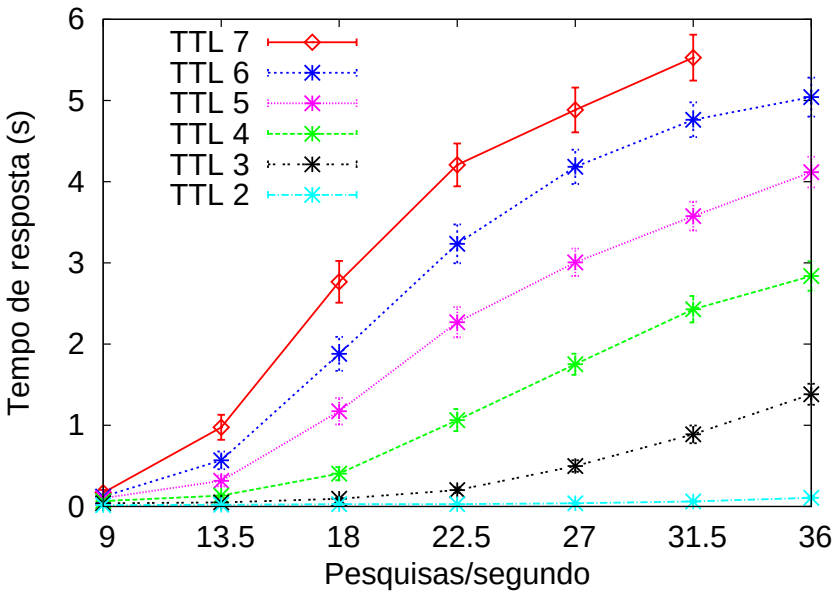

Figura 4.2. Tempo de resposta de diferentes configurações de TTL sobre demanda variada

Tabela 4.2. Melhores configurações manuais do TTL

Pesquisas / segundo	Melhor TTL
9	7
13,5	5
18	4
22,5	3
27	3
31,5	3
36	2

O que se observa na utilização de configurações manuais é um compromisso entre abrangência da pesquisa e carga gerada. Utilizando valores altos de TTL forçamos o recebimento da pesquisa pela maioria dos pares, mas geramos maior número de mensagens. Se esse número de mensagens ultrapassar um certocerto limite, começam a se formar filas nos pares e muitas colisões de pacotes, degradando a qualidade do serviço. Na Tabela 4.2 vemos a melhor configuração manual do TTL para cada cenário. A grande variação do TTL "ideal" entre cenários ressalta a necessidade da utilização de um algoritmo de auto-configuraçãoautoconfiguração do TTL.

4.3 Escolha de parâmetros

As soluções propostas vemAs soluções propostas vêm para resolver o problema da configuração de parâmetros de aplicações par-a-par aplicadas a redes ad hoc. No entanto, estes mesmos algoritmos também possuem parâmetros que também precisam ser configurados. Faremos agora uma análise do efeito da modificações destes parâmetros no desempenho das soluções propostas.

4.3.1 P-AIMD

O controlador P-AIMD possui os seguintes parâmetros a serem configurados:

- Função de ativação do perceptron: Função logística (Equação 3.6) ou função arcotangente (Equação 3.7). Utilizamos a função logística como base para nossos estudos.
- Valor β: Valor responsável pela sensibilidade da função de saída com relação à diferença da medida e o valor γ. Utilizamos o valor 1 em todos os experimentos.
- Valor do TTL inicial: Ajustado para 4 em todos os experimentos.
- Valor limite γ: Utilizamos $\gamma = 2,76549$ em todos os experimentos, conforme descrição na Sub-subseção 3.3.2.2.
- Valor α: Multiplicando no decréscimo multiplicativo do AIMD
- Valor δ: Soma no acréscimo aditivo do AIMD

4.3.1.1 Parâmetros α e δ

Como visto na Subseção 3.3.3, os parâmetros α e δ regulam a dinamicidade de adaptação do protocolo. Se o controlador toma sempre medidas muito enérgicas, o protocolo não estabilizará o parâmetro e haverá perda de desempenho pela oscilação de valores. Por outro lado, ele pode se tornar ineficiente caso sempre atue forma branda.

O parâmetro $\alpha \in [0, 1]$ indica quão intensamente deve ser diminuído o TTL caso seja detectado indícios de congestionamentos pela etapa de análise. Na Figura 4.3 vemos que quanto menor o valor de α, menor o tempo de resposta da aplicação. Isso ocorre porque menores valores de α implicam em uma maior redução no TTL quando um congestionamento é detectado.

No entanto, valores menores de α podem limitar o alcance das pesquisas, possivelmente não encontrando o recurso desejado. Vemos na Figura 4.4 que a taxa de

