UNIVERSIDADE ESTADUAL DO CEARÁ

JOSÉ ALEX PONTES MARTINS

A-DYMO: UM ALGORITMO DE ROTEAMENTO BIO-INSPIRADO PARA REDES AD HOC

FORTALEZA – CEARÁ
2009

JOSÉ ALEX PONTES MARTINS

A-DYMO: UM ALGORITMO DE ROTEAMENTO BIO-INSPIRADO PARA REDES *AD HOC*

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Centro de Ciência e Tecnologia como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. Joaquim Celestino Júnior.

FORTALEZA - CEARÁ
2009

M379a Martins, José Alex Pontes

A-DYMO: Um Algoritmo de roteamento bio-inspirado para redes *ad hoc* / José Alex Pontes Martins. - Fortaleza, 2009.

89p. ; il.

Orientador: Prof. Dr. Joaquim Celestino Jr.

Dissertação (Mestrado Acadêmico em Ciência da Computação) – Universidade Estadual do Ceará, Centro de Ciência e Tecnologia.

1. Redes *Ad hoc*. 2. Roteamento. 3. Comportamento Coletivo. 4. Otimização de Colônias de Formigas. I. Universidade Estadual do Ceará, Centro de Ciência e Tecnologia.

CDD:004.6

JOSÉ ALEX PONTES MARTINS

A-DYMO: UM ALGORITMO DE ROTEAMENTO BIO-INSPIRADO PARA REDES *AD HOC*

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Centro de Ciência e Tecnologia como requisito parcial para obtenção do grau de Mestre.

Aprovado em: ____/____/____

BANCA EXAMINADORA

_____

Prof. Dr. Joaquim Celestino Júnior - UECE
(Orientador)

_____

Prof. Dr. André Ribeiro Cardoso - UECE

_____

Prof. Dr. Gustavo Augusto Lima de Campos - UECE

_____

Prof. Dr. Djamel Fawzi Hadj Sadok - UFPE

_____

Prof. Dr. Elias Teodoro da Silva Júnior - IFCE

A minha amada esposa Keila, por todo o amor, paciência e apoio que dedicou a mim durante toda essa jornada.

Agradecimentos

Mais do que nunca tenho certeza de que tão importante quanto alcançar o objetivo é trilhar o caminho até ele. Nesses últimos meses aprendi muito. Aprendi a lidar com pessoas, a compartilhar, a receber críticas, a ver as coisas de forma diferente.

Quero agradecer a todos os colegas com os quais tive o prazer de conviver. Em especial os colegas da primeira turma: Fabiano, Carlos Cidade, Robson, Renato, Vigno, Viviane e Márcia. Também não poderia esquecer dos colegas de "república": Fabiano, Walisson, Marcelo, Daladier, Vigno, Marçal II, Alysson, que fizeram da convivência do dia-a-dia uma grata surpresa, pois sempre estávamos juntos, depois das aulas, no espaço de convivência de nossa residência discutindo assuntos pertinentes ao mestrado e a vida.

Agradecimento especial ao colega Sérgio Luis, que tive o prazer de conhecer no final do curso de mestrado, mas cuja contribuição foi de fundamental importância na implementação do protocolo A-DYMO. Ele é o que chamo de Sayajin nível 7 do Linux.

Quanto aos mestres, só tenho que agradecer pela dedicação, empenho, conversas de corredor e exemplos. Sei que todos fizeram o melhor que podiam para contribuir com nossa formação. Quero agradecer especialmente ao Prof. Celestino pela sua dedicação, presença e direcionamento, sempre empenhado em resolver os problemas do momento, sem jamais esquecer dos objetivos finais.

Alex Martins

Resumo

Redes móveis *ad hoc* são um conjunto de dispositivos móveis sem fio que se comunicam sem infra-estrutura fixa, formando redes temporárias dinamicamente. Dessa forma, cada nó da rede é mais do que um receptor/transmissor de dados, é também um roteador repassando pacotes de dados para o seu devido destino. As principais características das redes *ad hoc* são: constante mudança na topologia da rede, capacidade limitada de energia de seus enlaces e restrição na largura de banda. Um protocolo de roteamento para redes *ad hoc* é composto por um algoritmo de roteamento juntamente com um conjunto de regras que monitoram o funcionamento da rede. Assim, os nós que participam da rede têm papel importante no gerenciamento dos recursos de redes *ad hoc*.Roteamento de formigas é um esquema de roteamento inspirado no comportamento forrageiro das formigas. O estudo do comportamento coletivo das formigas mostra que elas são hábeis em encontrar o menor caminho do ninho para uma fonte de alimentos, usando um particular modo de comunicação através de uma substância química chamada feromônio. Este trabalho produzirá um estudo sobre os principais protocolos para redes *ad hoc*, inteligência coletiva e inteligência coletiva aplicada a redes *ad hoc*, em especial a aplicação de roteamento de formigas em redes *ad hoc*. Finalmente, será criado um novo protocolo de roteamento para redes *ad hoc*, definido sobre uma variação de um dos principais protocolos para redes *ad hoc* já existente, através da inserção do roteamento de formigas e fazendo as modificações necessárias para viabilizar tal mecanismo. O novo algoritmo será comparado com sua versão tradicional em vários ambientes.

PALAVRAS-CHAVE: Redes *Ad hoc*. Roteamento. Comportamento Coletivo. Otimização de Colônias de Formigas.

Abstract

Mobile *ad hoc* networks are a set of wireless mobile devices that communicate without fixed infrastructure, forming temporary networks dynamically. Thus each node of the network is more than one receiver/sender of data, is also a router forwarding data packets to its proper destination. The main characteristics of *ad hoc* networks are: constant change in network topology, limited power of its links and restriction on bandwidth. A routing protocol for *ad hoc* networks are composed of a routing algorithm with a set of rules that monitor the operation of the network. Thus, the nodes participating in the network has important role in the management of resources in *ad hoc* networks. Routing of ants is a new routing scheme based on the behavior of ants forage. The study of collective behavior of ants shows that they are able to finding the least path of the nest to a food source, using a particular mode of communication by means of a chemical called pheromone. This work produced a study on the main protocols for *ad hoc* networks, collective intelligence and collective intelligence applied to *ad hoc* networks, in particular the application of ants to routing in *ad hoc* networks. Finally, will a new routing protocol for *ad hoc* networks, defined on a variation of one of the main protocols for *ad hoc* networks, through integration of routing of ants and making the necessary changes to enable this mechanism. The new algorithm will be compared with the traditional version in various environments.

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas e Siglas

ABR	Associativity Based Routing
ACO	Ant Colony Optimization
A-DYMO	Ant- Dynamic MANET On-demand Routing Protocol
AODV	*Ad hoc* On-demand Distance Vector
AS	Ant System
CBM	Content based multicast in *ad hoc* networks
CBRP	Cluster Based Routing Protocol
CEDAR	Core-Extraction Distributed *Ad hoc* Routing
CGSR	Clusterhead Gateway Switch Routing
CO	Combinatorial Optimization
DNA	Deoxyribonucleic Acid
DREAM	Distance Routing Effect Algorithm for Mobility
DSDV	Destination Sequence Distance Vector
DSR	Dynamic Source Routing
DYMO	Dynamic MANET On-demand Routing Protocol
FSR	Fisheye State Routing
GLS	Grid Location Service
GPS	Global Positioning System
HARP	Hybrid *Ad hoc* Routing Protocol
HSR	Hierarchical State Routing
IARP	Intra-zone Routing Protocol
ICMP	Internet Control Message Protocol
IERP	Inter-zone Routing Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LANMAR	Landmark *Ad hoc* Routing
LAR	Location Aided Routing
MANET	Mobile *ad hoc* network
NS-2	Network Simulator 2
OLSR	Optimized Link State Routing protocol
PBL	Protocolo Baseado em Localização
PEL	Protocolo baseado em Estabilidade do Link
PRH	Protocolo de Roteamento Híbrido
PRP	Protocolo de Roteamento Proativo
PRR	Protocolo de Roteamento Reativo
PSO	Particle Sworm Optimization
QoS	Quality of Service
RERR	Route Error
ROUTE_DELETE_PERIOD	Período para Exclusão de Rota
ROUTE_ERROR	Erro de Rota
ROUTE_VALID_TIMEOUT	Tempo de Validade de Rota
RREP	Route Reply
RREQ	Route Requisition
RREQ_TRIES	Tentativas de Requisição de Rota
SSR	Signal Stability-base adaptive Routing protocol
Target.HopCnt	Quantidade de saltos para alvo

Target.SeqNum	Número de sequência do alvo
TORA	Temporally Ordered Routing Algorithm
TSP	Traveling Salesman Problem
TTL	Time To Live
UDP	User Datagram Protocol
WRP	Wireless Routing Protocol
ZHLS	Zone-based Hierarchical Link State routing
ZRP	Zone Routing Protocol

Sumário

1 Introdução

Nos últimos anos, as tecnologias de dispositivos sem fio têm estado cada vez mais presentes no nosso cotidiano. Os avanços tecnológicos, baixo custo e aumento no poder dos transmissores sem fio vêm proporcionando uma grande utilização dos dispositivos móveis. Assim, redes móveis atraem significativamente mais pesquisadores, graças aos melhoramentos em sua flexibilidade e diminuição nos custos.

Segundo [1], em comparação com redes fixas, redes móveis possuem algumas características que as distinguem. Nelas, a mobilidade pode causar mudanças frequentes na topologia da rede, o que é raro acontecer nas redes fixas. Ao contrário da estabilidade na capacidade do link das redes fixas, a capacidade dos links sem fio varia constantemente devido a vários fatores, tais como: *poder de transmissão, sensibilidade do receptor, atenuação do sinal e outras interferências*. Adicionalmente, em redes móveis sem fio deve-se assumir como regra que ocorrerão *altas taxas de erro, restrição na energia dos dispositivos e limitação na largura de banda*.

Redes móveis podem ser classificadas em *redes móveis infra-estruturada de apóio* e *redes móveis ad hoc* que são as que não possuem infra-estrutura fixa de apóio. Nas redes móveis com infra-estrutura (ver Figura 1.1-a), os nós móveis têm estações base cabeadas dentro de seu raio de transmissão e recepção. As estações de base compõem a espinha dorsal de uma rede infra-estruturada. Por outro lado, redes móveis *ad hoc* (ver Figura 1.1-b) são autonomamente auto-organizáveis sem infra-estrutura de apóio. Nas redes móveis *ad hoc*, os nós movem-se arbitrariamente, fazendo com que a rede mude, rápida e inesperadamente, sua topologia. Adicionalmente, devido à limitação nas taxas de transmissão dos nós ser uma característica sempre presente,

alguns nós não podem comunicar-se diretamente com cada um dos outros. Por essa razão, os caminhos de roteamento nas redes *ad hoc* potencialmente contêm múltiplos saltos, e dessa forma muitos nós nas redes *ad hoc* tem responsabilidade de atuar como roteador.

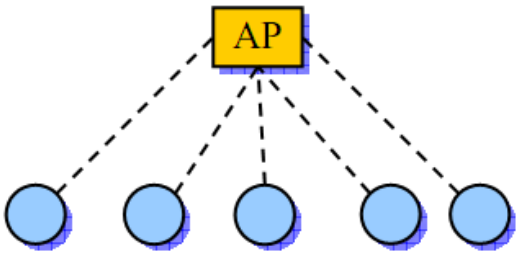


Figura 1.1-a: Rede infra-estruturada.

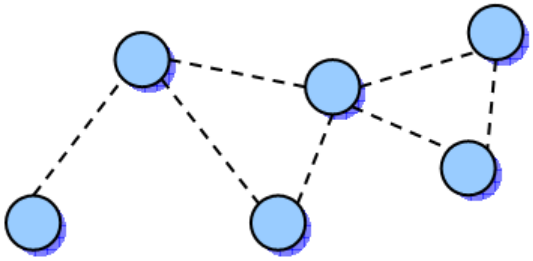


Figura 1.1-b: Rede *ad hoc*.

As atuais pesquisas em redes *ad hoc* estão sendo conduzidas principalmente nos campos de acesso ao meio, roteamento, gerenciamento de recursos, controle de energia e segurança[8][22][16]. Devido à importância dos protocolos de roteamento em redes dinâmicas de múltiplos saltos, uma grande quantidade de protocolos de roteamento para redes móveis *ad hoc* tem sido propostos nos últimos anos.

Em [2] vemos que existem alguns desafios que tornam o projeto de protocolos de roteamento para redes *ad hoc* uma tarefa bastante desafiadora. Inicialmente, em redes móveis *ad hoc*, a mobilidade dos nós causa frequentemente mudanças na topologia, o que resulta em partição na rede. Em segundo lugar, devido à variação e à imprevisível capacidade dos links sem fio, a perda de pacotes pode frequentemente acontecer. Além disso, a natureza de transmissão do meio sem fio introduz os problemas do terminal escondido[1] e do terminal exposto[2]. Dessa forma, nós móveis têm restrição de recursos nas áreas de energia, processamento e largura de banda, o que efetivamente necessita de esquemas de roteamento apropriados [2].

[1] Problema onde um terminal fora de alcance interfere na recepção de outro terminal.
[2] Problema que pode resultar em subutilização de um terminal.

Como um tipo de rede promissora em futuras aplicações móveis, redes móveis *ad hoc* estão atraindo mais e mais grupos de trabalho.

Devido a grande quantidade de trabalhos desenvolvidos sobre redes MANETs, novas abordagens foram utilizadas em busca de melhores resultados. Entre elas, destaca-se a meta-heurística Ant Colony Optimization (ACO). Uma técnica largamente utilizada em problemas de otimização combinatória, que se mostrou eficaz para resolver, entre outros, o problema de roteamento de redes cabeadas e sem fio[2].

Em redes *ad hoc* a otimização em colônias de formigas mostra-se adequada, pois tem como característica principal a constante renovação dos caminhos de roteamento, através de um mecanismo constante de busca, que, juntamente com um mecanismo probabilístico, encarrega-se de escolher a solução mais promissora. Num ambiente de conexão *ad hoc*, esse tipo de abordagem atende a um pré-requisito básico, que é renovação de oferta de soluções, já que a característica dinâmica dessas redes é um dos fatores mais importantes.

1.1 Objetivo

O objetivo principal desta pesquisa é a proposta e a implementação de um protocolo de roteamento para redes *ad hoc* chamado A-DYMO.

A-DYMO é um protocolo híbrido que utiliza em sua fase proativa um algoritmo do tipo ACO. Nesta fase uma formiga exploradora (EANT) irá descobrir e manter a topologia da rede durante todo o tempo. O caminho traçado por essa formiga utilizará como parâmetro de qualidade na escolha da rota a distância entre os nós em número de saltos. Na fase reativa, uma formiga de busca (ARREQ) irá proceder à ligação entre o nó de origem e o nó de destino para posterior efetiva entrega de dados. Os demais mecanismos da fase reativa são baseados no protocolo DYMO. Também no protocolo A-DYMO um modelo probabilístico de transição foi implementado, semelhante as trilhas de feromônio das formigas reais.

1.2 Organização da Dissertação

Esta dissertação está organizada da seguinte maneira:

O capítulo 1 é introdutório e descreve a motivação, os objetivos e a metodologia aplicada na pesquisa.

O capítulo 2 apresenta as características gerais das redes *ad hoc* e uma taxonomia bem aceita na literatura sobre os protocolos de roteamento para redes *ad hoc*. Este capítulo também descreve os protocolos que mais contribuíram para a criação do protocolo alvo da pesquisa, dentre eles o protocolo DYMO.

O capítulo 3 discorre sobre inteligência coletiva, meta-heurísticas e mais especificamente sobre a meta-heurística otimização de colônias de formigas. Também apresenta os protocolos para redes *ad hoc* que já implementam a otimização de colônia de formigas. Descreve ainda o funcionamento geral de alguns desses protocolos além das modificações feitas por cada um deles para adaptar-se ao ambiente de redes *ad hoc*.

O capítulo 4 tratará sobre o algoritmo A-DYMO. Será mostrado como foi possível inserir o algoritmo de roteamento de formigas no protocolo alvo da pesquisa e quais ajustes foram necessários para viabilizar tal procedimento.

O capítulo 5 finalmente apresentará um comparativo entre a performance dos protocolos DYMO e A-DYMO que será obtido através de simulações utilizando o NS-2. No capítulo final serão mostradas as conclusões e sugeridos os rumos para trabalhos futuros utilizando essa pesquisa.

2 Protocolos para Redes *Ad hoc*

2.1 Introdução

Segundo [1], muitas aplicações estão desfrutando de comunicação móvel, graças aos avanços das tecnologias de comunicação sem fio que, devido ao baixo custo e ao aumento do poder dos transmissores têm se mostrado cada vez mais viável tecnológico e economicamente. Dessa forma, redes móveis têm atraído cada vez mais interesse da comunidade científica devido ao melhoramento da flexibilidade em aplicações e à redução do custo de implementação.

Comparadas com as redes cabeadas, as redes móveis têm características únicas. Nelas, a mobilidade dos nós pode causar frequentes mudanças na topologia, o que é raro acontecer em redes cabeadas. Ao contrário da estabilidade dos links nas redes fixas, nas redes móveis os links variam continuamente devido aos impactos na transmissão, tais como: sensibilidade do receptor, ruído, desvanecimento e interferências. Adicionalmente, redes móveis sem fio têm altas taxas de erros, restrição de energia e limitação na largura de banda.

Redes móveis podem ser classificadas quanto à dependência de uma infra-estrutura de apoio. Dessa forma existem as redes móveis infra-estruturadas e as redes móveis *ad hoc*. Nas redes móveis infra-estruturadas, os nós móveis têm pontos de acesso (também conhecidos como estação base) dentro do seu raio de transmissão. Os pontos de acesso criam uma espinha dorsal para a rede infra-estruturada. Ao contrário, as redes móveis *ad hoc* são autonomamente auto-organizáveis e sem infra-estrutura de suporte. Nas redes móveis *ad hoc*, os nós movimentam-se arbitrariamente, fazendo com

a que a rede venha a sofrer uma rápida e imprevisível mudança em sua topologia. Adicionalmente, devido os nós em redes móveis *ad hoc* terem normalmente uma limitação em suas taxas de transmissão, alguns nós não conseguem comunicar-se diretamente com todos os outros. Portanto, os caminhos de roteamento nas redes *ad hoc* potencialmente irão conter múltiplos saltos e cada nó em uma rede móvel *ad hoc* terá a responsabilidade de atuar como um roteador.

2.2 Taxonomia

Vemos em [2], que para comparar e analisar protocolos de roteamento para redes móveis *ad hoc*, é importante observar o método apropriado de classificação de tais protocolos. Métodos de classificação ajudam pesquisadores e projetistas a entender características distintas de um protocolo e procurar sua relação com outros.

2.2.1 Como a informação de roteamento é adquirida e mantida

Um dos mais populares métodos para distinguir os protocolos de roteamento de redes móveis *ad hoc* é baseado em *como a informação de roteamento é adquirida e mantida pelos nós móveis.* Usando esse método, os protocolos de roteamento para redes móveis *ad hoc* podem ser divididos em roteamento proativo, roteamento reativo e roteamento híbrido [2].

O roteamento proativo é também chamado de protocolo de roteamento *dirigido por tabela*. Ao usar um protocolo de roteamento proativo, os nós em uma rede móvel *ad hoc*, continuamente avaliam as rotas para *todos* os nós alcançáveis e tentam manter uma informação consistente de roteamento, através de tabelas. Dessa forma, o nó de origem tem à disposição um caminho de roteamento sempre que desejar.

Quando ocorre uma mudança na topologia da rede, imediatamente atualizações devem ser enviadas a todos os nós de forma a propagar a nova visão da rede. Muitos protocolos de roteamento proativos propostos para redes móveis *ad hoc* têm propriedades herdadas de algoritmos usados em redes fixas ou cabeadas. Por isso, para adaptar as características dinâmicas da rede móvel *ad hoc*, algumas modificações são necessárias. Usando algoritmos de roteamento proativos, nós móveis proativamente atualizam o estado da rede, através de tabelas de roteamento e mantêm as rotas sem levar em consideração se existe ou não tráfego de dados. Dessa forma a sobrecarga de

tráfego de controle na rede é muito alta. Como exemplo de protocolos proativos temos DSDV[8], WRP[10], FSR[16], dentre outros.

Protocolos de roteamento reativos para redes móveis *ad hoc* são também chamados protocolos de roteamento *sob demanda*. Nos protocolos de roteamento reativos, os caminhos de roteamento são buscados apenas quando há necessidade. Uma operação de descoberta de rota chama este procedimento para determinar a rota. O procedimento de descoberta termina quando uma rota é encontrada ou quando, depois de analisar toda a rede, não é possível determinar o caminho buscado. Como exemplos de protocolos reativos temos DSR[7], AODV[6], FSR[16], dentre outros.

Nas redes móveis *ad hoc*, rotas ativas podem ser desconectadas devido a mobilidade de nós. Dessa forma, a manutenção de rotas é uma importante operação dos protocolos reativos. Comparado com protocolos de roteamento proativos, para redes móveis *ad hoc*, a menor sobrecarga de tráfego de controle é uma vantagem que os distingue dos protocolos de roteamento proativos. Assim, protocolos de roteamento reativos têm melhor escalabilidade que protocolos de roteamento proativos em redes móveis *ad hoc*. Porém, quando protocolos de roteamento reativos são usados, os nós de origem sofrem longos *delays* pela busca de rotas antes que possam encaminha seus pacotes de dados.

Protocolos de roteamento híbridos são propostos para combinar características de ambos os protocolos de roteamento proativos e reativos e superar suas desvantagens. Normalmente, protocolos de roteamento híbridos para redes móveis *ad hoc* exploram arquiteturas de redes hierárquicas. São exemplos de protocolos híbridos ZRP[14], HARP[17], dentre outros.

2.2.2 Quais regras regem o nó no esquema de roteamento

Outro método de classificação é baseado em regras através das quais os nós relacionam-se no esquema de roteamento. Assim, existem os protocolos de roteamento *uniformes e não-uniformes*. Os uniformes são aqueles em que todos os nós têm a mesma regra de atuação, importância e funcionalidade. Protocolos de roteamento uniforme normalmente assumem uma estrutura plana de rede. Exemplos de protocolos de roteamento uniformes são o WRP, DSR, AODV, DSDV etc. Nos protocolos de roteamento não-uniformes para redes móveis *ad hoc*, alguns nós cumprem tarefas

distintas e/ou funções de roteamento. Normalmente, algoritmos distribuídos são explorados para selecionar aqueles nós especiais. Em alguns casos, abordagens de roteamento não-uniforme são referenciadas para estruturas de redes hierárquicas, a fim de facilitar o gerenciamento e organização dos nós. Além disso, protocolos de roteamento não-uniformes podem ser divididos em roteamento hierárquico baseado em *zona*, roteamento hierárquico baseado em aglomeração *(cluster)* e roteamento hierárquico baseado *nós de núcleo (core-node).*

2.2.3 Usando métricas para construção do caminho

As métricas usadas para construir caminhos de roteamento podem ser usadas como critério para classificação de protocolos de roteamento para redes móveis *ad hoc*. Por isso, muitos protocolos para redes móveis *ad hoc* usam *número de saltos* como uma métrica. Se há múltiplos caminhos de roteamento a avaliar, o caminho com o menor número de saltos será selecionado. Se todos os links sem fio na rede têm a mesma probabilidade de falhar, os caminhos de roteamento mais curtos são mais estáveis que os mais longos e podem obviamente diminuir a sobrecarga de tráfego e reduzir os pacotes de colisão. Porém, a suposição de mesma probabilidade de falha pode não ser verdadeira em redes móveis *ad hoc*. Então, a estabilidade de um link pode ser considerada na fase de construção da rota. Por exemplo, os protocolos ABR[11] e SSR[15] são propostos para usar a estabilidade do link ou intensidade do sinal como métrica para roteamento.

Com a popularidade da computação móvel, algumas aplicações podem ter diferentes requisitos de Qualidade de Serviço (QoS). Para adequada especificação de requisito de QoS, métricas apropriadas devem ser usadas para roteamento e encaminhamento de pacotes em redes móveis *ad hoc*. Como nas redes cabeadas, os protocolos de roteamento com QoS para redes móveis *ad hoc* podem usar métricas tais como: largura de banda, atraso, pacotes perdidos etc. Como exemplo, temos o protocolo CEDAR[13] no qual são usadas a largura de banda e a estabilidade do link como métricas para construção de caminhos de roteamento.

2.2.4 Baseado na topologia, destino e localização

Em um protocolo de roteamento baseado em topologia para redes móveis *ad hoc*, os nós coletam informações sobre a topologia da rede para criar decisões de roteamento. De forma semelhante, há algumas propostas de protocolos de roteamento baseados em destino para redes móveis *ad hoc*. Nesse tipo de protocolo, um nó precisa conhecer apenas o próximo salto através do caminho de roteamento quando encaminha o pacote para o destino. Por exemplo, DSR é um protocolo de roteamento baseado em topologia. Já o AODV e DSDV são protocolos de roteamento baseados em destino. A disponibilidade de um GPS ou sistema de localização semelhante permite que nós móveis acessem informações geográficas facilmente. Em protocolos de roteamento baseados em localização, a posição relativa entre um nó que repassa um pacote e o seu destino, juntamente com a mobilidade do nó, podem ser usadas para, ao mesmo tempo, descobrir uma rota e encaminhar um pacote. Abordagens de roteamento baseadas em localização para redes móveis *ad hoc* podem ser divididas em dois esquemas. No primeiro esquema, os nós móveis enviam pacotes levando em consideração somente a informação de localização e não precisam de nenhum outro conhecimento. No outro esquema, usam ambas as informações de localização e de topologia. LAR[18] e DREAM[15] são exemplos de protocolos de roteamento baseados em localização para redes móveis *ad hoc*. Por fim, para algumas aplicações móveis, é desejável o uso do conteúdo da mensagem como critério para decisão de roteamento ao invés do critério de endereço de destino. Assim, temos roteamento baseado em conteúdo. CBM[12] é um exemplo desse tipo de protocolo.

Dentre todos os protocolos acima discutidos, alguns apresentam certos mecanismos que se tornaram, naturalmente, um padrão a ser seguido. Além disso, o pioneirismo de outros os colocaram como uma referência natural para o desenvolvimento e evolução de protocolos para redes *ad hoc*. Dentre os mais relevantes para esta pesquisa podemos citar DSDV, DSR, AODV, ZRP, DYMO os quais serão melhor detalhados nas próximas seções.

2.3 DSDV

O *Destination Sequence Distance Vector* [8] (DSDV) é um protocolo de roteamento proativo unicast[3] para redes móveis *ad hoc*. Ele é baseado no tradicional algoritmo de Bellman-Ford[4].

Nas tabelas de roteamento do DSDV, uma entrada armazena o próximo salto em direção de um destino, a métrica de custo para o caminho de roteamento, o tempo de inserção da rota e o número de sequência do destino que é criado apenas por ele mesmo. Números de sequência são usados no DSDV para distinguir entre rotas desatualizadas e rotas atualizadas e assim evitar a formação de laços.

Destino	Prox. Salto	Saltos	Num. Seq.	Tempo inst.
A	A	0	A-846	001000
B	B	1	B-942	001200
C	B	2	C-378	001500
D	B	3	D-234	001300

Tabela 2-1: Tabela de roteamento do nó *A*

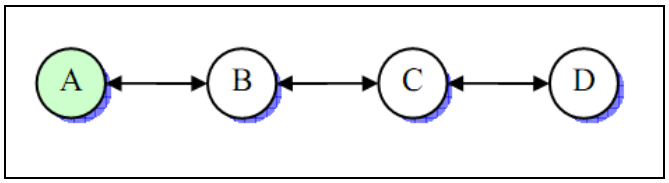


Figura 2.3-a: Rede *ad hoc* utilizando DSDV

As atualizações de rota do DSDV podem ser dirigidas por tempo ou por evento. Cada nó transmite periodicamente atualizações, incluindo sua informação de rota para seu vizinho imediato. Quando uma mudança significativa ocorre desde a última atualização, um nó pode transmitir sua tabela de rotas alterada em um evento do tipo gatilho. Além disso, o DSDV envia dois tipos de atualização de tabelas de rotas. Um é do tipo *cópia completa*, onde toda a tabela é incluída dentro da atualização. Uma cópia completa atualizada pode transportar vários pacotes. O outro tipo é uma *atualização parcial* que contém somente aquelas entradas em que certas métricas foram alteradas desde a última atualização. Adicionalmente, a atualização parcial cabe apenas em um pacote.

[3] Endereçamento a um único e específico destino.
[4] Algoritmo de estado de enlace.

A Tabela 2-1 mostra a estrutura de armazenamento de rotas do ponto de vista do nó A definido na Figura 2.3-a. Observe que A armazena o próximo salto, número de saltos, número de sequência e tempo de inserção da rota. Todas as rotas só são alcançáveis através do nó B. Nesse caso, quando houver uma mudança na rede o nó A irá atualizar sua tabela de rotas através das informações passadas pelo nó B.

2.4 DSR

Dynamic Source Routing [7] (DSR) é um protocolo de roteamento reativo unicast que utiliza um algoritmo de roteamento na origem. Em um algoritmo de roteamento na origem, cada pacote de dados possui a informação de roteamento completa para cada destino alcançável. Além disso, no DSR cada nó usa uma tecnologia de *cache*[5] para manter as informações de rotas que ele aprendeu com o trafego.

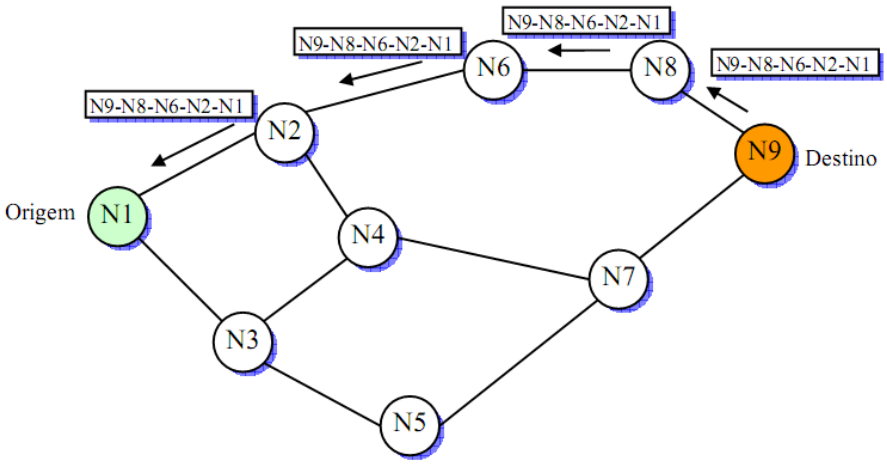


Figura 2.4-a: Roteamento utilizando DSR. Roteamento na Origem.

Há duas grandes fases no DSR, a fase de descoberta de rota e a fase de manutenção de rotas, conforme pode ser observado na Figura 2.4-a. Quando um nó de origem deseja enviar um pacote, ele inicialmente consulta seu cache de rotas. Se a rota necessária está disponível, o nó de origem inclui essa informação de roteamento dentro do pacote de dados antes de enviá-lo. Caso contrário, o nó de origem inicia uma operação de descoberta de rotas pelo envio por *broadcast*[6] de um pacote de descoberta de rotas. Um pacote de descoberta de rotas contém o endereço de ambos, origem e destino e um número de sequência para identificar a requisição. Ao receber um pacote de requisição de rota, um nó verifica seu cache de rotas. Se o nó não tem uma

[5] Tipo de armazenamento temporário.
[6] Envio de mensagens para todos os possíveis receptores.

informação para o destino requerido, ele adiciona seu endereço no campo de registro de rota do pacote de requisição de rota. Então, o pacote de requisição de rota – RREQ – é repassado para seus vizinhos. Para limitar a sobrecarga de comunicação dos pacotes de requisição de rotas, os nós os processam de forma a verificar se ele já o recebeu antes e se seu endereço já está presente no registro de rotas do pacote. Se o pacote de requisição de rotas chegar ao destino ou atingir um nó intermediário que tenha a informação do destino, um pacote de resposta de rota é criado. Quando um pacote de resposta de rota – RREP –é gerado pelo destino, ele conterá o endereço dos nós que repassaram o pacote de requisição de rotas. Por outro lado, o pacote de resposta de rota conterá os endereços dos nós contidos no pacote de requisição de rotas, repassados a ele, concatenados com a rota existente no cache de um nó intermediário.

Depois de criado um pacote de resposta de rota, seja por um nó intermediário seja pelo destino, será necessário enviá-lo de volta até a origem. Há três possibilidades para se obter a rota de retorno até a origem. A primeira delas é que o nó tenha uma rota armazenada para a origem. A segunda possibilidade é que a rede seja simétrica[7] (links bidirecionais). O pacote de resposta de rota é enviado usando a coleção de informações de roteamento registrado no campo do pacote, mas sendo lido de forma inversa. Então, um novo procedimento de descoberta de rota será iniciado até a origem. A rota descoberta é adicionada a este pacote de descoberta de rota.

No DSR, quando a camada de enlace detecta uma desconexão, um pacote ROUTE_ERROR é enviado de volta até a origem. Depois de receber o pacote ROUTE_ERROR, a origem inicia outra operação de descoberta de rota. Durante este processo, os nós intermediários também recebem o pacote ROUTE_ERROR, então verificam se uma entrada para a rota falha existe em seu cache, se sim, removem a rota que causou a desconexão.

O DSR frequentemente aumenta a sobrecarga de tráfego por conter a informação de roteamento completa em cada pacote de dados, o qual degrada sua performance de roteamento [2].

[7] Rede onde seus componentes têm mesma capacidade.

2.5 AODV

O protocolo *Ad hoc On-demand Distance Vector Routing* [6] (AODV) é um protocolo de roteamento reativo unicast para redes *ad hoc*. Como um protocolo de roteamento reativo, AODV mantêm apenas as informações de rotas sobre os caminhos ativos. No AODV, a informação de roteamento é mantida nas tabelas de roteamento dos nós. Cada nó móvel mantém uma tabela de roteamento de *próximo-salto*, no qual contêm os destinos para as rotas ativas. Uma entrada na tabela de roteamento expira, se ela não foi usada ou através de um tempo pré-determinado. Além disso, AODV adota a técnica de número de sequência usado pelo DSDV e adaptado para roteamento sob demanda.

No AODV, quando um nó de origem deseja enviar pacotes para um destino, mas não possui a rota disponível, ele inicia uma operação de descoberta de rota. Na operação de descoberta de rota, a origem envia, por broadcast, pacotes de requisição de rota, chamados RREQ. Um RREQ inclui o endereço da origem e do destino, o ID do broadcast, o qual é utilizado como seu identificador, o último número de sequência do destino além do número de sequência do nó de origem. Os números de sequência são importantes para garantir a ausência de laços e a atualização das rotas. Para reduzir a sobrecarga criada pela inundação, um nó descarta os RREQs já processados. O RREQ inicia com um pequeno valor de TTL (Time-To-Live). Se o destino não é encontrado, o TTL é aumentado nos próximos RREQs.

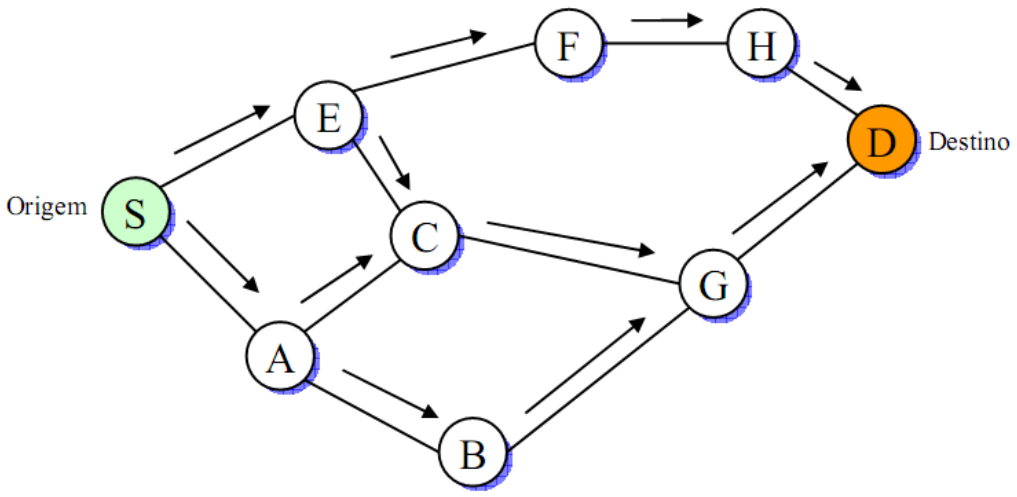


Figura 2.5-a: Busca de rota utilizando AODV.

No AODV, cada nó preserva um cache para manter a trilha de RREQs que ele recebeu. O cache também armazena o caminho de volta para cada nó que originou uma busca. Quando o destino ou um nó que tem uma rota para o destino recebe o RREQ, ele verifica o número de sequência do destino conhecido por ele e o especificado no RREQ. Para garantir informações de roteamento atualizadas, um pacote resposta de rota (RREP) é criado e enviado de volta para a origem, somente se o número de sequência do destino for igual ou maior que o número especificado no RREQ. AODV usa somente links simétricos e um RREP segue o caminho reverso do respectivo RREQ. Ao receber o pacote RREP, cada nó intermediário ao longo da rota atualiza as entradas de sua tabela de próximo-salto para o respectivo nó de destino. Pacotes RREP redundantes ou com menor número de sequência serão descartados.