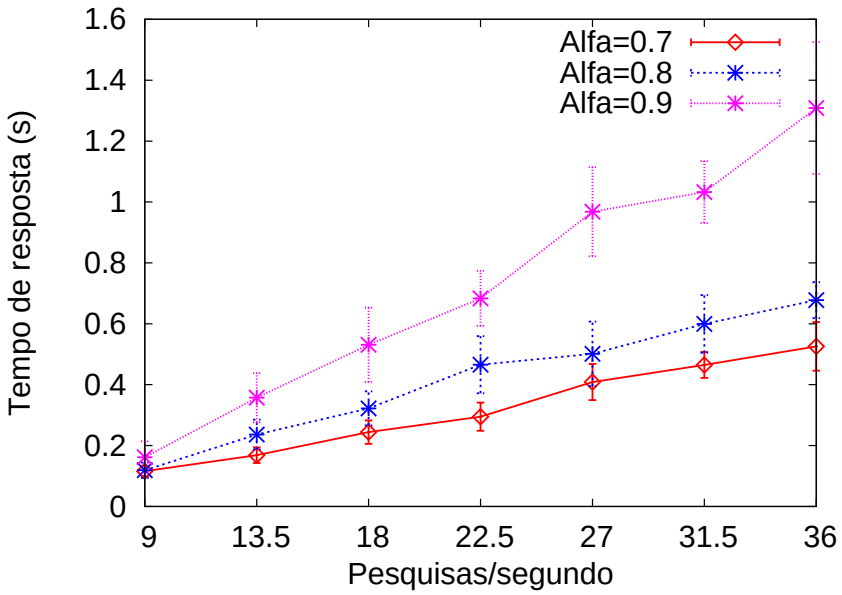

Figura 4.3. Tempo de resposta ao utilizar diferentes valores para α/. $\delta = 0.1$.

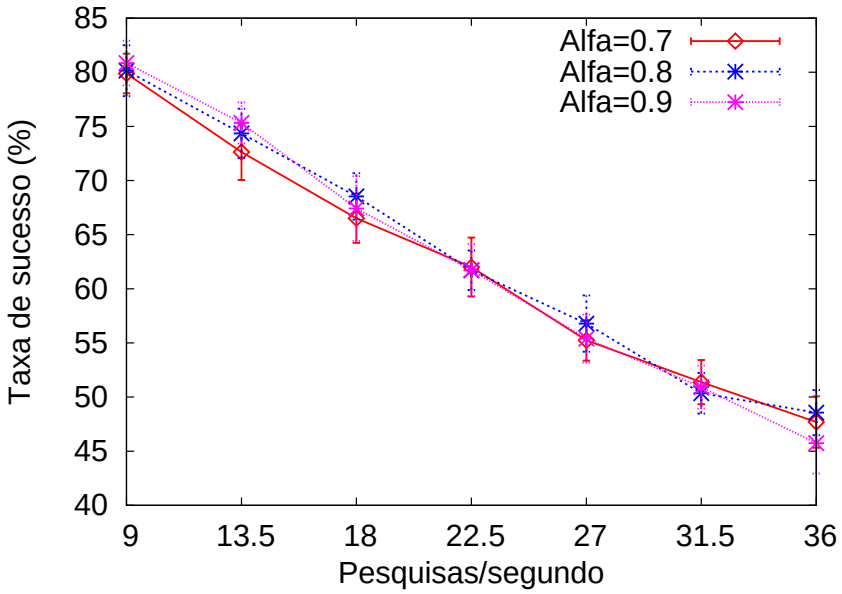

Figura 4.4. Taxa de sucesso ao utilizar diferentes valores para α. $\delta = 0.1$.

sucesso de $\alpha = 0.7$ é muito próxima da taxa de sucesso de $\alpha = 0.8$, muito embora a o tempo de resposta do primeiro seja muito menor. Isto significa que enquanto algumas pesquisas sem sucesso com $\alpha = 0.8$ falharam por colisões enquanto outras falharam por insuficienciainsuficiência de TTL com $\alpha = 0.7$. Neste cenário, notavelmente $\alpha = 0.9$ apresenta o pior resultado nas duas métricas, indicando uma adaptação pouco efetiva.

Vemos na Figura 4.5 o impacto na taxa de sucesso da variação do δ. A utilização

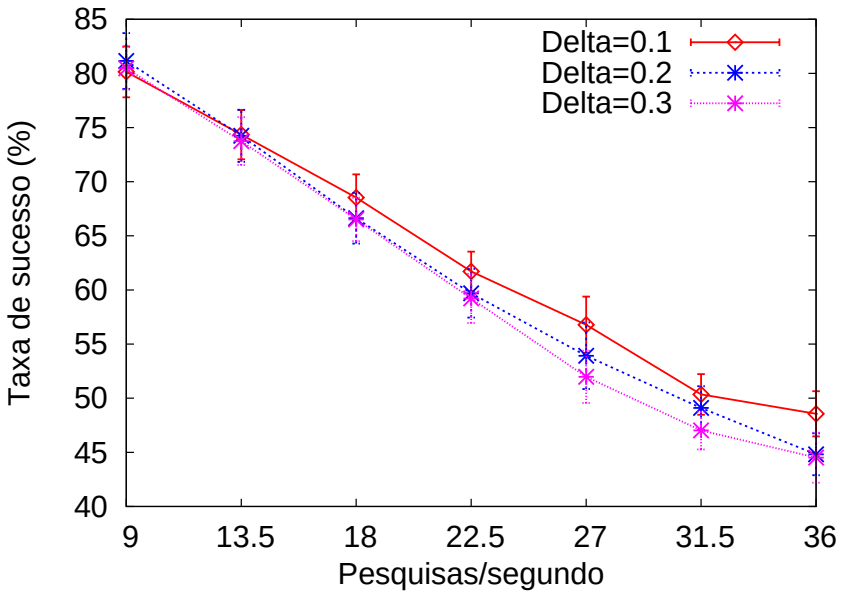

Figura 4.5. Taxa de sucesso ao utilizar diferentes valores para δ. $\alpha = 0.8$.

de diferentes valores de δ leve àa resultados similares nos cenários com baixa carga da rede. No entanto, nos cenários onde há maior probabilidade de congestionamento valores mais altos de δ impactam negativamente na taxa de sucesso da aplicação. Isso pode ser visto na Figura 4.6, onde vemos que a diferença do tempo de resposta entre $\delta = 0.1$ e $\delta = 0.3$ é por volta de $1s$ nos cenários de maior carga.