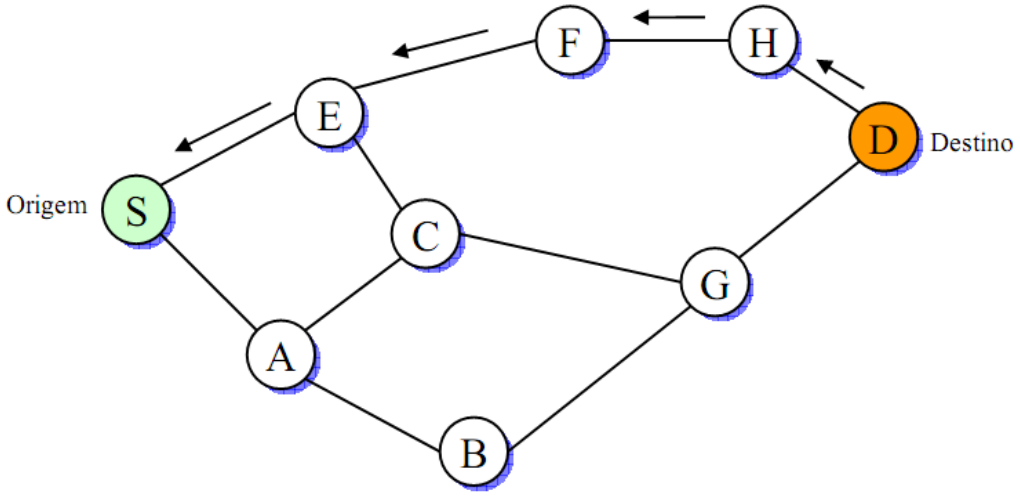


Figura 2.5-b: Resposta de rota utilizando AODV

No AODV, o nó utiliza mensagens *hello*[8] para notificar sua existência a seus vizinhos. Então, o status do link para o próximo salto em uma rota reativa pode ser monitorada. Quando um nó descobre uma quebra de link, ele envia por broadcast um pacote de erro de rota (RERR) para seus vizinhos, o qual é propagado por toda a rede informando aos nós que possuem aquela entrada sobre a desconexão. Quando a origem recebe um RRER, ela pode reiniciar a operação de descoberta de rota, se ainda tiver dados a serem enviados.

Na Figura 2.5-a um RREQ é propagado pela rede até chegar ao destino. Na Figura 2.5-b o destino responde com um RREP e os nós participantes do caminho de resposta conhecem apenas seus predecessores.

[8] Tipo de mensagem para detecção de vizinhança.

2.6 ZRP

O *Zone Routing Protocol* [14] (ZRP) é um protocolo de roteamento híbrido para redes móveis *ad hoc*. Os protocolos híbridos são propostos, em sua maioria, para reduzir a sobrecarga de controle da abordagem de roteamento proativa e diminuir a latência[9] causada pela operação de busca de rota na abordagem de roteamento reativa.

No ZRP, a rede é divida em zonas de roteamento, de acordo com a distância entre os nós móveis. Dado uma distância em saltos *d* e um nó *N*, todos os nós dentro da distância *d* de saltos pertencem à zona de roteamento do nó *N*. Os nós de borda de *N* são seus nós de vizinhança e estão exatamente a *d* saltos de distância de *N*.

No ZRP, diferentes abordagens de roteamento são exploradas para pacotes interzona e intrazona. A abordagem proativa, utilizando o protocolo IARP (IntrA-zone Routing Protocol) é usada dentro da zona de roteamento enquanto o protocolo IERP (IntEr-zone Routing Protocol), é usado entre as zonas de roteamento. O IARP mantém as informações do estado do link para os nós dentro da distância especificada em *d*. Logo, se um nó de origem e um de destino estão dentro de uma mesma zona de roteamento, a rota pode ser disponibilizada imediatamente. Muitos dos esquemas de roteamento proativo existentes podem usar os protocolos como um IARP para ZRP. O protocolo IERP, reativamente, inicia uma descoberta de rota quando o nó de origem ou o nó de destino estão dentro de diferentes zonas. A descoberta de rota no IERP é semelhante ao DSR com a exceção de que as requisições de rotas são propagadas através dos nós de borda.

[9] Tempo aguardado até que uma rota seja disponibilizada para o encaminhamento dos pacotes.

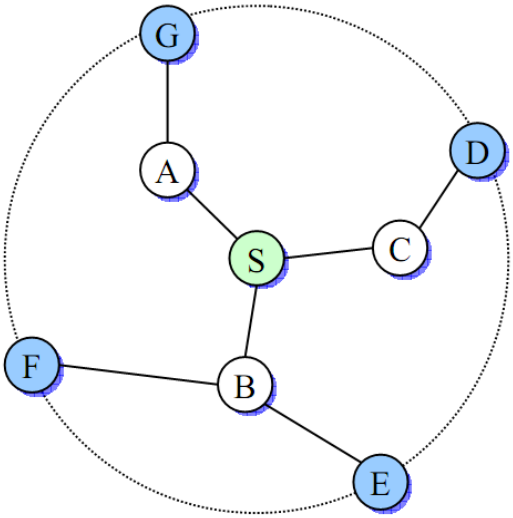


Figura 2.6-a: Estado da rede utilizando ZRP.

2.7 DYMO

O protocolo de roteamento reativo DYMO foi escolhido para ser a base da fase reativa do protocolo A-DYMO. Grande parte dos seus mecanismos ficaram intactos, tais como, verificação de pacotes duplicados, criação do RREP, manutenção de rotas, entre outros. Já a fase de busca de rotas foi alterada com a modificação da função do pacote de busca RREQ. Assim, uma explicação detalhada do DYMO se faz necessária.

O *Dynamic MANET On-demand* [3] (DYMO) é um protocolo de roteamento reativo desenvolvido para redes sem-fio de múltiplos saltos, oferecendo adaptação para mudanças na topologia e determinando rotas *unicast* entre nós sob demanda. Suas atividades principais são a descoberta e manutenção de rotas, o que é feito através das mensagens Requisição de Rotas (RREQ), Resposta de Rotas (RREP) e Erro de Rota (RERR). Para evitar a utilização de informações de roteamento antigas, assim como evitar a formação de *loops*, são utilizados números de sequência.

Este protocolo foi desenvolvido para atender redes móveis de pequena, média e larga escala, formadas por nós com restrições de memória. Ele é capaz de trabalhar com diversos tipos de tráfego, embora tenha melhor desempenho com tráfego esparso[10]. A confiança entre os nós é uma premissa assumida, uma vez que o DYMO depende da colaboração entre os nós para o encaminhamento dos pacotes. Para atender aos

[10] Informalmente é um tipo de tráfego com poucas conexões.

requisitos de restrição de recursos, poucos estados são utilizados no roteamento e apenas as informações sobre destinos ativos são mantidas nas tabelas de roteamento.

A sua operação básica consiste do envio dos RREQ pela rede para o descobrimento de rotas. Cada nó intermediário que recebe um RREQ guarda a rota para o nó que originou o RREQ e o re-encaminha. O nó destino, ao recebê-lo, responde com um RREP por unicast. Da mesma forma, todo nó intermediário que encaminha um RREP guarda a rota para o nó que originou o RREP em sua tabela de roteamento.

Para se adaptar às mudanças da topologia, ao receber um pacote que deve ser enviado para um enlace que não está mais disponível, o nó notifica a fonte através de uma mensagem de RERR. Caso a fonte ainda deseje enviar pacotes para aquele destino, ela deve iniciar um novo processo de descoberta de rotas.

2.7.1 Tabela de Roteamento

Os campos obrigatórios de cada entrada da tabela são o endereço de destino, o número de sequência associado, o próximo salto, a interface utilizada, o tempo de validade da rota e o tempo para se excluir uma rota. É importante ressaltar que mesmo que uma rota não seja mais válida, ela continua sendo uma informação útil ao roteamento até que se atinja o tempo de exclusão. Os campos opcionais são o número de saltos e as indicações de se é *gateway*[11] ou não, se é endereço de rede ou *host*[12] e se aquela rota já foi ou não utilizada para encaminhamento de dados.

A tabela de roteamento mantida por cada nó é constituída de campos obrigatórios e opcionais, sendo que a não utilização dos opcionais gera uma queda no desempenho do protocolo, embora seja garantido o funcionamento correto [3].

2.7.2 Mensagens

As mensagens do DYMO são enviadas utilizando UDP pela porta TBD [3], e seguem o modelo generalizado recomendado pelo IETF. O comprimento do campo IP pode ser adaptado tanto para 32 quanto para 128 bits, atendendo ao IPv4 e IPv6.

[11] Porta de comunicação.
[12] Qualquer máquina ou computador conectado a uma rede.

As mensagens de roteamento, RREQ e RREP, possuem os campos obrigatórios IP de destino, número máximo de saltos pelos quais a mensagem ainda pode passar, IP do próximo salto, IP de origem e número de sequência. No caso do RREQ, o campo próximo salto é preenchido com uma indicação de inundação. Entre os campos opcionais estão: o último número de sequência para o destino, rotas para outros nós da rede, adicionadas pelos nós intermediários. Ainda temos, o número de saltos desde a origem, as identificações se é um endereço de rede ou *host* e se é ou não um *host* no caminho para a Internet. Temos também, os identificadores de qual é o destino no bloco de endereços, qual a origem e quais são os nós que não estão entre a origem e o destino do pacote, além de um indicador de se as informações sobre aquele nó devem ser ignoradas ou não.

As mensagens RERR são utilizadas para avisar que uma rota válida não está disponível para um destino particular ou um conjunto de destinos. Os seus campos obrigatórios são IP de origem, IP de destino, IP do próximo salto inalcançável, número de saltos dados e o número máximo de saltos que ainda podem ser dados até o destino. Os opcionais são as demais rotas que utilizam o enlace indisponível, o número de sequência dos nós inalcançáveis e o indicador de ignorar ou não as informações para aquele endereço para invalidar rotas.

2.7.3 Números de Sequência

Os números de sequência desempenham um papel importante no funcionamento do protocolo, pois permitem avaliar se as rotas são novas ou não e garantem que não ocorrerão *loops*.

Cada nó utilizando o DYMO deve manter os seus próprios números de sequência em um total de 16 bits, representando um número inteiro sem sinal. O número de sequência zero é reservado para estruturas de dados que representam um número de sequência desconhecido. Se o número de sequência chegar ao seu valor máximo, ou seja, 65535, ao ser incrementado, ele deve passar a 256, para indicar aos outros nós que ele passou do valor máximo, e não se perdeu.

No caso de perda do número de seqüência, é necessário tomar algumas atitudes para evitar a formação de *loops*. Assim, sempre que o nó perder o seu número de seqüência, ele deve esperar por pelo menos um período chamado

ROUTE_DELETE_PERIOD antes de voltar a participar completamente do protocolo. Durante este tempo, ele deve processar e armazenar as informações obtidas com mensagens de controle, embora não deva retransmiti-las. Caso seja necessário encaminhar algum pacote de dados, o nó deve enviar um RERR para o nó que fez o pedido e em seguida deve reiniciar o seu período de espera. Após o período de espera, o nó inicializa seu número de sequência com um e deve iniciar a sua operação normal.

2.7.4 Atualização da Tabela de Roteamento

Sempre que uma informação de roteamento nova para um determinado nó é obtida, ela é avaliada segundo a sua qualidade antes de ser inserida na tabela. A primeira comparação feita é sobre a idade da rota:

- Se o número de sequência da nova informação de roteamento for menor que o número de sequência que o nó guarda para a rota, a nova informação é considerada desatualizada e não é utilizada na tabela de roteamento.

- Se os dois números de sequência forem iguais, para evitar *loops*, é necessário observar informações adicionais sobre o número de saltos.

    - Se a nova mensagem de roteamento ou a entrada na tabela não possuir número de saltos, ou este estiver igual a zero, a tabela fica propensa a causar *loops*, significando que essa nova informação não deve ser utilizada.

    - Se o número de saltos da nova rota for maior que o número de saltos da rota inserida na tabela mais um, a informação também é considerada propensa a *loops* e não deve ser utilizada.

    - Se o numero de saltos da nova rota for maior ou igual ao número de saltos da rota da tabela, a nova rota é considerada inferior e também não deve ser usada.

- Qualquer informação nova que não obedeça a nenhuma das suposições acima é considerada atualizada e incapaz de gerar *loops*, devendo ser utilizada para atualizar a tabela de roteamento.

A atualização da tabela é feita substituindo os valores antigos armazenados pelos novos. Cabe destacar que o número de sequência do nó para a rota atualizada passa a ser o valor do número de sequência obtido com a nova informação de roteamento e o valor do tempo de validade da rota será dado pelo tempo atual mais o tempo determinado por ROUTE_VALID_TIMEOUT. Todos os valores de campos que não forem descritos na mensagem devem receber zero.

Sempre antes de enviar um dado, é necessário observar o tempo de expiração da rota. Se o tempo de descarte da rota tiver sido atingido, ela deverá ser eliminada da tabela. Se o tempo de validade já tiver passado, a rota também não deve ser utilizada e um novo processo de descobrimento de rotas é necessário. A rota inválida deve ser mantida para preencher campos nas mensagens de roteamento com os últimos valores conhecidos.

2.7.5 Route Request (RREQ)

Para criar um RREQ, o nó deve atualizar o seu número de sequência, incrementando-o com um, e em seguida, adicioná-lo à mensagem. Logo depois, se existir algum valor prévio de número de sequência para aquela rota, ele deve ser inserido no campo opcional da mensagem de roteamento para o número de sequência conhecido. Se esse valor não for preenchido, ele será assumido como desconhecido pelos nós intermediários e apenas o nó destino poderá responder a essa mensagem. Se ele for preenchido com zero, os nós intermediários que conhecerem alguma rota também poderão responder, embora esse mecanismo ainda não tenha sido especificado. O mesmo ocorre se for conhecido algum valor prévio de número de saltos até o destino. Esses dois valores opcionais ajudam a obter a máxima eficiência do protocolo, porém podem ser omitidos, caso deseje-se reduzir o tamanho da mensagem. Em seguida, o nó deverá inserir o seu IP e outros campos opcionais que desejar na mensagem. O número de saltos da mensagem deve ser inicializado com zero e o número máximo de saltos com o diâmetro da rede ou um valor inferior.

2.7.6 Route Reply (RREP)

Um nó deve incrementar o seu número de sequência ao emitir um RREP sempre que o campo opcional do RREQ, o Target.SeqNum, que contém o número de sequência

da última rota para o destino, esteja em um dos casos: não preenchido, com valor zero ou for maior que o seu próprio número de sequência. Também deve ser incrementado quando o Target.SeqNum for igual ao próprio número de sequência e, o último registro do número de saltos conhecido pela origem para o destino (Target.HopCnt) é desconhecido ou é conhecido e maior que Target.HopCnt.

Em seguida, ele deve preencher o campo de destino do RREP com o endereço de origem do RREQ, o campo do número de sequência com seu próprio número de sequência e colocar o seu IP no campo de origem. Os campos opcionais podem ser preenchidos segundo opção do emissor, o campo do número de saltos é inicializado com zero e o número máximo de saltos com o diâmetro da rede.

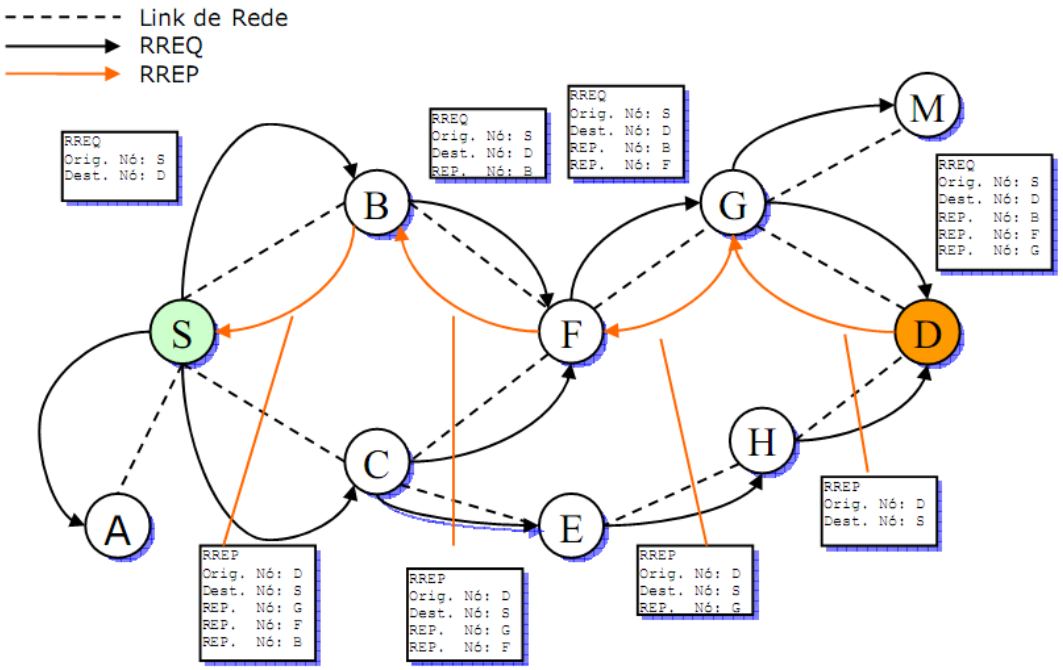


Figura 2.7-a: Geração de RREQ e RREP por uma rede utilizando DYMO.

A Figura 2.7-a mostra o preenchimento dos pacotes RREQ e RREP.

2.7.7 Processamento das Mensagens de Roteamento

Após processar ou criar mensagens de roteamento, um nó pode inserir novas informações de roteamento para reduzir o número de processos de descobrimento de rotas, como custo de enviar mensagens mais longas. Cada nó intermediário verifica se as informações contidas na mensagem podem ser utilizadas para atualizar sua tabela de

roteamento. Se as novas informações não existirem na tabela, são adicionadas, e, se já existirem e forem mais novas que as da tabela, são atualizadas. Se forem desatualizadas, devem ser retiradas da mensagem, antes de reencaminhá-la. Se toda a informação gerada pelo emissor contida na mensagem de roteamento for desatualizada, a mensagem deve ser descartada. Além disso, se o tamanho da tabela de roteamento da mensagem não for limitado, todas as rotas atualizadas podem ser inseridas na mensagem como atualização, evitando o envio de novos RREQs. Primeiramente, o nó insere o seu próprio IP e número de sequência, além de outras informações próprias que julgar necessárias. Para inserir essas informações, o seu número de sequência deve ser incrementado de acordo com as regras mencionadas. As demais rotas devem ser acrescentadas ativando o bit que informa que elas não pertencem ao caminho entre origem e destino. É importante acrescentar que não podem existir diferentes entradas para a mesma rota na mensagem, devendo manter-se sempre a rota mais atualizada.

Ao receber uma mensagem de roteamento, o nó deve decrementar o limite de saltos de um e incrementar o número de saltos com um. O mesmo deve ser feito para os endereços adicionais incrementados à mensagem que também possuam número de saltos.

O encaminhamento das mensagens de roteamento deve seguir a seguinte regra:

- Se o número limite de saltos for maior que um, o nó não é o destino e se trata de um RREQ, então o nó deve enviar a mensagem para todos os nós no seu alcance.
- Se o número limite de saltos for maior que um, o nó não é o destino e se trata de um RREP, então o nó deve enviar a mensagem para o IP indicado como próximo salto na sua tabela de roteamento.
- Se o nó é o destino e a mensagem é um RREQ, ele deve responder com um RREP.
- Se o nó é o destino e a mensagem é um RREP, ele deve apenas armazenar a informação. Em todos os casos, sempre que a mensagem passar pelos nós intermediários, esses devem armazenar a rota para o emissor da mensagem passando pelo último nó que a encaminhou.

No caso da rede possuir um *gateway* para a Internet, esse deverá responder aos RREQs para nós fora da rede, assim como será responsável pelo encaminhamento de pacotes para destinos exteriores à rede.

2.7.8 Descobrimento, Manutenção e Monitoramento de Rotas

Sempre que um nó desejar descobrir uma rota, ele deve inundar a rede com um RREQ, e esperar até que a rota seja formada por um tempo RREQ_WAIT_TIME.Se neste tempo não foi obtida nenhuma resposta, o nó deve realizar uma nova descoberta utilizando um *backoff exponencial binário*[13]. Através desse, o tempo de espera por resposta será multiplicado por dois a cada falha no processo, que deve continuar até um número RREQ_TRIES de vezes. Após esse número de tentativas, uma mensagem ICMP de destino inalcançável deve ser enviada a aplicação.

O monitoramento de enlaces é essencial para uma boa manutenção de rotas e pode ser feito utilizando mecanismos como o *feedback*[14] da camada enlace, descobrimento de vizinhos, tempo de expiração de rotas, entre outros. Após a descoberta de um enlace falho, o nó deve invalidar todas as suas rotas que o utilizam.

A manutenção das rotas é feita com atualizações do tempo de expiração e com o uso de RERRs. As atualizações de tempo de expiração devem ser feitas sempre que um novo pacote de dados atravessar aquele nó, reiniciando a contagem do tempo de expiração. A mensagem de RERR deve ser enviada por inundação sempre que um pacote de dados ou controle é recebido e não pode ser encaminhado para o próximo enlace, devido à ausência de uma entrada válida na tabela de roteamento. O RERR também pode ser gerado imediatamente após a detecção de um enlace quebrado de uma rota ativa, ou seja, que foi usada recentemente para o envio de dados. Um exemplo de uso de RERR é quando um RREP está sendo enviado e não existe uma rota para a origem.

[13] Algoritmo que dobra a faixa de atraso aleatório após cada colisão
[14] Um tipo de comunicação.

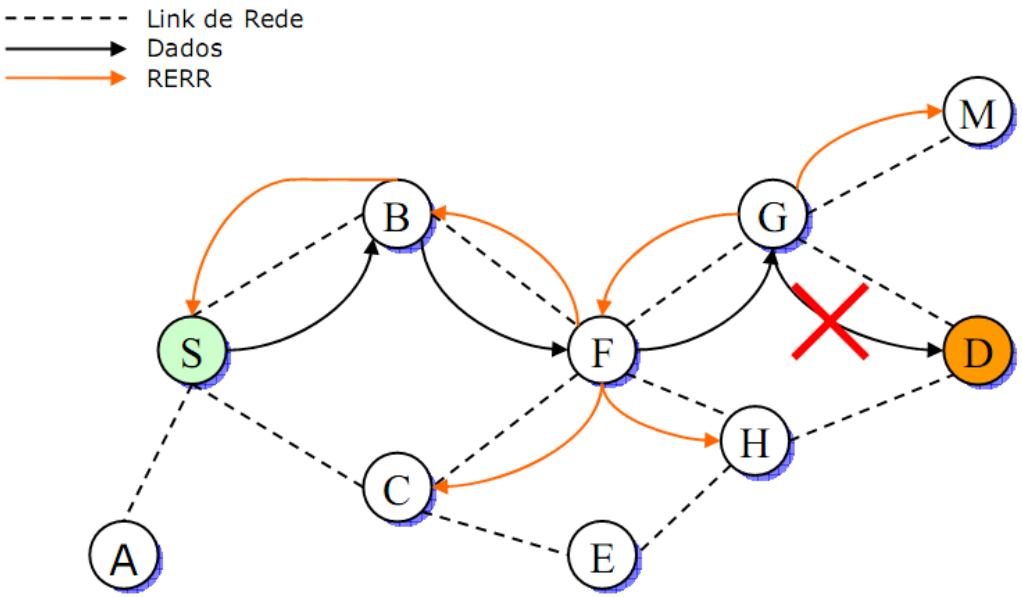


Figura 2.7-b: Envio de RERR em uma rede utilizando DYMO.

Ao receber uma mensagem de RERR (ver Figura 2.7-b), o nó a processa para atualizar o tempo de expiração das suas rotas atingidas pela mensagem para o momento atual. Cada entrada da mensagem que não resultar em uma mudança de tempo de expiração é retirado do RERR, já que a sua propagação não resultará em benefícios. Da mesma forma, todas as rotas associadas à rota que foi excluída também são excluídas. Enquanto o número de saltos máximo não for atingido e existir conteúdo na mensagem, ela é repassada.

2.7.9 Conclusão

O DYMO é um protocolo simples, que não garante o melhor caminho, mas sim, a existência de rotas sem *loops*. Suas mensagens são simples e buscam sempre reduzir ao máximo o número de mensagens de controle na rede. O ponto principal de seu funcionamento está no controle do número de sequência para manter as rotas atualizadas e na manutenção das rotas pelo monitoramento dos enlaces e envio de RERR.

Porém o DYMO, assim como a maioria dos protocolos reativos, tem um problema central: o tempo que o pacote fica no buffer do nó aguardando por uma rota. Esse problema é ainda mais agravado com essa operação que pode ser repetida

inúmeras vezes na fase de manutenção de rotas. Assim, o DYMO mostra-se ineficiente no que diz respeito ao tempo da entrega de dados.

O DYMO possui algumas implementações disponíveis, pois a sua simplicidade facilita o desenvolvimento do código, além de permitir que ele seja utilizado até em redes com poucos recursos.

3 Otimização Combinatória com Colônias de Formigas

3.1 Inteligência Coletiva e Otimização Combinatória

Inteligência coletiva é segundo [26] um tipo de inteligência artificial baseada no comportamento coletivo de sistemas descentralizados e auto-organizados. Inteligência coletiva é criada tipicamente através de uma população de agentes simples interagindo localmente com outros, dentro do seu ambiente. Os agentes geralmente seguem regras bastante simples. Embora não haja uma estrutura de controle centralizada ditando como os agentes devem se comportar localmente, as interações entre os agentes levam ao surgimento de um comportamento global inteligente, mesmo desconhecido para os mesmos.

A inteligência coletiva tem inspiração biológica, pois baseou-se no comportamento coletivo de sociedades de insetos como formigas, cupins, abelhas e vespas, bem como de outras sociedades de animais como bando de pássaros e cardume de peixes, dentre outros. O estudo do comportamento social de insetos tem atraído muitos pesquisadores durante muitos anos, mesmo assim o mecanismo que rege suas interações ficaram por muito tempo desconhecidos. Embora os membros dessas sociedades sejam, isoladamente, indivíduos limitados, eles são hábeis em cumprir tarefas complexas quando em cooperação. O comportamento coordenado da colônia surge de ações relativamente simples, ou de interações entre os membros da colônia. A auto-organização do trabalho descentralizado são aspectos relevantes das sociedades dos insetos. Por exemplo, formigas cortam pedaços de folhas para levar de volta ao formigueiro, para que nas folhas, desenvolvam-se fungos usados como alimentação de

suas larvas. Outro exemplo inclui a capacidade de cupins e vespas construírem abrigos sofisticados, ou a habilidade de abelhas e formigas de orientar-se no ambiente [22].

Segundo [22] o termo inteligência coletiva (*swarm intelligence*) foi usado inicialmente por Beni no contexto de sistemas robóticos celulares, onde simples agentes organizam-se através da '*interação com o vizinho mais próximo*'.

Os métodos de inteligência coletiva têm grande sucesso na resolução de problemas para a área de otimização, os quais são de grande importância tanto para a ciência quanto para a indústria.

A comunidade científica tem simplificado alguns desses problemas a fim de obter testes científicos com o bem conhecido problema do caixeiro viajante – (Traveling Salesman Problem - TSP), que é um dos assim chamados *problemas-conceito*. No problema, um caixeiro viajante precisa visitar um número de cidades, visitando cada uma apenas uma vez, de tal forma que a distância percorrida seja a menor possível. Outro exemplo é o problema da dobradura da proteína, um dos mais desafiadores problemas da biologia computacional [22], onde se deseja encontrar a forma funcional ou configuração de uma proteína no espaço de duas ou três dimensões. O TSP e a dobradura da proteína pertencem a uma importante classe de problemas de otimização conhecidos como otimização combinatória (OC).

Devido à importância prática dos problemas de otimização, muitos algoritmos têm sido desenvolvidos. No contexto da OC, esses algoritmos são classificados em *completos* e *aproximados*. Os algoritmos completos *garantem* encontrar por cada instância de tamanho finito de um problema de OC uma solução ótima em um tempo determinado. Para problemas de OC onde não exista um algoritmo que encontre uma solução em tempo polinomial, são ditos NP-Hard. Assim, os métodos completos exigem a necessidade de processamento em tempo exponencial para vencer esses casos. Na prática, isso leva a necessidade de tempo de processamento elevado para resolver o problema, o que pode tornar inviável sua utilização.

Nos métodos aproximativos, como nos algoritmos de inteligência coletiva, a garantia de encontrar a solução ótima é sacrificada, pela opção de obter soluções satisfatórias, ou seja, boas soluções em tempo computacional significativamente

reduzido. Logo, o uso de métodos aproximativos tem recebido mais e mais atenção da comunidade científica nos últimos anos.

Uma das mais notáveis técnicas de inteligência coletiva para obtenção de soluções aproximativas para problemas de otimização em aceitável quantidade de tempo computacional é a Otimização de Colônia de Formigas (ACO – Ant Colony Optimization).

3.2 Otimização com Colônia de Formigas

Otimização com Colônia de Formigas [24][25][26] foi umas das primeiras técnicas para otimização aproximativa inspirada pela inteligência coletiva. Mais especificamente, a ACO é inspirada no comportamento forrageiro das colônias de formigas. O coração desse comportamento é a comunicação indireta (*stigmergy*[15]) entre as formigas através de uma trilha de feromônio químico, que lhes possibilita encontrar os menores caminhos entre o formigueiro e as fontes de comida. Esta característica das colônias de formigas é explorada nos algoritmos ACO a fim de resolver, por exemplo, problemas de *otimização discreta*[16] [22]. No contexto dos algoritmos de busca, os algoritmos ACO pertencem a classe das meta-heurísticas. Juntamente com ACO, outros algoritmos como, computação evolucionária, busca local iterada, têmpera simulada e busca tabu, são frequentemente utilizados.

3.2.1 O início da ACO

Marco Dorigo e Cols introduziram o primeiro algoritmo ACO no início dos anos 1990 [26]. O desenvolvimento desse algoritmo foi inspirado pela observação das colônias de formigas. Formigas são insetos sociais. Elas vivem em colônias e seu comportamento é governado pelo objetivo da sobrevivência da colônia acima da sobrevivência do indivíduo. O comportamento que deu a inspiração para ACO foi o *comportamento forrageiro das formigas*, e em particular, como as formigas podem encontrar caminhos mais curtos entre a fonte de alimento e o seu formigueiro.

Quando estão procurando por comida, as formigas inicialmente exploram a área ao redor do formigueiro de forma randômica. Quando se movimentam, as formigas

[15] Tipo de comunicação através do ambiente.
[16] Ramo de otimização aplicado a matemática e ciência da computação.

deixam uma trilha de feromônio químico por onde passam. Formigas podem distinguir através do cheiro a concentração de feromônio. Quando escolhem seu caminho, elas tendem a escolher, com certa probabilidade, caminhos marcados por forte concentração de feromônio. Logo uma formiga encontra uma fonte de alimento, ela avalia a quantidade e a qualidade da comida e carrega certa quantidade do alimento enquanto volta ao formigueiro. Durante a viagem de volta, a quantidade de feromônio que uma formiga deixa pelo caminho pode depender da quantidade e da qualidade do alimento encontrado. A trilha de feromônio guiará outras formigas para encontrar a fonte de alimento – comportamento conhecido como stigmergy – permitindo-lhes, dessa forma, encontrar o menor caminho entre seu formigueiro e a fonte de alimento. A Figura, baseado no trabalho [26], ilustra Coesta idéia.

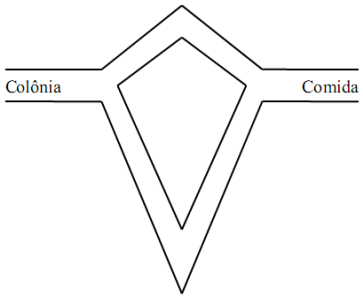


Figura 3.2-a: Todas as formigas estão na colônia. Não há feromônio no ambiente

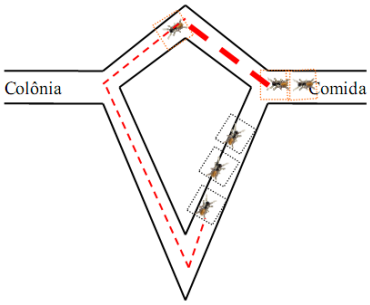


Figura 3.2-b: As formigas saem e escolhem caminhos diferentes

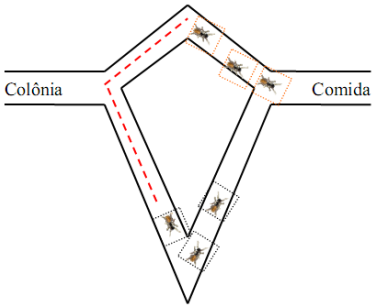


Figura 3.2-c: As formigas que escolheram o menor caminho chegam primeiro.

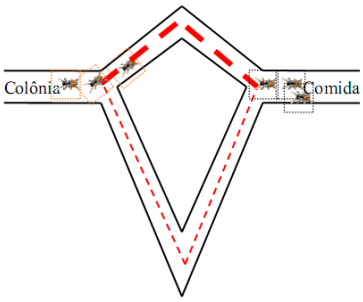


Figura 3.2-d: O menor caminho tem maior quantidade de feromônio.

Na Figura 3.2-a todas as formigas estão na colônia, não há feromônio no ambiente. Na Figura 3.2-b inicia-se o forrageamento, com probabilidade de 50% das formigas escolherem o menor caminho. Na Figura 3.2-c, as formigas que escolheram o menor caminho chegam antes à fonte de alimento. Assim, quando retornam, aumentam a probabilidade de escolha do menor caminho. Na Figura 3.2-d a trilha de feromônio sob o menor caminho é reforçada. Aliado ao processo de evaporação, o menor caminho tem sua probabilidade de escolha favorecida.

Em [22] é utilizado um grafo para representar o espaço de estados possíveis para o problema das formigas, conforme ilustram as figuras acima. Foi considerado um grafo $G = (V, E)$, onde V contém dois nós, denominados v_s (representando a colônia) e v_d (representando a fonte de comida). Além disso, E contém dois enlaces (conexões), e_1 e e_2, entre v_s e v_d, respectivamente de comprimentos l_1 e l_2, tal que $l_2 > l_1$. Em outras palavras, e_1 representa o menor caminho entre v_s e v_d. Formigas reais depositam feromônio sobre o caminho por onde passam. Assim, nas trilhas de feromônio químico, um valor artificial de feromônio τ_i para cada um dos dois links e_i, $i = 1,2$, indica a intensidade de feromônio na trilha sobre o caminho correspondente.

Considerando um grafo onde foi inserido em v_s uma colônia composta por η_a formigas artificiais, a probabilidade de cada formiga escolher um caminho, entre o caminho e_1 e e_2 para encontra a fontes de alimento v_d. é dada por:

$$P_i = \frac{\tau_i}{\tau_1 + \tau_2} \quad , i = 1,2, \qquad (3.1)$$

Obviamente, se $\tau_1 > \tau_2$, a probabilidade de escolher e_1 é mais alta e, vice e versa. Para retornar de v_d para v_s, uma formiga usa o mesmo caminho que ela escolheu para encontrar v_d, e incrementa o valor do feromônio artificial associado à aresta escolhida.

Em maiores detalhes, tendo escolhido a aresta e_1, o valor do feromônio é incrementado de acordo com a seguinte expressão como segue:

$$\tau_i \leftarrow \tau_i + \frac{Q}{l_i} \qquad (3.2)$$

onde Q, um parâmetro do modelo, representa um valor constante e positivo. Em outras palavras, a quantidade de feromônio artificial que é adicionado depende do tamanho do caminho de escolha: quanto menor o caminho, maior será a quantidade de feromônio adicionada.

O forrageamento de uma colônia de formigas é nesse modelo simulado iterativamente. A cada passo (ou iteração), todas as formigas são inicialmente colocadas em um nó v_s. Em seguida, cada formiga movimenta-se de v_s para v_d conforme esquematizado acima. Como indica a Figura 3.2-d, na natureza o feromônio depositado tende a evaporar com o passar do tempo. Finalmente, depois da etapa de evaporação, todas as formigas retornam e reforçam os caminhos escolhidos como esquematizado acima. Neste contexto, a evaporação do feromônio no ambiente artificial é representada pela seguinte expressão:

$$\tau_i \leftarrow (1-p)\cdot \tau_i, \quad i = 1,2 \qquad (3.3)$$

onde o parâmetro $p \in (0,1]$ é utilizado para regular a evaporação do feromônio no ambiente.

O trabalho [22] simulou este exemplo inicializando o em cada aresta com valor 0.5 e os seguintes parâmetros: l_1 = 1, l_2 = 2, Q = 1. Os resultados das simulações demonstraram em [24], claramente, que ao longo do tempo a colônia artificial de formigas converge para ao menor caminho, i.e., depois de algum tempo todas as formigas percorrem o menor caminho. Foram feitos experimentos com 10 e 100 formigas. No caso η_a = 10, a flutuação randômica é maior que no caso de η_a = 100, o que indicou que a habilidade de encontrar o menor o caminho por uma colônia de formigas depende da cooperação entre as formigas.