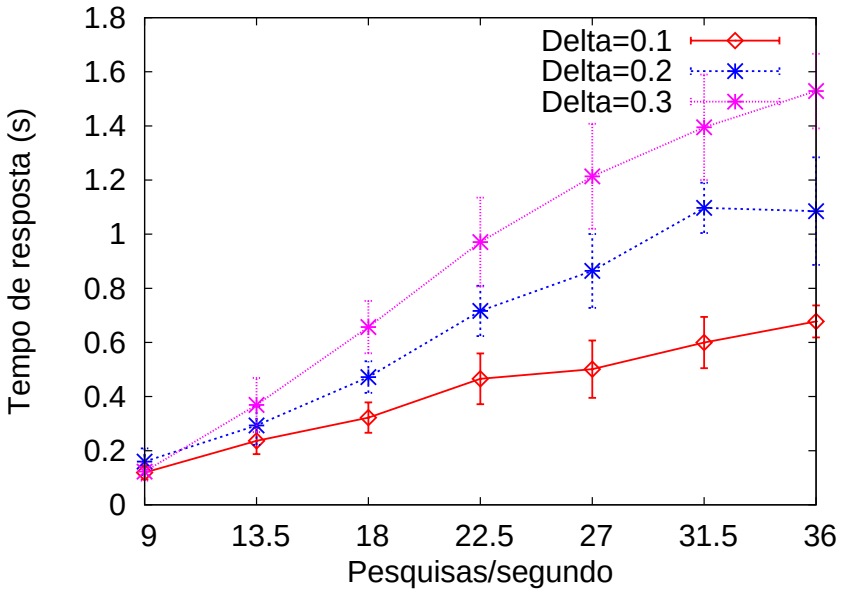

Figura 4.6. Tempo de resposta ao utilizar diferentes valores para δ. $\alpha = 0.8$.

Os parâmetros escolhidos para utilização no P-AIMD foram $\alpha = 0.8$ e $\delta = 0.1$. Os resultados da comparação de parâmetros mostram que valores mais conservadores de crescimento do TTL δ são recomendáveis. Para a escolha de α, tanto $\alpha = 0.7$ quanto $\alpha = 0.8$ apresentaram resultados comparáveis, indicando que este é um parâmetro menos sensível. A utilização de $\alpha = 0.9$ não reduz o TTL de forma efetiva, resultando em aumento do tempo de respotaresposta pela formação de congestionamentos, não sendo por isso valor recomendável.

4.3.2 P-ML

O controlador P-ML possui os seguintes parâmetros:

- Valor do TTL inicial: Ajustado para 4 em todos os experimentos.
- Valor limite de latência global fim-a-fim T: O valor T é utilizado na geração da base de dados para o classificador, como referência para as situações de congestionamento e não congestionamento. Em nossas avaliações, experimentamos $T \in \{1, 2, 3\}$ s.
- O parâmetro δ: este parâmetro é o módulo das modificações realizadas pelo controlador na etapa de execução. Avaliamos $\delta \in \{0.25, 0.50, 0.75, 1.00\}$.

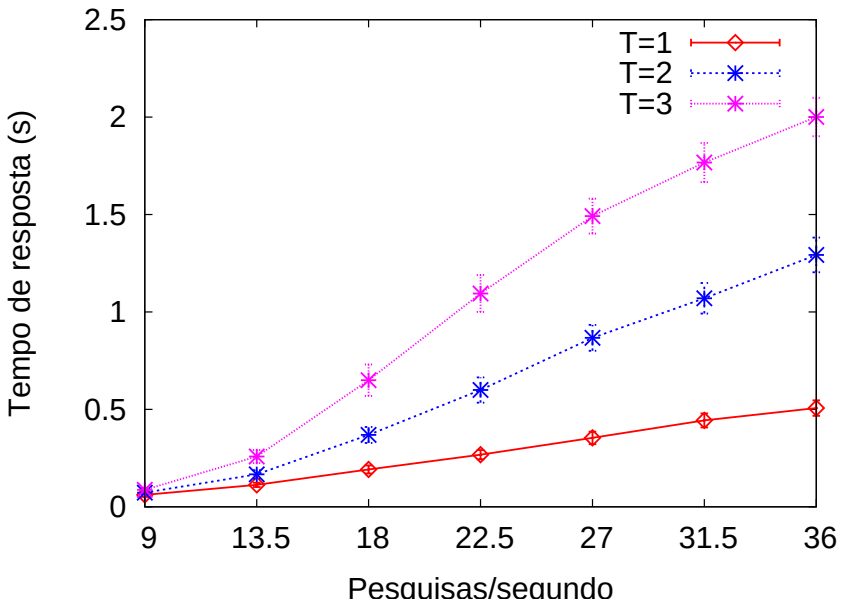

Figura 4.7. Tempo de resposta ao utilizar diferentes valores para T. $\delta = 0.25$.

Avaliamos as diferentes combinações dos parâmetros T e δ. Na Figura 4.7 comparamos diferentes valores de T. Pela definição, consideramos que a formação de congestionamento ocorre quando o tempo de resposta médio da aplicação é maior ou igual a T. No cenário de 36 pesquisas/s vemos que o tempo de resposta médio é 0.56, 1.29 e 2.00 s para $T = 1$, 2 e 3 s respectivamente. Isso indica que o algoritmo de adaptação de fato controla o congestionamento segundo a definição utilizada, mantendo o tempo de resposta médio da aplicação menor do que T.