Ainda [22] indica as principais diferenças entre o comportamento de uma formiga real e o comportamento de uma formiga artificial, ou seja:

1. Enquanto formigas reais movem-se pelo ambiente de forma assíncrona, as formigas artificiais são sincronizadas, i.e., em cada iteração de um sistema simulado, cada formiga artificial move-se da colônia para a fonte de comida e seguem o mesmo caminho de volta.
2. Enquanto uma formiga real deixa feromônio sobre o caminho por onde se move, as formigas artificiais somente depositam feromônio artificial sobre o caminho de volta ao formigueiro.
3. O comportamento forrageiro das formigas reais são baseados em uma avaliação implícita de uma solução (i.e., um caminho do formigueiro para a fonte de alimento), ou seja, o menor caminho será completado antes dos caminhos maiores, os quais receberão reforço de feromônio mais rapidamente. Em contraste, uma formiga artificial avalia a solução com respeito a algumas qualidades mensuradas para determinar a intensidade do reforço do feromônio que as formigas depositam durante o caminho de retorno à colônia.

3.2.2 Ant System: O primeiro algoritmo ACO

O modelo que foi usado nas seções anteriores para simular o comportamento forrageiro das formigas reais, ilustrado nas **Figuras 3.1.a**–**d**, deve ser adequado para resolver problemas de otimização combinatória, já que valores de feromônio foram associados diretamente com a solução do problema (i.e. um parâmetro para o menor caminho, e um parâmetro para o maior caminho). Isso pressupõe que as soluções para o problema considerado já são conhecidas, o que não ocorre nos problemas de otimização combinatória. Assim, nestes casos, os valores de feromônio são associados com uma *solução componente*, ou seja, soluções, unidades, pelas quais as soluções de um problema sobre consideração podem ser montadas. Geralmente, o conjunto de *soluções componentes* é finito e de tamanho moderado.

Para exemplificar o primeiro algoritmo ACO, denominado *Ant System*(AS) [25], a seguir é apresentada a definição formal do problema do caixeiro viajante (TSP): *é dado um grafo completamente conectado e não direcionado G = (V, E) com pesos nas arestas. Os nós V (vértices) deste grafo representam as cidades, e as arestas valoradas representam a distância entre as cidades. A meta é encontrar um caminho fechado em G que contenha cada nó (cidade) exatamente uma vez (passeio ou viagem) e que o*

tamanho seja mínimo. Dessa forma, o espaço de busca S consiste de todos os passeios em G. O valor da função objetivo f(s) do passeio $s \in S$ *é definido como a soma dos valores das arestas de todas as arestas que estão em* s.

Com relação à abordagem AS, as arestas do grafo do TSP são consideradas *componentes da solução*, i.e., para cada $e_{i,j}$ é introduzido um valor de feromônio $\tau_{i,j}$. A tarefa de cada formiga consiste na construção de uma solução TSP possível, i.e., um passeio possível. Em outras palavras, a idéia de *tarefa de uma formiga* muda de "*escolher um caminho do formigueiro para a fonte de alimento*" para "*construir uma solução possível para resolver um problema de otimização*". Assim, com esta mudança de tarefa, as idéias da colônia e fonte de comida perdem o sentido.

Para construir uma solução, primeiro, um dos nós do grafo TSP é randomicamente escolhido como nó inicial. Então, cada formiga constrói um passeio no grafo do TSP pela movimentação, em cada passo executado, do nó corrente (i.e., a cidade na qual ela está localizada) para outros nós ainda não visitados. Em cada passo, a aresta percorrida é adicionada à solução em construção. Quando não existirem mais nós a serem visitados, a formiga fecha o passeio pelo movimento do nó corrente para o nó inicial e constrói a solução. Esta forma de construção de uma solução implica que uma formiga tem uma memória *T* para armazenar os nós já visitados.

Assim, assumindo que a uma formiga esteja em um nó v_i, o próximo passo, subsequente da construção, é escolhido com probabilidade de acordo com[22]:

$$p\left(e_{i,j}\right) = \frac{\tau_{i,j}}{\sum_{\{k \in \{1,\ldots,|V|\} | v_k \notin T\}} \tau_{i,k}}, \forall j \in \{1,\ldots,|V|\}, v_j \notin T \qquad (3.3).$$

Assim que todas as formigas da colônia completem a construção de suas respectivas soluções, o feromônio depositado é evaporado de acordo com a seguinte formulação:

$$\tau_{i,j} \leftarrow (1-p) \cdot \tau_{i,j} \quad , \forall \tau_{i,j} \in \tau \qquad (3.4)$$

Em seguida, as formigas realizam a viajem de retorno ao formigueiro. Tendo construído uma solução s, para cada $e_{i,j} \in s$ cada formiga deposita uma quantidade de feromônio expressa por:

$$\tau_{i,j} \leftarrow \tau_{i,j} + \frac{Q}{f(s)}, \qquad (3.5)$$

onde, Q representa uma constante positiva dada e $f(s)$ é o valor da função objetiva da solução s. Como explicado nas sessões anteriores, estes passos são repetidos, iterativamente para as η_a formigas, até uma condição de parada (e.g., um limite de tempo) seja satisfeita.

Mesmo que o algoritmo AS comprove que o comportamento forrageiro da formiga possa ser capaz de dar origem a um algoritmo de otimização combinatória, foram necessários mais alguns anos de trabalho e melhoramentos no algoritmo original AS para equipará-lo aos algoritmos no estado da arte. A próxima seção define um *framework ACO*, resultante do trabalho realizado nos últimos anos.

3.3 Otimização de Colônias de Formigas: Uma descrição geral

O framework ACO foi definido inicialmente por Dorigo et als em 1999 [26]. O recente livro de Dorigo e Stutzle mostra uma descrição mais abrangente [24]. A definição de um framework ACO trata de muitas – mas não de todas – variantes ACO existentes para problemas de otimização discreta. A figura abaixo apresenta uma descrição geral deste framework.

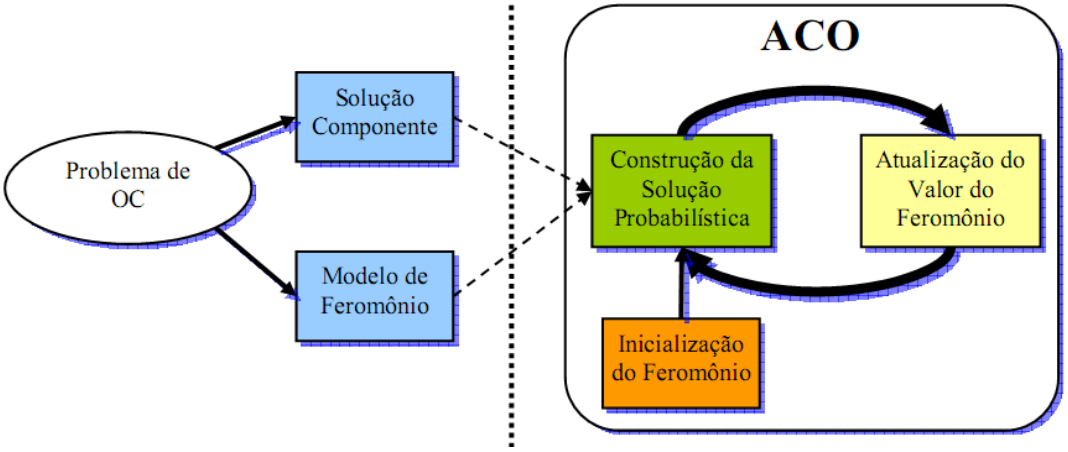


Figura 3.3-a: Framework ACO

O caminho básico do trabalho de um algoritmo ACO é graficamente mostrado na Figura 3.3-a[17]. Dado um problema de OC a ser resolvido, tem-se inicialmente que produzir um conjunto finito *C* de soluções componentes as quais serão usadas para montar a solução para o problema de OC. Segundo, deve-se definir um conjunto de *valores para o feromônio* T. Este conjunto de valores é frequentemente chamado de *modelo de feromônio*, o qual é – visto de um ponto de vista técnico – um modelo probabilístico parametrizado. O modelo de feromônio é um dos componentes centrais do ACO. Os valores de feromônio $\tau_i \in \mathrm{T}$ são usualmente associados aos componentes da solução. O modelo de feromônio é usado para probabilisticamente gerar soluções para o problema a ser construído pela sua montagem de um conjunto de *soluções componentes*. Em geral, a abordagem ACO tentar resolver um problema de otimização pela iteração de dois passos[22]:

- Soluções candidatas são construídas usando-se um modelo de feromônio, que é uma distribuição probabilística parametrizada sobre o espaço de soluções;
- As soluções candidatas são usadas para modificar os valores de feromônio nos caminhos que consideram mais aptos de chegar a caminhos de qualidade superior. O objetivo da atualização do feromônio é concentrar a busca nas regiões do espaço de busca contendo soluções superiores (alta qualidade). Implicitamente assume-se que soluções boas consistem de boas soluções componentes.

3.3.1 Construção da solução

Formigas artificiais podem ser vistas como uma heurística construtiva probabilística que monta soluções como uma sequência de *soluções componentes*. O conjunto finito de soluções componentes $C = \{c_1,...,c_n\}$ é por isso derivado de um problema de otimização. Por exemplo, no caso do AS aplicado ao TSP, cada aresta do grafo é uma *solução componente*. Inicialmente, cada solução é construída com uma sequência vazia $s = \langle\ \rangle$. Em seguida, a sequência corrente s é construída pela adição de uma solução componente possível em um conjunto $N(s) \subseteq C$. A especificação de $N(s)$

[17] Figura tirada de [22]

depende do mecanismo de soluções construído. No caso do algoritmo AS para o TSP, para cada formiga, o mecanismo de construção da solução restringe o conjunto de arestas visitáveis àquelas que estejam conectadas ao nó atual da formiga. A cada passo, a escolha de uma solução componente em $N(s)$ é realizada probabilisticamente levando em consideração o modelo de feromônio adotado. Em muitos algoritmos ACO a medida de probabilidade – também denominada probabilidade de transição – é definida por:

$$P(c_i \mid s) = \frac{[\tau_i] \cdot [\eta(c_i)]^{\beta}}{\sum_{c_i \in N(s)} [\tau_j]^{\alpha} \cdot [\eta(c_i)]^{\beta}}, \quad \forall\, c_i \in N(s), \qquad (3.6)$$

onde η é uma função valorada opcional, significando que, as vezes depende da sequência corrente, determinando em cada nó o passo da construção de um valor heurístico $\eta(c_j)$ para cada componente possível para a solução $c_j \in N(s)$. Os valores que são dados pela função valorada são frequentemente chamados de *informação heurística*. Além do mais, os expoentes α e β são parâmetros positivos que pertencem a um determinado valor de relacionamento entre a informação de feromônio e a informação heurística. Nas sessões anteriores que exemplificaram o TSP, a função valorada η não foi utilizada e foi atribuído a α o valor 1.

3.3.2 Atualização do feromônio

As variantes ACO diferem principalmente quanto aos mecanismos de atualização de feromônio empregados . Esta subseção apresenta uma regra geral de atualização do feromônio [22], consistindo de dois passos. Primeiro, é realizada uma *evaporação de feromônio* uniforme em todos os valores de feromônio. A evaporação do feromônio é necessária para evitar uma convergência rápida demais do algoritmo para uma região sub-ótima. A evaporação representa uma forma útil de '*esquecimento*', que favorece a exploração de novas áreas dentro do espaço de busca. Segundo, uma ou mais soluções da iteração corrente e/ou das iterações anteriores são usadas para incrementar os valores dos parâmetros das trilhas de feromônio sobre as soluções componentes que são partes destas soluções:

$$\tau_i \leftarrow (1-p) \cdot \tau_i + p \cdot \sum_{\{s \in S_{upd} \mid c_i \in s\}} w_s \cdot F(s), \text{ para } i = 1...., s \qquad (3.7)$$

onde S_{upd} denota o conjunto das soluções que são usadas para atualização, $p \in (0,1]$ é um parâmetro denominado *taxa de evaporação*, e $F : S \to \Re^+$ é denominada *função de qualidade* tal que $f(s) < f(s') \Rightarrow F(s) \geq F(s'), \forall s \neq s' \in S$. Em outras palavras, se o valor da função objetivo de uma solução s é maior que o valor da função objetivo de uma solução s', a qualidade da solução s será pelo menos igual a qualidade solução s'. A Expressão também permite um valor adicional da função de qualidade, i.e., $w_s \in \Re^+$ denota o valor de uma solução s.

Instâncias dessa regra de atualização são obtidas por diferentes especificações de S_{upd} e por diferentes configurações de valores. Em muitos casos, S_{upd} é composta de algumas das soluções geradas na respectiva iteração, S_{iter}, e a melhor solução encontrada desde o início do algoritmo, s_{bs}. Soluções s_{bs} são frequentemente chamadas de a *melhor solução até agora*. Como exemplo, vale mencionar a regra de atualização do AS, que pode ser obtida a partir de (3.8) fazendo-se $S_{upd} \leftarrow S_{iter}$ *e* $w_s = 1, \forall s \in S_{upd}$. Outro exemplo é a regra de atualização de feromônio denominada *atualização-IB* (*iteration-best*, ou melhor-iteração), onde e $w_{sib} = 1$ e $S_{upd} \leftarrow \{s_{ib} = \arg\max\{F(s) \mid s \in S_{iter}\}\}$, priorizando a escolha da melhor solução gerada na respectiva iteração para atualização dos valores de feromônio. Esta solução, denotada por s_{ib}, tem valor igual a 1. A regra de *atualização-IB* e a noção de boas soluções parece mais poderosa que a regra de *atualização-AS*, entretant aumenta o perigo de uma convergência prematura.

Por outro lado, na regra de *atualização-BS* (*melhor solução até o momento* s_{bs}), neste caso, S_{upd} assume $\{s_{bs}\}$, e s_{bs} tem valor 1 e $w_{sbs} = 1$. Na prática, os algoritmos ACO que usam a variação das regras de *atualização-IB* ou a *atualização-BS* e adicionalmente incluem mecanismo que evitam convergência prematura alcançam melhores resultados que algoritmos que usam a regra de *atualização-AS*. A próxima seção apresenta alguns exemplos.

3.3.3 Variantes ACO

Embora o algoritmo original AS tenha alcançado resultados animadores para o problema do TSP, ele é inferior aos algoritmos no estado da arte para o TSP assim como para outros problemas de OC. Assim, várias extensões e melhoramentos do algoritmo

original AS foram introduzidas nos últimos anos. Uma visão geral é mostrada na **Tabela 3-1** retirada da fonte [22]. Estas variantes ACO diferem principalmente na regra de atualização do feromônio aplicada.

Variante ACO	Autores
Elitist AS (EAS)	Dorigo
Elitist AS (EAS)	Dorigo, Maniezzo e Colorni
Rank-based AS (RAS)	Bullnheimer, Hartl e Strauss
MAX-MIN Ant System (MMAS)	Stutzle e Hoos
Ant Colony System (ACS)	Dorigo e Gambardella
Hyper-Cube Framework (HCF)	Blum e Dorigo

Tabela 3-1: Seleção de variantes ACO.

Adicionalmente para estas variantes ACO, a comunidade ACO tem desenvolvido características adicionais ao algoritmo para melhorar o processo de busca executado pelos algoritmos ACO. Um exemplo importante é a *estratégia de lista candidata*, que é um mecanismo para restringir o número de escolhas possíveis em cada passo de construção da solução. Usualmente, esta restrição aplica-se ao número de melhores escolhas com sua respectiva transição de probabilidades. Por exemplo, no caso da variante ACS para o TSP, a restrição para o fechamento de cidades em cada passo de construção tanto melhorou a qualidade da solução final quanto deu uma significativa velocidade ao algoritmo, pois: primeiro, a fim de construir soluções de alta qualidade, apenas as escolhas *promissoras* em cada passo da construção são oferecidas; segundo, aumenta-se a velocidade do processo de construção, pois a redução do número de escolhas reduz o tempo de processamento necessário para construção de uma solução.

3.4 Aplicações dos algoritmos ACO

Como mencionado anteriormente, ACO foi introduzida por meio da aplicação do problema conceito TSP. Desde então, os algoritmos ACO têm sido aplicados para diversos problemas de otimização. Inicialmente problemas clássicos como o TSP, assim como *problemas de atribuição, problemas de agendamento*, *coloração de grafos*, *problema de clique máximo*, e até mesmo *problemas de roteamento de veículos*. Mais recentemente o algoritmo gerou outras aplicações como, por exemplo, a localização de células existentes em projetos de circuitos, o projetos de redes de comunicação, para problemas de bioinformática e de otimização contínua. Nos últimos anos, alguns

pesquisadores têm focado aplicações de algoritmos ACO para problemas multiobjetivo e problemas estocásticos ou dinâmicos.

3.4.1 Protocolos de roteamento para MANETs

A idéia básica por trás dos algoritmos ACO para roteamento está na aquisição de informações de rotas através de amostras de caminhos usados por agentes formigas. Estes agentes simples são gerados ao mesmo tempo e de forma independentes dos nós, com a tarefa de procurar ou informar um caminho para um dado destino. Uma formiga indo de uma origem *o* para um destino *d* coleta informações sobre a qualidade do caminho (e.g. atraso fim-a-fim e número de saltos), e pode reconstruir o caminho de volta de *d* para *o*, usando esta informação para atualizar as tabelas de roteamento em cada nó intermediário. Formigas sempre fornecem caminhos completos.

A tabela de roteamento da formiga, chamada de *tabela de feromônio*, contém para cada destino um vetor de valores de entrada, sendo um para cada vizinho conhecido. Estas entradas são uma medida de qualidade do caminho sobre esse vizinho para o destino. Estas variáveis feromônio são continuamente atualizadas de acordo com a qualidade dos caminhos encontrados pelas formigas. A geração repetida e simultânea de resultados obtidos pelas formigas em cada nó de um conjunto de caminhos possíveis é dada através de uma medida de qualidade. Uma após a outra, as formigas usam as tabelas de feromônio para encontrar ou informar o caminho para um destino: em cada nó elas escolhem estocasticamente o próximo salto, dando alta probabilidade para aqueles saltos com altos valores de feromônio associados.

Algoritmos de roteamento baseados em formigas têm algumas propriedades que são desejáveis em redes móveis *ad hoc* - MANETs - , ou seja: são altamente adaptativos às mudanças da rede, usam amostras de caminhos ativos, são robustos em falhas de agentes, oferecem um roteamento com múltiplos caminhos, e podem cuidar da difusão de dados. Porém, o fato deles contarem com repetidas amostras de caminhos podem causar uma sobrecarga significativa se não tratadas com cuidado. Há um grande número de tentativas para modelar algoritmos de roteamento baseado em formigas para MANETs. A subseção a seguir apresenta alguns exemplos e suas respectivas referências.

3.5 Tabela de protocolos

Após um amplo levantamento bibliográfico, a **Tabela 3.2** foi compilada com um grande número de protocolos de roteamento baseados em formigas para MANETs.

Algoritmo	Referência
ARA - The Ant-Colony Based Routing Algorithm for MANETs	[27]
ANTHOCNET - Ant Agents for Hybrid Multipath Routing in Mobile *Ad hoc* Networks	[28]
Ant-AODV - Mobile Agents based Routing Protocol for Mobile *Ad hoc* Networks	[29]
Ant-DSR - Cache Maintenance Based Routing Protocol for Mobile Ad-Hoc Networks	[30]
ADAA - Application Research Based Ant Colony Optimization for MANET	[31]
ARAMA - Ant Routing Algorithm for Mobile Ad-hoc networks	[32]
EARA - A Swarm Intelligence Routing Algorithm for Manets	[33]
EARA-QoS - A Biologically Inspired QoS Routing Algorithm for Mobile *Ad hoc* Networks	[34]
HOPNET – A hybrid Ant Colony Optimization Routing Algorithm for Mobile *Ad hoc* Networks	[35]
IAQR - An Improved Ant Colony QoS Routing Algorithm Applied to Mobile *Ad hoc* Networks	[36]
MABR - Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks	[37]
PACONET - imProved Ant Colony Optimization routing algorithm for mobile *ad hoc* NETworks	[38]
PERA - A Probabilistic Emergent Routing Algorithm for Mobile *Ad hoc* Networks	[39]
POSANT - A Position Based Ant Colony Routing Algorithm for Mobile Ad-hoc Networks	[40]
SACO - An Experimental Study of the Simple Ant Colony Optimization Algorithm	[41]
SIO-AODV - A Biologically Inspired Optimisation to AODV Routing Protocol	[42]
ACODV_Ant - Colony Optimisation Distance Vector Routing in *Ad hoc* Networks	[43]

Tabela 3-2: Algoritmos ACO para redes *Ad hoc*.

Ant-Colony-Based Routing Algorithm (ARA) é um protocolo de roteamento para MANETs que usa a meta-heurística ACO. O algoritmo ARA não é baseado em um protocolo existente, porém é reativo. Dessa forma implementa as duas grandes fases que caracterizam esse tipo de algoritmo, a fase da descoberta e a fase da manutenção de rotas. Na fase de descoberta de rotas são criadas duas formigas artificiais FANT e BANT. A formiga FANT é responsável pela construção de uma trilha de feromônio artificial da origem para o destino, enquanto a formiga BANT cria uma trilha no sentido contrario, ou seja, do destino para a origem. Este algoritmo não utiliza nenhuma formiga especial para fase de manutenção. Ademais os pacotes de dados cumprem a tarefa de reforçar a rota escolhida.

O modelo de feromônio artificial do ARA, que é o modelo probabilístico de transição e de reforço dos caminhos escolhidos, é baseado no número de saltos. Através de um procedimento de busca uma trilha de feromônio artificial é criada por uma FANT da origem ao destino. A formiga BANT cumpre o caminho inverso, ou seja, depois que a FANT atinge o nó de destino, ela é destruída e é criada uma BANT que através de outro procedimento de busca, semelhante ao da FANT encontrará a origem, criando ou reforçando uma trilha de feromônio do destino a origem. Adicionalmente um reforço de

feromônio é adicionado à trilha do caminho escolhido pela origem para enviar os pacotes de dados ao destino.

Ant On-Demand Distance Vector (Ant-AODV) é um algoritmo de roteamento híbrido que utiliza o protocolo para MANETs AODV como base para sua implementação, não mudando nenhuma de suas características nativas. AODV cumpre a fase reativa do protocolo Ant-AODV. Já o algoritmo de formiga introduzido cumpre a fase proativa, assim ele o utiliza de forma independente do algoritmo AODV. O principal objetivo do algoritmo de formiga é criar rotas durante todo o tempo em que a rede existir para tentar diminuir o atraso fim-a-fim e a latência da rede. Assim sempre que o AODV buscar uma rota para um destino desconhecido a probabilidade de encontrar essa rota mais rapidamente é incrementada.

O modelo de feromônio artificial do Ant-AODV é baseado no número de saltos e tem como objetivo principal "descobrir a topologia da rede", sem nenhuma outra função específica, diferentemente da grande maioria dos algoritmos ACO.

3.6 Conclusão

Algoritmos baseados em formigas têm-se mostrado adequados para resolver muitos problemas de roteamento, em particular em redes *ad hoc*, contribuindo na melhoria da performance computacional em uma grande variedade de cenários.

Os protocolos para redes *ad hoc* que utilizam ACO mostram um certo êxito na tarefa de descoberta de caminhos dentro de um ambiente dinâmico.

Dos protocolos mencionados acima alguns foram fundamentais na concepção da do protocolo A-DYMO. O protocolo ARA mostrou como adaptar os mecanismos do ACO de forma eficiente através de uma abordagem reativa. Já o protocolo Ant-AODV, por sua vez, mostrou que uma a abordagem proativa tem suas vantagens. Dessa forma surgiu a idéia de utilizar uma abordagem híbrida, combinando o que há de melhor nos dois paradigmas.

4 A-DYMO

Dois dos três grandes grupos de protocolos para redes *ad hoc* apresentam limitações bastante evidentes. Nos protocolos de roteamento proativo a informação sobre toda a topologia da rede é mantida todo o tempo. Se por um lado a rota está sempre disponível, por outro há necessidade de se alocarem recursos para manter essas informações, o que consome grande parte da capacidade da rede para manter o fluxo do tráfego de controle, reduzindo, assim, a capacidade disponível da rede para efetiva comunicação de dados.

Já nos protocolos de roteamento reativos o problema de tráfego de controle não existe, pois as informações sobre rotas só são mantidas quando há necessidade de entregar pacotes de dados. Porém, como os protocolos reativos utilizam uma fase de descoberta de rotas, os pacotes ficam muito tempo no buffer aguardando uma rota. Isto faz com que o uso desse tipo de protocolo não seja desejável em grande número de aplicações existentes, tais como aplicações de tempo-real e comunicações multimídia.

Assim, a abordagem híbrida se mostrou bastante eficiente, na medida em que ela utiliza o que há de melhor nos dois tipos de protocolos. Ou seja, a utilização da abordagem proativa através de pacotes pequenos (e de raio de atuação limitado, para não comprometer o desempenho da rede), porém que seja suficiente para prover um grande número de rotas durante todo o tempo de atividade da rede. Juntamente, com a manutenção completa de uma rota apenas quando houver necessidade de entrega de pacotes. Finalizando essa manutenção quando não houver mais dados a serem entregues.

Agentes móveis[18] baseados em colônias de formigas mostraram-se eficientes na tarefa de rotear dados e descobrir topologias, provendo, assim, alta conectividade aos nós participantes da rede. Porém, grande parte dos protocolos existentes baseados em formigas herdou um comportamento totalmente reativo, o que os remete aos mesmos problemas desta classe de protocolos.

A idéia por trás do protocolo A-DYMO é criar um mecanismo para superar os problemas acima descritos, utilizando um esquema híbrido de roteamento e um algoritmo baseado em colônias de formigas. Dessa forma, provendo alta conectividade, descobrindo novas opções de rotas, diminuindo a latência da rede e o atraso fim-a-fim.

Durante esta pesquisa observou-se que os objetivos acima descritos encaixam-se perfeitamente dentro de uma abordagem híbrida, pois assim, é possível tratar isoladamente e dar mais atenção aos problemas pertinentes a cada uma das estratégias proativa e reativa.

Para a fase proativa um esquema baseado unicamente em colônias de formigas foi utilizado. Nela as formigas levantam a topologia da rede, além de mapear vários caminhos sinalizando de forma reforçada aqueles mais promissores. Essas informações são colocadas à disposição dos agentes de busca através de uma tabela de rotas probabilística.

Para a fase reativa foi tomado como base o protocolo de roteamento reativo DYMO[3]. Esta escolha se justifica, pois esse protocolo une as características de grandes outros protocolos reativos [6][7][8] desenvolvidos até aqui. Durante a pesquisa foi observado a necessidade de adaptar o processo de descoberta de rotas, inserindo uma formiga de busca para interpretar os dados levantados na fase proativa ao invés de descobrir o caminho com o pacote de busca tradicional. Dessa forma foi possível o perfeito acoplamento entre as fases proativa e reativa.

Assim, o protocolo A-DYMO une as principais técnicas para redes *ad hoc*. Ele é um protocolo híbrido, pois assim exclui ou mesmo diminui problemas hereditários de outros paradigmas. E utiliza uma meta-heurística bem consolidada, que é a ACO, para trazer qualidade a tarefa de encontrar caminhos.

[18] Programa capaz de migrar entre dispositivos carregando e informando dados.

4.1 Descrição do modelo

Definido como um grafo G = (V, E) valorado, onde V representa um conjunto finito não vazio de nós, e E representa um conjunto de links conectando os nós em V e um par de nós que podem comunicar-se entre si diretamente pela presença de pelo menos um link. |V| e |E| representam um conjunto de nós e links respectivamente. Para cada link *e(i, j)* ∈ E, foi utilizado o parâmetro distância D(e) → R para descrever seu estado, onde R representa pesos positivos.

4.2 Definições

Definição 1: A-DYMO define dois tipos de formigas artificiais: *Formiga Exploradora* (EANT) que tem função de criar rotas para sua origem e *Formiga de Busca* (ARREQ) que irá procurar por um destino específico e dessa forma utilizar a tabela de probabilidade de transição. Semelhante a BANT de outros algoritmos [27][29], a EANT carrega a informação de destino tendo com objetivo criar trilhas de feromônio. Os nós interpretam o endereço de origem da EANT como endereço de destino.

O formato da EANT é definido como uma t-upla [endereço da origem, número de sequência, registro de rotas (é uma lista dinamicamente incrementado incluindo os nós intermediários passados por ela e o seu correspondente tempo), Número de saltos].

ARREQ é um pacote que une as características das FANTs e dos RREQs de outros algoritmos [27][31], tendo como objetivo principal procurar por um destino específico. Ele herda o formato do RREQ do DYMO, como descrito no capítulo anterior. Porém insere um mecanismo de busca probabilística levando em conta o nível de feromônio sob o caminho, ou seja, o valor do reforço contido na tabela do nó.

Definição 2: Definimos a probabilidade de transição da formiga localizada no nó **i** para o nó **j,** como próximo salto, pela seguinte fórmula:

$$p(i,j) = \begin{cases} \dfrac{\tau(i,j)}{\sum_{s \in N_i} \tau(i,s)} & se\ s \in N_i \\ 0 & caso\ contrário \end{cases} \qquad (4.1)$$

Na equação (4.1), $\mathbf{N_i}$ é o conjunto de nós vizinhos de um salto de **i. τ(i, j)** é a concentração de feromônio sobre o link ***e(i, j)***.

A probabilidade de transição ***p(i, j)*** satisfaz:

$$\sum_{j \in N_i} p(i, j) = 1 \qquad (4.2)$$

Definição 3: regras de atualização: quando uma EANT passa através de um link ***e(i, j)***, ***τ(i, j)*** é atualizado pela equação 3:

$$\tau(i, j) \leftarrow \tau(i, j) + \Delta\tau(i, j) \qquad (4.3)$$

$$\Delta\tau(i, j) = \frac{\alpha}{D(i, j)} \qquad (4.4)$$

$$\tau(i, j) \leftarrow (1 - q)\tau(i, j) \qquad (4.5)$$

Em (4.3), é o reforço adicionado a rota o qual recebeu uma formiga, ***Δτ(i, j)*** é calculado pela equação (4.4).

Em (4.4), ***α*** é um parâmetro ajustável que representa o grau de influência da distância ***D(i, j)*** para o feromônio, o qual pode ser usado para influenciar diferentes demandas de quantidade de saltos (Distância).

Em (4.5), é aplicado de forma semelhante às formigas reais um mecanismo de evaporação, o qual diminuirá com o tempo ***t***.

Esses parâmetros serão mais bem explicados no exemplo a seguir.

Definição 4: Valores dos parâmetros. Será necessário inicializar o feromônio após a primeira detecção de vizinhança, assim, a quantidade inicial será de ***τ(i, j)* = 1**, a quantidade inicial de formigas será dada através de um parâmetro a ser definido a seguir, baseado na quantidade de nós (os nós serão informados randomicamente no início da simulação se irão ou não criar formigas), **α = 1, q = 0,5 e *t* = *1***.

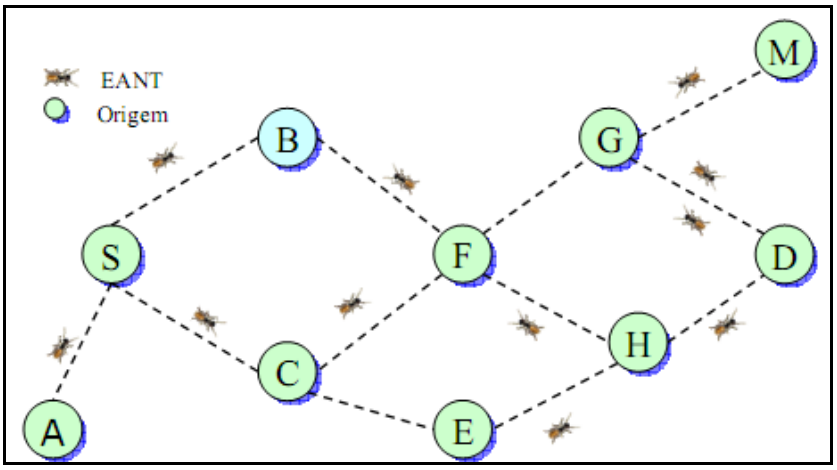


Figura 4.2-a: Operação de exploração de rotas pelas EANTs.

A Figura 4.2-a mostra a operação de uma rede em que nenhuma operação de busca de rotas está acontecendo. Assim as EANTs visitam os nós da rede provendo um caminho para sua origem. Dessa forma quando uma operação de busca iniciar-se encontrará nos nós rotas para vários caminhos.

4.3 Descrição do Protocolo

A-DYMO é um algoritmo híbrido e de múltiplos caminhos. Nele, o nó adquire as informações sobre sua vizinhança pela inundação limitada do protocolo Hello, como definido em [3]. De acordo com as respostas recebidas, cada nó cria sua tabela de probabilidades de rotas (semelhante com a tabela de feromônio em ACO) para substituir a tabela de rotas tradicional do nó. Podemos ver na tabela 1, que ***k*** e ***n*** são o número de vizinhos de um salto e os nós de destino do nó ***i*** respectivamente, e P^i_{nk} representa a probabilidade com a qual uma EANT localizada no nó ***i*** selecionará um de seus vizinhos l^i_k como próximo salto para um nó com destino N_n.

Destinos	**Vizinhos**			
	l^i_1	l^i_2	....	l^i_k
N_1	P^i_{11}	P^i_{12}	....	P^i_{1k}
N_2	P^i_{21}	P^i_{22}	....	P^i_{2k}
....	....	....	....	....
N_n	P^i_{n1}	P^i_{n2}	....	P^i_{nk}

Tabela 4-1: Tabela de Probabilidades

4.4 Exploração de Rotas

Logo após o início da simulação cada nó contendo uma EANT (Exploration ANT) a enviará por broadcast para seus vizinhos. Um nó que receber uma EANT pela primeira vez criará um registro em sua tabela de rotas. O registro na tabela de rotas adicionará o valor de 3 campos: *endereço de destino, próximo salto e valor de feromônio*. O nó interpretará o endereço de origem da EANT como endereço de destino, o endereço do nó anterior como próximo salto e atualizará o valor do feromônio do link *e(i, j)* sobre o número de saltos que a EANT atravessou para chegar até o nó. Em seguida o contador de saltos será atualizado e a EANT reenviada para seus vizinhos.

EANTs duplicadas serão identificadas através do número de sequência único e destruídas pelos nós.

Todo o processo acima não cessará até o final da simulação.

4.5 Descoberta de Rota

Quando nó S desejar enviar pacotes de dados para um destino D que não tem uma rota em sua tabela de rotas, ele criará uma ARREQ (Ant Route Requisition) e registrará o seu endereço, bem como o tempo em que a ARREQ foi criada no registro de rota da ARREQ. Então S enviará por broadcast a ARREQ para seus vizinhos.

- O nó *i* recebe a ARREQ pela primeira vez. O endereço de destino não é igual ao seu e o nó *i* não possui o caminho para o destino.
    - O nó *i* registra seu endereço e o tempo em que a ARREQ chegou no registro de rotas da ARREQ.
    - O nó *i* cria um registro em sua tabela de rotas. O formato do registro é como segue: *endereço de destino, próximo salto, valor do feromônio*. No registro, o nó *i* interpreta o endereço de origem da ARREQ como endereço de destino, o endereço do nó anterior como próximo salto, e calcula o feromônio pela equação 3.
    - O contador de saltos da ARREQ é atualizado por Ns = Ns + 1.
    - O nó *i* enviará por broadcast a ARREQ para seus vizinhos.
- O nó *i* recebe a ARREQ pela primeira vez. O endereço de destino não é igual ao seu, porém o nó *i* possui o caminho para o destino.

- A ARREQ irá escolher probabilisticamente a rota para o destino. Dentre as rotas presentes no nó *i*, com base na tabela 4-1.
- Então um RREP será criado e enviado para a origem com o endereço destino, bem como com a lista de nós participantes do caminho do destino.

- O nó *i* recebe a ARREQ pela primeira vez e o endereço de destino é igual ao seu.
  - Um RREP será criado e enviado para a origem com o endereço destino, bem como com a lista de nós participantes do caminho do destino. Da mesma forma que o algoritmo DYMO procede.
- O nó *i* recebe uma ARREQ duplicada.
  - Seguindo as mesmas regras do DYMO para RREQ a ARREQ será processada, dessa forma ARREQs duplicadas serão descartadas.

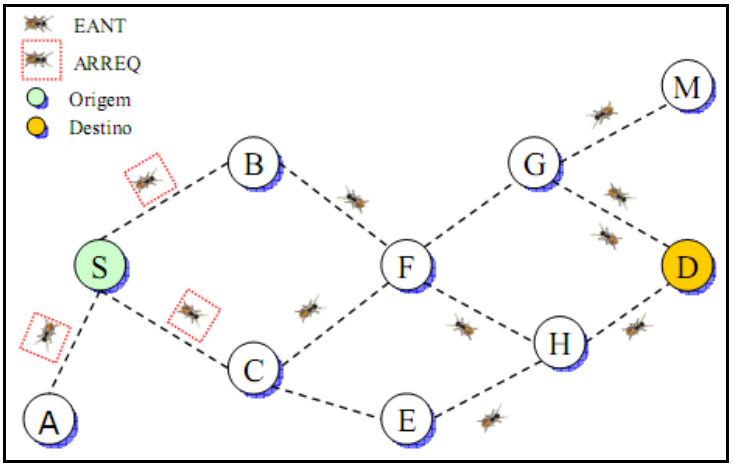


Figura 4.5-a: Protocolo total ARREQ + EANTs

A Figura 4.5-a mostra um processo de busca de rotas em uma rede utilizando o protocolo A-DYMO. Enquanto as EANTs criam rotas para seus nós de origem, as ARREQs iniciam o processo de busca por um caminho para o destino.