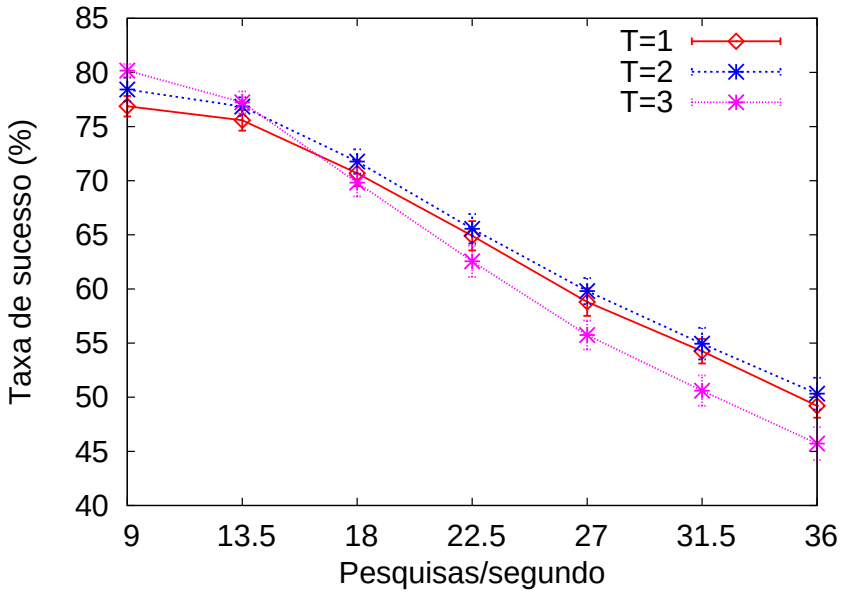

Figura 4.8. Taxa de sucesso ao utilizar diferentes valores para T. $\delta = 0.25$.

Ao utilizarmos $T = 3$ s como definição de congestionamento, estamos permitindo uma maior exploração no uso de recursos da rede ao mesmo tempo queao mesmo tempo em que toleramos a formação de pequenos congestionamentos. Isso fica evidente na Figura 4.8, onde vemos que a maior exploração de recursos provê uma pequena melhoria de desempenhos no cenário de 9 pesquisas/s e menor taxa de sucesso nos cenários com mais do que 18 pesquisas/s devido à perda de pacotes por congestionamento.

Como esperado, vemos que $T = 2$ s apresenta maior taxa de sucesso do que $T = 1$ s nos cenários de baixa carga. No entanto1, isso também acontece nos cenários de carga elevada. Isso indica que $T = 2$ s é um bom parâmetro de definição de congestionamento, que permite a exploração de recursos da rede evitando, contudo, congestionamentos mais sérios.

Ao variarmos δ, analisamos o impacto do módulo das modificações no TTL no desempenho do controlador. A diferença na taxa de sucesso dos quatro valores avaliados é muito pequena, como pode ser vista na Figura 4.9. De forma geral, aparentemente o

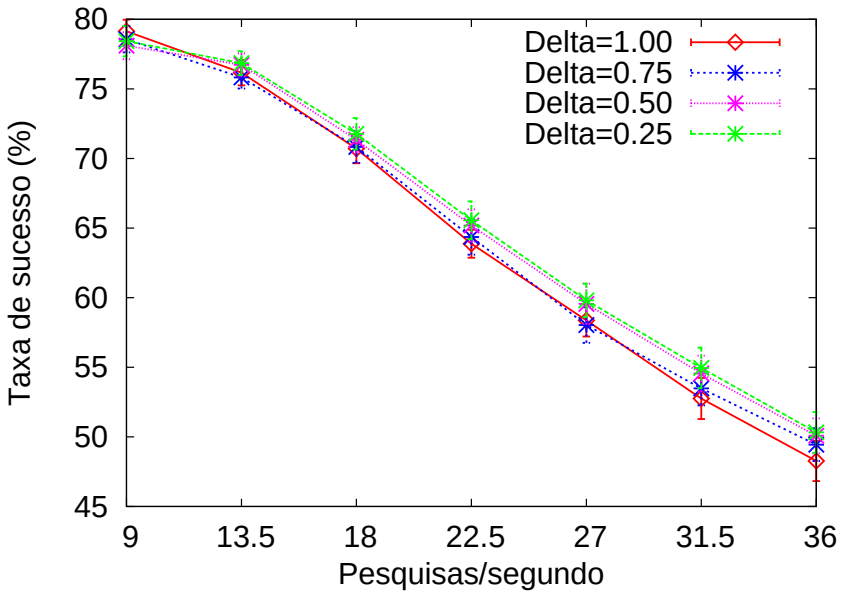

Figura 4.9. Taxa de sucesso ao utilizar diferentes valores para δ. $T = 2$.

uso de menores valores de δ favorece um controle mais fino da adaptação pelo controlador, levando a resultados marginalmente melhores.

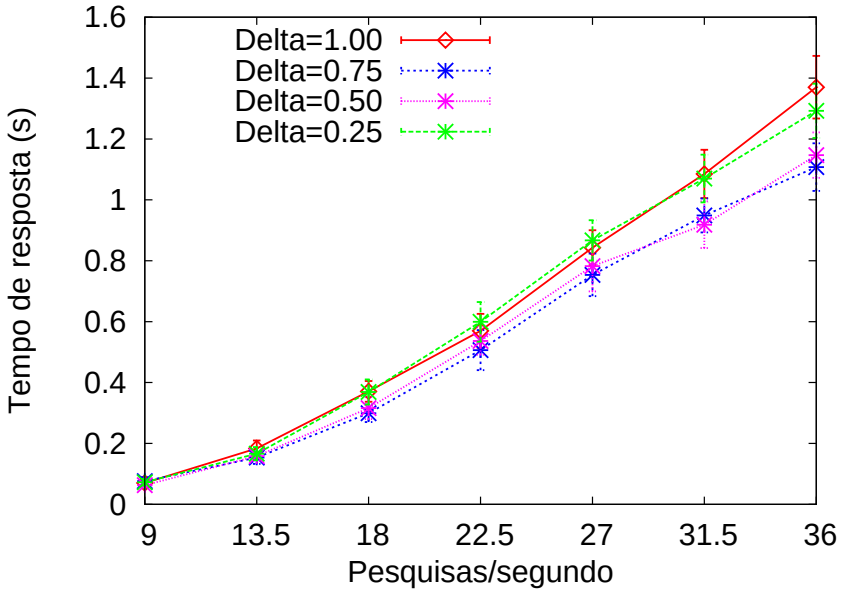

Figura 4.10. Tempo de resposta ao utilizar diferentes valores para δ. $T = 2$.

O impacto da variação do δ no tempo de resposta pode ser visto na Figura 4.10. Os valores $\delta = 1.00\ s$ e $\delta = 0.25\ s$ apresentaram tempo de resposta ligeiramente superior aos demais, mostrando que exploram mais os recursos da rede. No cenário

de 36 pesquisas/s, ainda que $\delta = 1.00\ s$ e $\delta = 0.25$ apresentaram tempo de resposta similares, $\delta = 1.00\ s$ obteve a pior e $\delta = 0.25$ a melhor taxa de sucesso. Isso indica que e exploração de recursos é feita de forma muito mais eficaz com $\delta = 0.25$, uma vez que as alterações mais conservadoras no TTL evitam desperdício de recursos.

Os parâmetros escolhidos para o P-ML foram $T = 2\ s$ e $\delta = 0.25$. Os resultados da comparação de parâmetros mostram que o uso de valores mais conservadores de crescimento do TTL δ são recomendáveis, de forma similar ao observado no P-AIMD. O uso de $T = 1\ s$ e $T = 2\ s$ provê resultados similares, sendo que $T = 2\ s$ apresenta taxa de sucesso um pouco superior em todos os cenários. $T = 3\ s$ tolera a formação de congestionamentos prejudiciais ao funcionamento da aplicação, não sendo então uma configuração recomendável.

4.4 Auto-configuraçãoAutoconfiguração do TTL

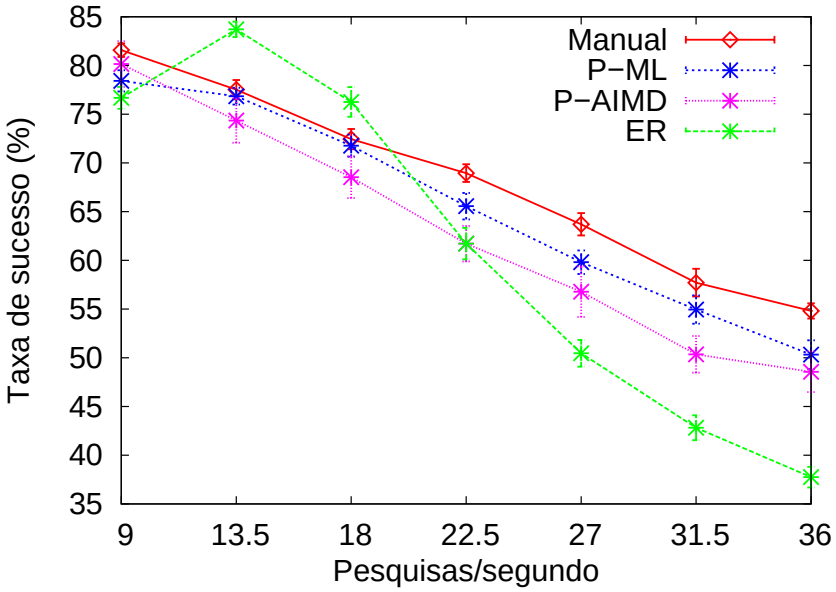

Figura 4.11. Taxa de sucesso das soluções propostas

Analisamos aqui o resultado dos algoritmos de configuração do TTL propostos P-ML e P-AIMD. Também mostramos os resultados do protocolo *Expanding Rings*, descrito na Seção 2.2. O *Expanding Rings* é uma variação do Gnutella que adapta de forma dinâmica o TTL utilizado, a fim de evitar a geração de carga desnecessária. Cabe ressaltar que ele é utilizado com sucesso em redes cabeadas, no entanto, não há estudo de seu funcionamento em MANETs. Como métrica de comparação, aprensen-

tamosapresentamos também o desempenho da melhor configuração fixa de TTL, sob o rótulo "Manual". O valor do TTL em cada cenário pode ser encontrado na Tabela 4.2.

Na Figura 4.11 observamos a taxa de sucesso do diversos protocolos. A solução "Manual" apresenta, como esperado, o melhor resultado em quase todos os cenários. A solução "Manual" apenas mostra a potencialidade de otimização de cada cenário, pois é uma solução impraticável. Surpreendentemente, o protocolo *Expanding Rings* apresenta desempenho, em termos da taxa de sucesso, superior à configuração manual nos cenários de 13,5 e 18 pesquisas / segundo. Isso é possível porque o *Expanding Rings* explora de forma muito boa a disponibilidade de réplicas, em especial se poucas pesquisas são realizadas simultâneamentesimultaneamente.