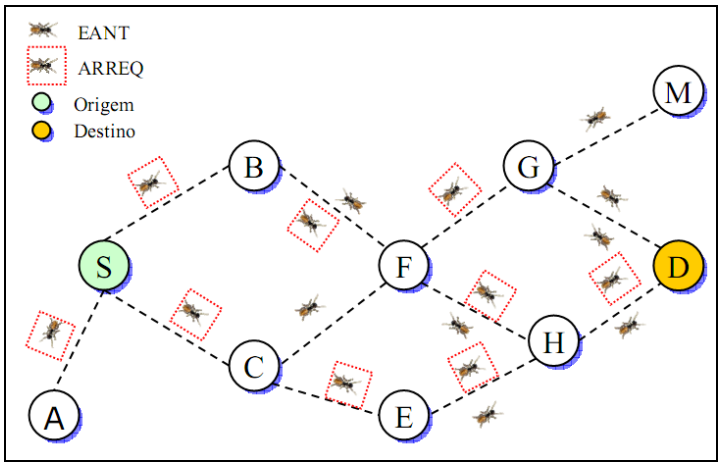


Figura 4.5-b: Busca por um destino

A Figura 4.5-b mostra o processo de busca de rotas utilizando A-DYMO. Aqui se destacam primeiramente as EANTs que continuam a prover rotas de forma independentes do processo de busca de rotas. Finalmente, as ARREQs transitam pela rede até encontrar o destino ou um nó que tenha o caminho para o destino, onde nesse caso, será utilizado o mecanismo de escolha probabilística de transição.

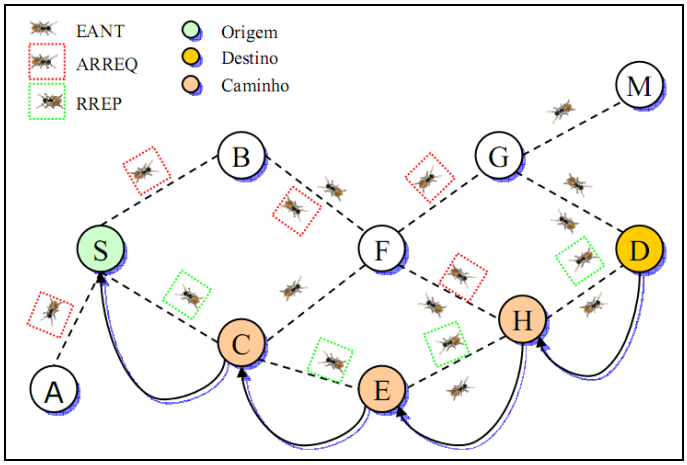


Figura 4.5-c: Geração de um RREP

A Figura 4.5-c mostra o processo de retorno de um RREP quando um nó de destino é encontrado. O RREP refaz o caminho reverso por unicast em direção da origem contendo a informação de roteamento para que os pacotes de dados sejam, enfim, enviados.

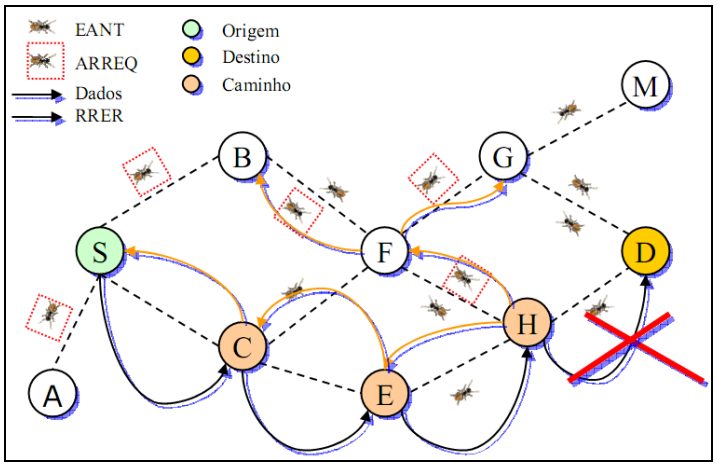


Figura 4.5-d: Erro de Rota

A Figura 4.5-d mostra o mecanismo de manutenção de rotas. Para esse mecanismo nenhum novo pacote foi definido. Assim um RRER é gerado de acordo com a descrição do capítulo anterior. E se a origem não possuir um outro caminho para o destino, um novo procedimento de descoberta de rotas é iniciado.

4.6 Manutenção de Rota

A fase da manutenção de rotas é a responsável pelo melhoramento das rotas existentes.

Em alguns algoritmos são gerados pacotes específicos para esse trabalho, porém, o A-DYMO não necessita de pacotes especiais. Por dois motivos: primeiro, as EANTS continuam trabalhando para prover rotas todo o tempo, o que aumenta a probabilidade de encontrar um caminho alternativo rapidamente quando houver erro de rota. E em segundo lugar, os pacotes de dados, imitando o comportamento das formigas reais, reforçarão as trilhas de feromônio sobre o caminho escolhido. Quando um nó S repassa um pacote de dados para o destino D, através de um nó vindo J, ele incrementa o valor do feromônio de entrada (D, J, τ) por, digamos $\Delta\tau$, assim o caminho para o destino é reforçado pelos pacotes de dados. Em contrapartida a isso, o próximo salto J incrementará o valor do feromônio pela entrada (S, I, τ) por, digamos $\Delta\tau$, assim o caminho para a origem também será reforçado. O processo de evaporação do feromônio real é simulado por um regular decréscimo dos valores de feromônio, o qual é executado pela equação 2.

Os demais processos de manutenção são nativos no DYMO.

4.7 Configuração do protocolo

Alguns parâmetros foram implementados de forma que podem ser configurados em tempo de execução. Ou seja, através da arquitetura interpretada do simulador.

Assim os parâmetros: quantidade de formigas (qtd_ants), tamanho da história (tam_hist), grau de influência da métrica de qualidade (alfa), podem ser ajustados de acordo com a necessidade de implementação.

O parâmetro qtd_ants diz respeito a quantas formigas serão inseridas no processo de descoberta de rotas. Trabalhos anteriores sugerem que a quantidade de formigas deva ser a mesma quantidade de nós. Porém essa abordagem mostrou-se capaz de sobrecarregar a rede, pois o tráfego gerado por essas formigas era superior ao desejado, o que diminuía o valor de seus benefícios. Trabalhos mais recentes mostram que essa quantidade pode ser diminuída, pois a operação de broadcast compensa a menor quantidade de formigas, na medida em que as multiplica de acordo com a quantidade de vizinhos. Assim, o parâmetro qtd_ants foi configurado inicialmente para n/2, onde n é a quantidade nós da rede.

Na fase proativa, a operação de exploração da rede por parte das EANTs pode causar uma sobrecarga indesejável, o que traria o problema dos protocolos proativos à tona. Assim, para resolver esse problema foi inserido o parâmetro tamanho da história. Esse parâmetro é responsável por limitar o raio de ação das EANTs, a fim de controlar a sobrecarga. Dessa forma o parâmetro tam_hist foi configurado inicialmente para 12, o que significa que após percorrer um caminho de 12 saltos as EANTs serão automaticamente descartadas.

O parâmetro alfa é responsável pelo grau de influência da métrica de qualidade, que nesse caso é a quantidade de saltos. Mesmo trabalhando com apenas um parâmetro de qualidade o protocolo está apto a receber outros parâmetros, necessitando apenas do ajuste no valor de seus parâmetros. O parâmetro de alfa foi configurado inicialmente com o valor 1.

4.8 Exemplo de atualização das tabelas de feromônio

O exemplo abaixo tem como objetivo mostrar o procedimento de atualização das tabelas de feromônio. Nesse exemplo, uma rede com quatro nós é criada onde a comunicação acontece em função de um nó central. Em seguida os procedimentos de atualização de feromônio são apresentados em duas iterações entre as formigas exploradoras.

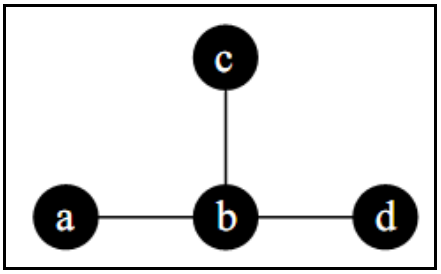


Figura 4.8-a: Exemplo de Atualização de feromônio

Round 1 – Primeiro broadcast
Movimentação de formigas:
a **(a1)** b
b **(b1)** a, b **(b1)** c, b **(b1)** d
c **(c1)** b
d **(d1)** b

Tabelas de feromônio

a		
Nó Dest.	Prox. Salto	τ
b	b	1

b		
Nó Dest.	Prox. Salto	τ
a	a	1
c	c	1
d	d	1

c		
Nó Dest.	Prox. Salto	τ
b	b	1

d		
Nó Dest.	Prox. Salto	τ
b	b	1

Tabela 4-2: Atualização de Feromônio depois da primeira passagem

Exemplo: Contas - Atualização do feromônio

EANTs

a **(a1)** b

a cria EANT **a1**.: **a1** = (a, 123, a, 1)

Formato da EANT [endereço da origem, número de sequência, registro de rotas (é uma lista dinamicamente incrementado incluindo os nós intermediários passados por ela e o seu correspondente tempo), Número de saltos].

$\alpha = 1$

$D(a,b) = 1$

$\tau(a,b) \leftarrow 0 + 1 / 1 = 1$ equação (3)

Round 2 – Segundo broadcast
Movimentação de formigas:
OBS1.: formigas em vermelho serão descartadas
a **(b1)** b
b **(a1)** a, b **(a1)** c, b **(a1)** d
b **(c1)** a, b **(c1)** c, b **(c1)** d
b **(d1)** a, b **(d1)** c, b **(d1)** d
c **(b1)** b, d **(b1)** b

Tabelas de feromônio

a		
Nó Dest.	Prox. Salto	τ
b	b	1
c	b	0,5
d	b	0,5

b		
Nó Dest.	Prox. Salto	τ
a	a	1
c	c	1
d	d	1

c		
Nó Dest.	Prox. Salto	τ
b	b	1
a	b	0,5
d	b	0,5

d		
Nó Dest.	Prox. Salto	τ
b	b	1
a	b	0,5
c	b	0,5

Tabela 4-3: Atualização de feromônio depois de segunda passagem

Exemplo: Contas - Atualização do feromônio

b **(a2)** c = 0 + 1 / 2 = 0,5 equação (3)
b altera a1.: a1 = (a, 123, ab, 2)
b insere seu endereço no pacote EANT e incrementa o número de saltos.
$\alpha = 1$
$D(c,a) = 2$
$\tau(c,a) \leftarrow 0 + 1 / 2 = 1$ equação (3)

4.9 Características do protocolo A-DYMO

Destacamos agora algumas características importantes que podem ser encontradas no protocolo A-DYMO que são herdadas de algoritmos de inteligência coletiva, em especial da Otimização de Colônia de Formigas e desejável em protocolos para redes *Ad hoc*:

- *Distribuição*: Um algoritmo ACO é distribuído, nele não existe um tipo de controle central. No A-DYMO vemos essa característica no trabalho de cada pacote EANT em informar novos destinos.
- *Paralelismo*: Os membros das colônias podem trabalhar ao mesmo tempo, o que geralmente melhora a performance das tarefas. Se as tarefas podem ser divididas em subtarefas os benefícios serão ainda melhores. EANTs e ARREQs trabalham em conjunto para prover novas rotas e topologias da rede.
- *Robustez*: Formigas ou agentes da colônia podem falhar e mesmo morrer sem afetar por completo a tarefa. Isto acontece devido a redundância de informações produzidas pelos agentes formigas.
- *Modularidade*: O problema pode ser analisado por partes, assim como o comportamento executado pelos membros da colônia. Criação de novas trilhas de feromônio, reforço das trilhas utilizadas, evaporação das trilhas velhas ou desatualizadas.
- *Escalabilidade*: Soluções para simples problemas podem ser estendidas também para grandes problemas.
- *Autonomia*: Indivíduos da colônia podem sentir e modificar o ambiente por conta própria.

Do ponto de vista da teoria da auto-organização, o comportamento social das formigas pode ser modelado com base em quatro elementos que são encontrado no A-DYMO:

- Reforço positivo:
    - Reforço no feromônio das rotas escolhidas;
- Reforço negativo:

- Evaporação do feromônio nas rotas menos usadas;

- Aleatoriedade:
  - Escolha probabilística das rotas para um dado destino;
- Múltiplas iterações
  - Um mesmo nó recebe a visita de várias EANTs e ARREQs para criar a topologia da rede e encontrar um dado destino, respectivamente.

4.10 Conclusão

O protocolo A-DYMO foi construído baseado em duas premissas: em primeiro lugar, o algoritmo de formiga mostrou-se hábil na tarefa de manter a topologia da rede na maior quantidade de tempo possível, bem como distribuir de forma probabilística o tráfego na rede, o que contribui com certo balanceamento de carga. Em segundo lugar, por se tratar de um protocolo híbrido, ele tem mais possibilidades de satisfazer as necessidades de aplicações reais [2], pois tanto trata o problema da latência da descoberta de rotas como também preocupa-se com o problema da sobrecarga na rede com relação ao tráfego de controle.

Baseado nas característica acima citadas, os resultados esperados são a diminuição do atraso fim-a-fim e da latência da rede com relação ao protocolo DYMO, que será o protocolo base de comparação. Isso possibilitará à rede consumir uma maior quantidade de recursos na efetiva entrega de dados e menos na descoberta e manutenção de rotas, o que dará maior vida útil aos nós componentes da rede.

5 Resultados: DYMO vs. A-DYMO

5.1 Ambiente de Simulação

O ambiente de simulação escolhido para testar os protocolos DYMO e A-DYMO foi o ns-2[5]. A versão utilizada do simulador foi a mais recente, ou seja, 2.34.

Ns-2 é um simulador de eventos discretos que tem como objetivo dar suporte à pesquisas em redes de computadores. O Ns-2 provê um suporte substancial para simulações de TCP, UDP, roteamento e protocolos multicast sobre redes cabeadas e redes sem fio (local e satélite).

O ns-2 é um simulador orientado a objeto, escrito em C++, com uma linguagem interpretada como *frontend* que é OTcl. O simulador é composto de duas hierarquias de classes, também conhecidas com hierarquia compilada, essa escrita em C++ e hierarquia interpretada escrita em OTcl[2].

Basicamente os protocolos são escritos na hierarquia compilada e sua manipulação é feita através de parâmetros na hierarquia interpretada. Também através da linguagem OTcl é possível criar e manipular cenários através de parâmetros contidos nos scripts.

5.2 Cenários e Métricas

Foram propostos vários cenários com objetivo de comparar as diferenças de comportamento do protocolo A-DYMO com o protocolo DYMO. Assim temos:

Cenários		
Parâmetro	**(a)**	**(b)**
Camada MAC	802.11	802.11
Modelo de Mobilidade	Random Waypoint	Random Waypoint
Tempo da Simulação	600s	600s
Protocolo de Transporte	UDP	UDP
Aplicação de tráfego	CBR	CBR
Tamanho do Cenário	1000m X 1000m	1000m X 1000m
Quantidade de Nós	50	20
Número de conexões	10	5
Tempo de Pausa	[0, 600]	[0, 600]
Velocidade	10m/s, 20m/s	10m/s, 20m/s
Tamanho do Pacote de dados	512 bytes	512 bytes
Intervalo de envio	0,25 s	0,25 s

Tabela 5-1: Cenários

5.2.1 Métrica 1: atraso fim-a-fim

Esta métrica inclui todos os possíveis atrasos causados pela latência da descoberta de rota, pela espera em fila, atraso causado por retransmissões, dentre outros. Ele é calculado da seguinte forma:

$$aff = \frac{\sum t}{\sum pkt\_dados\_suc} \qquad (5.1)$$

onde t é o tempo de entrega dos pacotes de dados e pkt_dados_suc, é o pacote de dados entregue com sucesso.

5.2.2 Métrica 2: Taxa de entrega

Relação de pacotes de dados entregues com sucesso para o destino por aqueles pacotes de dados gerados pelas origens CBR. O cálculo é como segue:

$$te = \frac{\sum pkt\_dados\_suc}{\sum pkt\_dados\_gera} \qquad (5.2)$$

onde, pkt_dados_suc denota todos os pacotes de dados entregues com sucesso e pkt_dados_gera representa todos os pacotes de dados gerados pelas origens CBR.

5.2.3 Métrica 3: Taxa de Perda

A taxa de perda descreve o total de pacotes de dados que, por algum motivo, não foram entregues aos seus destinos. Ele pode ser visto como um o somatório dos pacotes descartados por erro no pacote, colisões, descartados no final da simulação, contador excedido, dentre outros. Assim:

$$tp = \sum pkts\_desc \qquad (5.3)$$

onde, pkts_desc são os pacotes descartados.

5.2.4 Métrica 4: Sobrecarga de roteamento (Routing Overhead)

A sobrecarga de roteamento descreve quantos pacotes de roteamento são necessários na descoberta e manutenção de rotas para se propagar na rede pacotes de dados. Ele é dados por:

$$s = \sum pkts\_rot\_env \qquad (5.4)$$

onde, pkts_rot_env são os pacotes de roteamento gerados na rede na rede.

5.3 Parâmetros do protocolo A-DYMO – Hierarquia interpretada

Como visto no capítulo anterior, o protocolo A-DYMO traz uma série de parâmetros, a saber, **α, q e *t,*** que representam o valor da influência da métrica número de saltos, o valor do parâmetro de evaporação do feromônio e o intervalo da evaporação do feromônio, respectivamente.

Dessa forma, entendemos que é mais produtivo se pudermos atualizar esses parâmetros sem termos que recompilar todo o protocolo, para assim fazermos uma análise mais ágil. Diante dessa idéia cada um desses parâmetros, além de outros, ganharam uma ligação dentro da hierarquia interpretada, como veremos a seguir.

5.3.1 eants_percentage_

Esse parâmetro define o percentual de formigas que serão inseridas na rede. O cálculo é feito em função da quantidade de nós da simulação, assim:

$$qtd\_ants = qtd\_nos * eants\_percentage\_ \quad (5.5)$$

onde, qtd_ants é a quantidade de formigas a serem criadas na rede, qtd_nos é a quantidade de nós participantes da rede e eants_percentage_ é o percentual a ser aplicado, ele é dado na forma 0.x onde x = [1,9].

O processo de inserção das formigas é aleatório, assim a cada simulação as formigas são inseridas em diferentes nós.