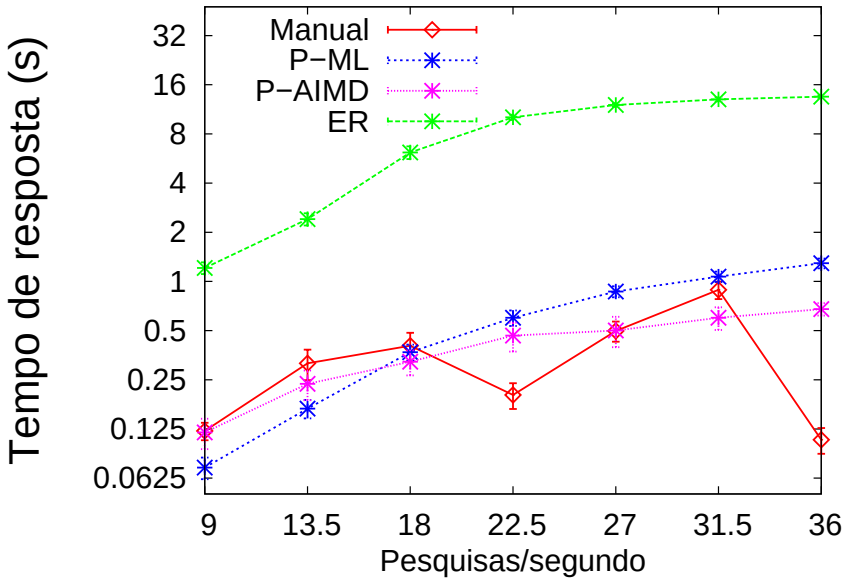

Figura 4.12. Tempo de resposta médio da aplicação das soluções propostas

No entanto, quando mais pesquisas são realizadas simultâneamentesimultaneamente seu desempenho cai drasticamente, aprensetandoapresentando o pior resultado nos cenários de com mais do que 22,5 pesquisas / segundo. Ao observar uma falha, o *Expanding rings* refaz a pesquisa com TTL superior até um TTL limite, por atribuir a falha na pesquisa ao baixo TTL. No entanto se a pesquisa falhar devido ao congestionamento de pacotes, esse mecanismo de aumento do TTL trabalha de firna contra-produtivamente, aumentando ainda mais a carga da rede e piorando o estado de congestionamento.

O P-AIMD apresenta boa adaptatividadeadaptabilidade, especialmente se compararmos seu desempenho com algumas configurações fixas. Seu desempenho está, em média, 5,18 pontos percentuais inferior à melhor configuração manual. Dentre os al-

goritmos comparados, no entanto, é o que apresenta os piores resultados em redes com poucas pesquisas / segundo. O P-ML, contudo, apresenta bom desempenho em todos os cenários, apresentando desempenho em média $2,72$ pontos percentuais inferior à melhor configuração manual do cenário, diferença aceitável em prol da adaptatividade-deadaptabilidade.

Na Figura 4.12 vemos o tempo de resposta da aplicação. O *Expanding rings* apresenta o pior resultado em todos os cenários, devido a seu mesmo mecanismo de incremento do TTL. A aplicação detecta que a pesquisa falhou após um tempo de *timeout* t configurável, de 5 segundos em nossas simulações. Além disso, notamos um crescimento exponencial do tempo de resposta, característico de redes congestionadas. Diferentemente, o tempo de resposta do P-ML e P-AIMD mostram que ambos conseguem prevenir de forma eficiente a formação de congestionamentos. O P-ML apresenta, no pior caso, tempo de resposta médio de $1,29$ segundos e o P-AIMD de $0,6$ segundos.

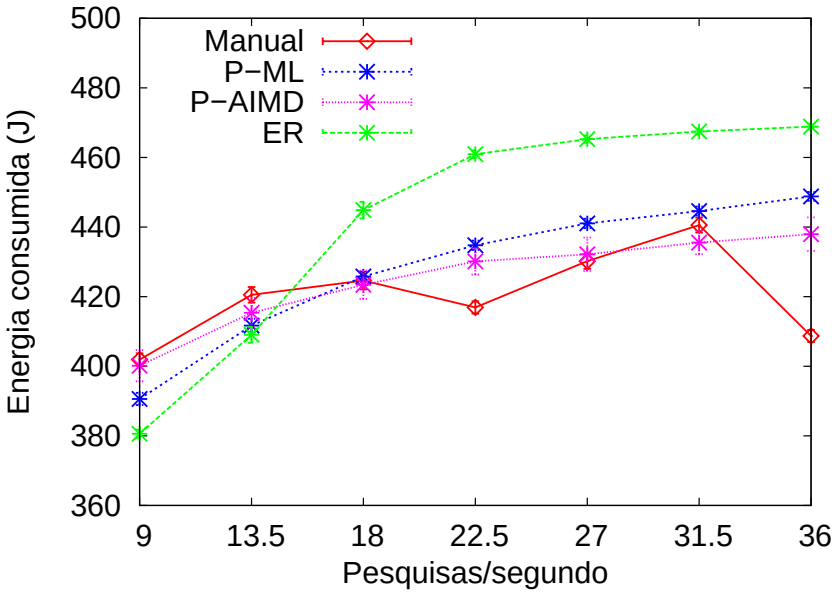

Figura 4.13. Consumo médio de energia das soluções propostas

O consumo geral de energia do *expanding rings* é, de forma geral, pior que a dos demais protocolos. Nos cenários com poucas pesquisas / segundo o *Expanding rings* apresenta um baixo consumo de energia devido à exploração otimista das réplicas. No entanto, em cenários com mais pesquisas / segundo, o congestionamento e frequentes *timeouts* da aplicação elevam o consumo de energia devido às novas tentativas de pesquisa. O consumo de energia do P-ML e do P-AIMD é comparável ao consumo de energia da melhor configuração manual de parâmetros, sendo que de forma geral o P-ML apresenta maior consumo de energia que o P-AIMD.

4.5 Conclusão

Apresentamos a metodologia de avaliação dos protocolos de auto-configuraçãoautoconfiguração do TTL propostos. Mostramos a necessidade da utilização de um algoritmo de auto-configuraçãoautoconfiguração do TTL e apresentamos os resultados dos dois algoritmos propostos, comparando seu desempenho com o da melhor configuração manual por cenário e do protocolo *Expanding rings*. Observamos que o *Expanding rings* não foi projetado para a utilização em MANETs, uma vez que confia em premissas de redes estruturadas. Vimos também que as soluções propostas apresentaram bons resultados em todos os cenários, mostrando sua capacidade de adaptação. O P-AIMD e o P-ML apresentaram, respectivamente, taxa de sucesso $5,18$ e $2,71$ pontos percentuais inferior à melhor configuração manual. AmbasAmbas as soluções conseguiram evitar a formação de congestionamento, evidenciado pela não crescimentopelo não crescimento exponencial do tempo de resposta.

Capítulo 5

Conclusão

As redes móveis ad hoc (MANETs) possibilitam a pronta execução de aplicações de apoio a resgate em situações de desastre e de troca de informações em campos de batalha. Elas são compostas por dispositivos móveis que se comunicam diretamente usando rotas de múltiplos saltos (*multi-hop*) sem fio. As redes Par-a-Par possuem grande sinergia com as redes móveis ad hoc, uma vez que buscam dividir a carga de trabalho entre os nós da redes. Devido à dinamicidade e diversidade das redes ad hoc, torna-se difícil configurar os parâmetros das redes P2P, sendo que a escolha do valor de cada parâmetro impacta no desempenho da aplicação.

Este trabalho abordou o problema da autoconfiguração de aplicações P2P não estruturadas sobre redes móveis ad hoc. Utilizando o arcabouço de um controlador MAPE propomos os controladores P-AIMD e P-ML. O P-AIMD é um controlador que utiliza um perceptron para decisão do congestionamento combinado à técnica de adaptação de parâmetro AIMD – "Aumento Aditivo, Decremento Multiplicativo". O P-ML é um controlador que utiliza um classificador na etapa de análise e um intuitivo algoritmo de adaptação de parâmetro.

Os controladores foram avaliados em uma rede móvel ad hoc empregando o protocolo Gnutella - no entanto as soluções podem ser aplicadas sobre outros protocolos P2P não estruturados. Verificamos experimentalmente a necessidade de adaptação do TTL no cenário de variação do número de pesquisas / segundo. Comparamos o desempenho da melhor configuração fixa de TTL com o do protocolo *Expanding Rings* e das soluções propostas.

Observamos que o *Expanding rings* não foi projetado para a utilização em MANETs, uma vez que confia em premissas de redes estruturadas. Vimos também que as soluções propostas apresentaram bons resultados em todos os cenários, mostrando sua capacidade de adaptação. O P-AIMD e o P-ML apresentaram, respectivamente,

taxa de sucesso 5,18 e 2,71 pontos percentuais inferior à melhor configuração manual. Ambas as soluções conseguiram evitar a formação de congestionamento, evidenciado pela não crescimento exponencial do tempo de resposta.

5.1 Trabalhos Futuros

Como trabalhos futuros pode-se realizar uma série de melhorias na proposta atual, bem como analisar propostas relacionadas. Em particular, é vital a análise das soluções propostas em diferentes cenários como variação do número de nós, da velocidade e da distribuição da carga.

Devido à estrutura modular do controlador MAPE, pode-se analisar de forma separada as etapas de análise e planejamento dos controladores P-ML e P-AIMD. Um exemplo seria utilizar o classificador da etapa de análise do P-ML e utilizar o AIMD da etapa de planejamento do P-AIMD. Outros algoritmos de análise e planejamento podem ser propostos de forma separada, uma vez que o arcabouço separa bem essas etapas.

Por fim, algumas propostas inovadoras deste trabalho podem ser utilizadas em outros contextos de redes. A técnica de utilizar algoritmos clássicos de aprendizado de máquina para classificação de eventos de rede pode ser aplicada, por exemplo, na detecção de intrusões em redes cabeadas. Pode-se estudar como as técnicas apresentadas podem auxiliar o desempenho de redes tolerantes a atraso.

Referências Bibliográficas

(2010). About olsr.

Adar, E. & Huberman, B. (2000). Free riding on gnutella. *First Monday*, 5(10-2).

Almeida, V. D. D. (2012). Análise de desempenho de protocolos protocolos de roteamento ad hoc e dtn em redes de emergência. Dissertação de mestrado, Universidade Federal de Minas Gerais.

Androutsopoulos, I.; Koutsias, J.; Chandrinos, K.; Paliouras, G. & Spyropoulos, C. (2000). An evaluation of naive bayesian anti-spam filtering. *Arxiv preprint cs/0006013*.

Borg, J. (2003). A comparative study of ad hoc & peer to peer networks. Dissertação de mestrado, University College London.

Boudec, J.-Y. L. & Vojnovic, M. (2006). The random trip model: stability, stationary regime, and perfect simulation. *IEEE/ACM Transactions on Networking*, 14(6):1153--1166.

Chiu, D. & Jain, R. (1989). Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems*, 17(1):1--14.

Cisco Systems (2005). Cisco aironet 350 series client adapters. http://www.cisco.com/en/US/products /hw/wireless/ps4555/products_data_sheet09186a0080088828.html.

Computing, A. (2006). An architectural blueprint for autonomic computing. *IBM White Paper*.

Conti, M.; Gregori, E. & Turi, G. (2005). A cross-layer optimization of gnutella for mobile ad hoc networks. Em *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pp. 343--354. ACM Press.

da Hora, D. N.; Macedo, D. F.; Oliveira, L. B.; Siqueira, I. G.; Loureiro, A. A. F.; Nogueira, J. M. & Pujolle, G. (2009). Enhancing peer-to-peer content discovery techniques over mobile ad hoc networks. *Elsevier Computer Communications*, 32:1445–1459.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. Em *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 27--34. ACM.

Federal, G. (2012). Programa um computador por aluno.

Franciscani, F. P.; Vasconcelos, M. A.; Couto, R. P. & Loureiro, A. A. F. (2005). Peer-to-peer over ad-hoc networks: (re)configuration algorithms. *Journal of Parallel and Distributed Computing (JPDC)*, 65(2):234--245.

Ganek, A. G. & Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5--18.

Haerri, J.; Filali, F. & Bonnet, C. (2006). Performance comparison of AODV and OLSR in VANETs urban environments under realistic mobility patterns. Em *Proceedings of the 5th IFIP Mediterranean Ad-Hoc Networking Workshop*.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P. & Witten, I. (2009). The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10--18.

Ishibuchi, H.; Nozaki, K.; Yamamoto, N. & Tanaka, H. (1995). Selecting fuzzy if-then rules for classification problems using genetic algorithms. *Fuzzy Systems, IEEE Transactions on*, 3(3):260--270.

Jacquet, P.; Muhlethaler, P.; Qayyum, A.; Laouiti, A.; Viennot, L. & Clausen, T. (1998). Optimized link state routing protocol. Relatório técnico, Internet Draft, draft-ietf-manetolsr-00. txt.

Jiang, H. & Jin, S. (2005). Exploiting dynamic querying like flooding techniques in unstructured peer-to-peer networks. Em *ICNP '05: Proceedings of the 13TH IEEE International Conference on Network Protocols*, pp. 122--131, Washington, DC, USA. IEEE Computer Society.

Katabi, D.; Handley, M. & Rohrs, C. (2002). Congestion control for high bandwidth-delay product networks. Em *SIGCOMM '02: Proceedings of the 2002 conference*

on Applications, technologies, architectures, and protocols for computer communications, pp. 89--102. ACM Press.

Langley, P. (1996). *Elements of machine learning*. Morgan Kaufmann Pub.

Lv, Q.; Cao, P.; Cohen, E.; Li, K. & Shenker, S. (2002). Search and Replication in Unstructured Peer-to-Peer Networks. Em *Proceedings of the 16th international conference on Supercomputing*, pp. 84--95.

Macedo, D. F.; Pujolle, G. & Nogueira, J. M. (2009). *Self-configuration of Multi-Hop Wireless Networks*. Tese de doutorado, Laboratoire d'Informatique Paris VI-LIP6, Université Pierre et Marie Curie-Paris Universitas.

Malone, D.; Duffy, K. & Leith, D. (2007). Modeling the 802.11 distributed coordination function in nonsaturated heterogeneous conditions. *IEEE/ACM Transactions on Networking*, 15(1):159--172.

McCanne, S.; Floyd, S.; Fall, K.; Varadhan, K. et al. (1997). Network simulator ns-2.

Mian, A.; Baldoni, R. & Beraldi, R. (2009). A survey of service discovery protocols in multihop mobile ad hoc networks. *Pervasive Computing, IEEE*, 8(1):66--74.

Oliveira, L. B.; Siqueira, I. G. & Loureiro, A. A. F. (2005). On the performance of ad hoc routing protocols under a peer-to-peer application. *Journal of Parallel and Distributed Computing (JPDC)*, 65(11):1337–1347.

Quinlan, J. (1986). Induction of decision trees. *Machine learning*, 1(1):81--106.

Quinlan, J. (1992). Learning with continuous classes. Em *Proceedings of the 5th Australian joint Conference on Artificial Intelligence*, pp. 343--348. Singapore.

Ripeanu, M. (2001). Peer-to-peer architecture case study: Gnutella network. Em *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pp. 99--100. IEEE.

Russell, S. & Norvig, P. (2010). *Artificial intelligence: a modern approach*. Prentice hall.

Saha, A. K. & Johnson, D. B. (2004). Modeling mobility for vehicular ad-hoc networks. Em *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks (VANET)*, pp. 91--92, New York, NY, USA. ACM.

Servetto, S. D. & Barrenechea, G. (2002). Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. Em *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 12--21. ACM Press.

Seys, S. & Preneel, B. (2009). Arm: Anonymous routing protocol for mobile ad hoc networks. *International Journal of Wireless and Mobile Computing*, 3(3):145--155.

Stoica, I.; Morris, R.; Karger, D.; Kaashoek, M. F. & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Computer Communications Review*, 31(4):149--160.

Sumathy, S. & Kumar, B. (2010). Secure key exchange and encryption mechanism for group communication in wireless ad hoc networks. *Arxiv preprint arXiv:1003.3564*.

Sundaresan, K.; Anantharaman, V.; Hsieh, H.-Y. & Sivakumar, R. (2005). ATP: A Reliable Transport Protocol for Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 4(6):588--603.

Tanenbaum, A. S. (2002). *Computer networks*. Prentice-Hall, Inc., 4 ed. edição.

Tsoumakos, D. & Roussopoulos, N. (2003). Adaptive probabilistic search for peer-to-peer networks. Em *Proceedings of the international conference on peer-to-peer computing (P2P)*, pp. 102--109.

Wang, J.; Osagie, E.; Thulasiraman, P. & Thulasiram, R. (2009). Hopnet: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. *Ad Hoc Networks*, 7(4):690--705.

Zhang, G. (2000). Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(4):451--462.