5.3.2 eants_history_

Esse parâmetro define o tamanho da memória da formiga. Ou seja, a quantidade de nós que a formiga pode armazenar em sua estrutura.

Assim podemos controlar a qualidade das informações transmitidas, principalmente em relação a mudança de topologia, pois caminhos muito grandes têm maior probabilidade de conter rotas inválidas.

5.3.3 evaporation_factor_

É a ligação interpretada com o parâmetro ***q***. Esse parâmetro tem como função simular o processo de evaporação das formigas reais. Assim, diminui o valor do feromônio sobre todos os caminhos e tem como parâmetro o tempo.

5.3.4 eants_route_expiration_time_

Esse parâmetro indica com quanto tempo de inatividade a rota descoberta por uma formiga será expirada.

5.3.5 hopcount_factor_

Esse é o parâmetro ajustável que define o grau de importância da métrica hop para o algoritmo. Ele está ligado ao parâmetro compilado **α**.

5.3.6 eant_interval

Esse parâmetro ajusta o intervalo de tempo no qual o mecanismo de evaporação irá ser executado. Assim **eant_interval_** está ligado com o parâmetro compilado ***t***.

5.4 A-DYMO$_1$ e A-DYMO$_2$

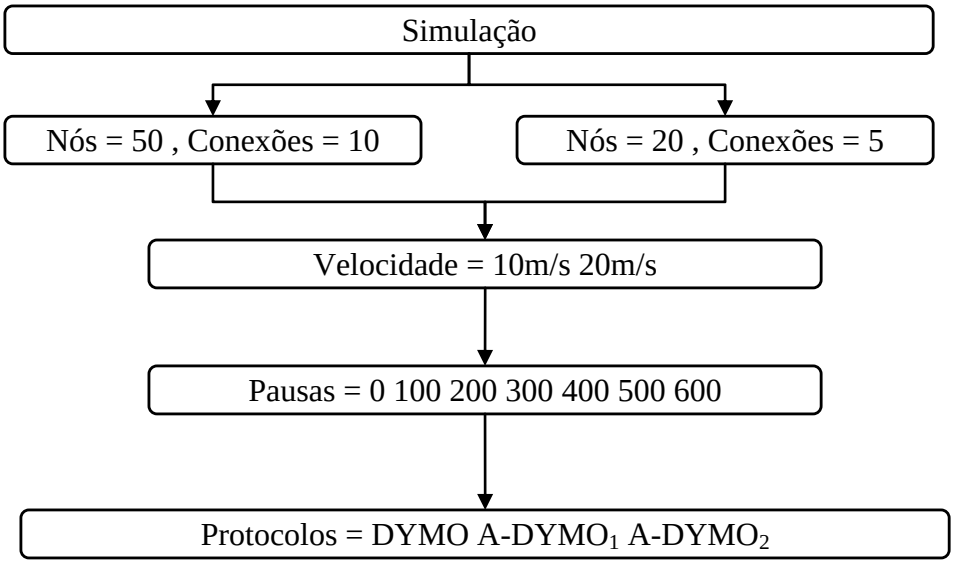
Através da configuração dos parâmetros vistos na sessão anterior foi possível criar algumas variações do protocolo A-DYMO. Dessa forma, este trabalho irá definir dois grupos de parâmetros para simulação que serão referenciados como A-DYMO$_1$ e A-DYMO$_2$. Os parâmetros estão definidos na tabela abaixo:

Parâmetro	**A-DYMO$_1$**	**DYMO$_2$**
eants_percentage_	0.3	0.6
eants_history_	12	20
evaporation_factor	0.5	0.1
eants_route_expiration_time_	10	10
hopcount_factor_	1.0	1.0

Tabela 5-2: Parâmetros A-DYMO

5.5 Resultado das simulações

As simulações foram feitas da seguintes forma:



Cada simulação foi repetida 10 vezes e o intervalo de confiança de 95% foi extraído. O módulo para extração do intervalo de confiança utilizado foi a biblioteca ConfInt v.1.0.1 [44] escrita em perl.

5.5.1 Atraso fim-a-fim

O atraso fim-a-fim é a métrica que diz a média de tempo que a rede gasta desde o momento em que um pacote de dados está no buffer do seu roteador, aguardando uma rota, até o momento de entrega deste pacote ao destino. Assim, ele compreende os atrasos totais da rede, tais como: latência, atraso de buffer, retransmissões, dentre outros.

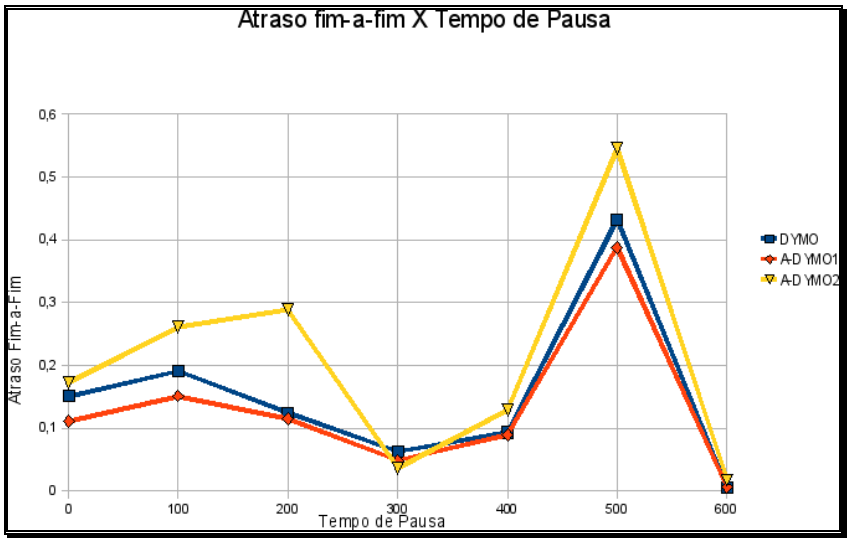


Figura 5.5-a: Atraso 20n x 5c x 10m/s

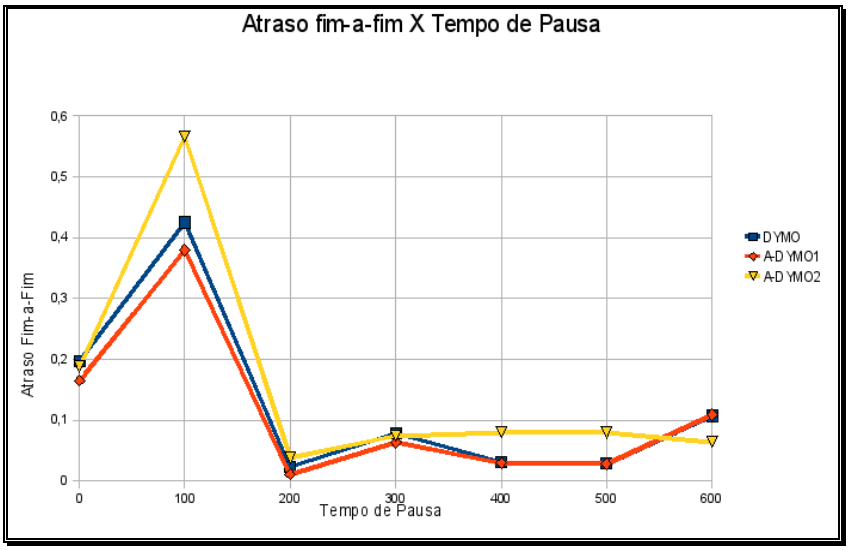


Figura 5.5-b: Atraso 20n x 5c x 20m/s

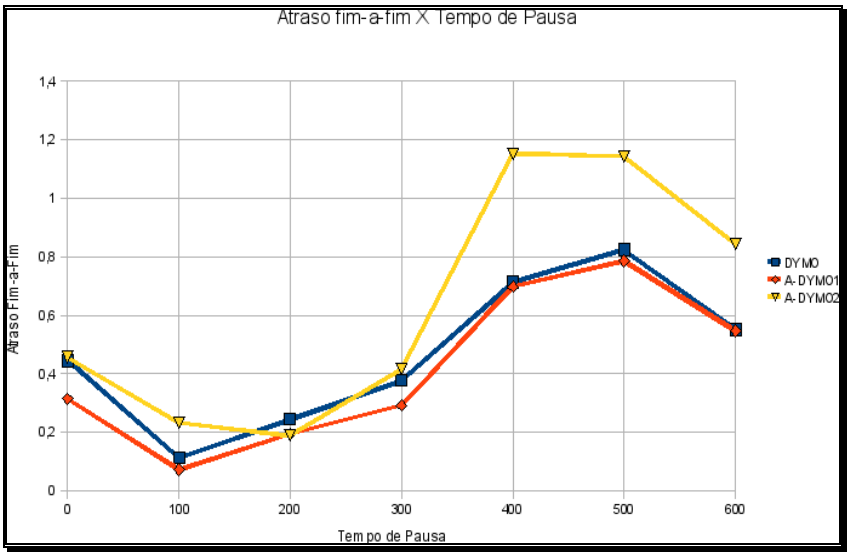


Figura 5.5-c: Atraso 50n x 15c x 10m/s

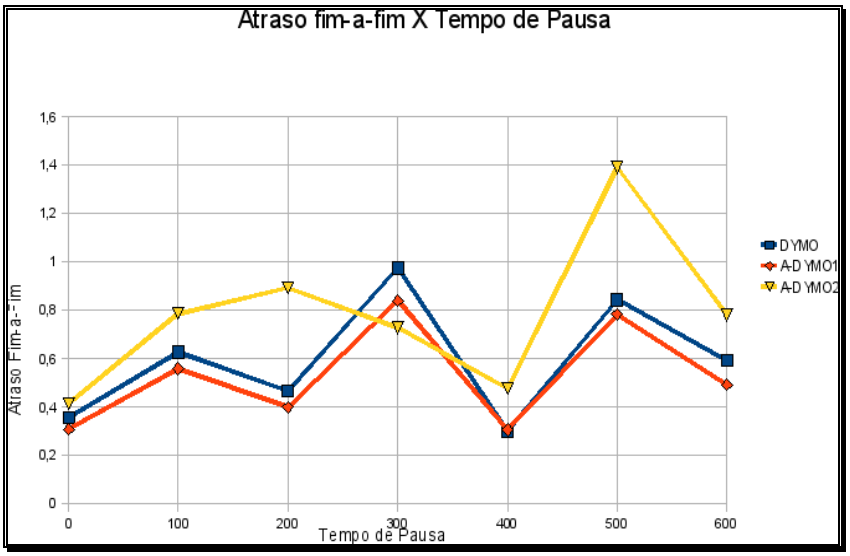


Figura 5.5-d: Atraso 50n x 15c x 20m/s

De acordo com os gráficos 5.5.1, o protocolo A-DYMO1, que utiliza parâmetros mais conservadores na configuração da meta-heurística Otimização com Colônias de Formigas, gasta menos tempo, na maioria das vezes, para entregar seus pacotes.

No gráfico 5.5.1-a, a média de ganho do protocolo A-DYMO1 em relação ao protocolo DYMO foi da ordem de 16,5%, com ganho máximo de 35% no tempo de pausa de 100s. No gráfico 5.5.1-b, a média de ganho foi de 10%, com valor máximo de 17% no tempo de pausa de 600s. A maior parte dos ganhos individuais aconteceu nos primeiros 400s de pausa, o que nos leva a concluir que o protocolo A-DYMO1 é mais hábil que o protocolo DYMO quando a rede apresenta maior mobilidade. Este ganho já era esperado no início da pesquisa, pois a topologia da rede mantida pelas EANTs,

mesmo antes da operação de busca de rotas, oferece maior probabilidade do nó de destino ser encontrado em menos tempo.

Também observa-se que o protocolo A-DYMO2 que utiliza parâmetros máximos na configuração da meta-heurística ACO, é o protocolo menos hábil em termos de tempo de entrega, isso parece acontecer pois o alto overhead causado por este protocolo ocupa boa parte da rede na manutenção do tráfego de controle.

5.5.2 Taxa de Entrega

Está métrica é uma das mais importantes na avaliação de um protocolo. Pois ela mostra o quão eficiente um protocolo é em termos de efetiva entrega de dados.

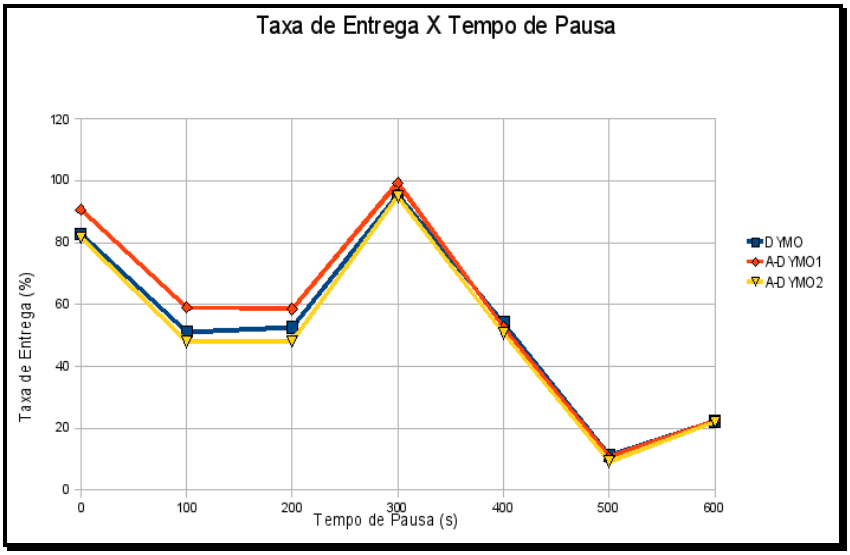


Figura 5.5-e: Entrega 20n x 5c x 10m/s

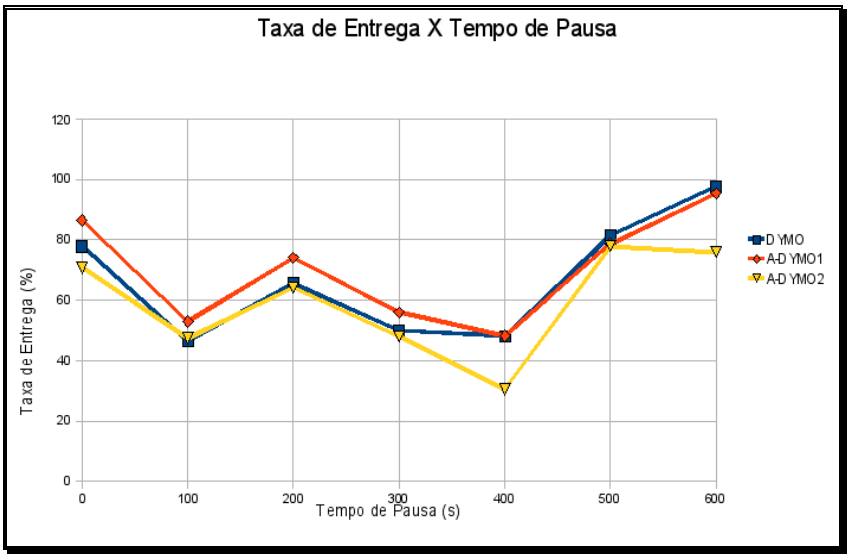


Figura 5.5-f: Entrega 20n x 5c x 20m/s

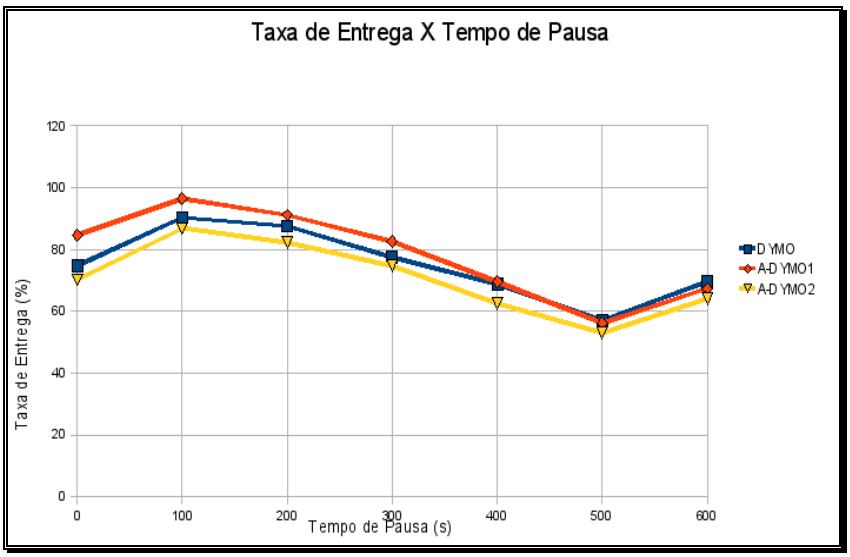


Figura 5.5-g: Entrega 50n x 15c x 10m/s

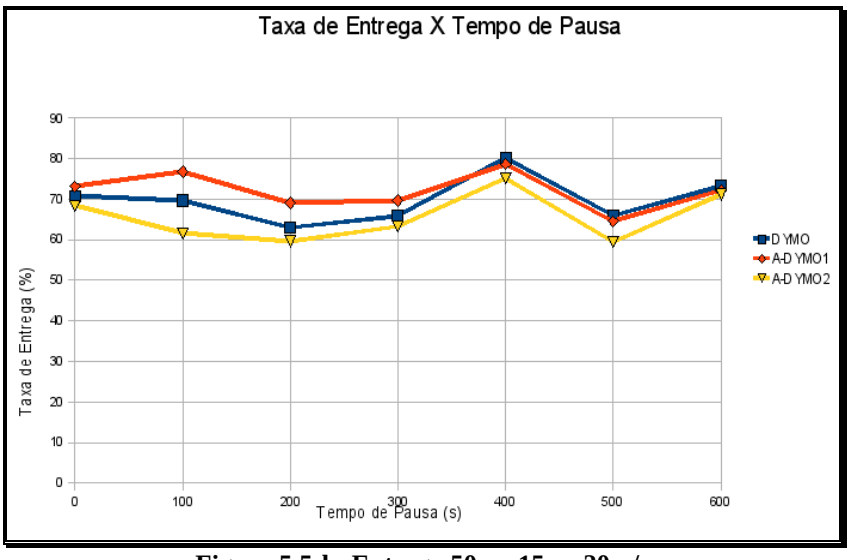


Figura 5.5-h: Entrega 50n x 15c x 20m/s

Os gráficos 5.5-e até 5.5-h mostram que o protocolo A-DYMO1 é mais hábil que os protocolos DYMO e A-DYMO2 em entregar os dados da rede na quase totalidade dos cenários propostos, principalmente nos cenários com até 400s de pausa. Os picos máximos de qualidade de entrega do protocolo A-DYMO1 em relação ao protocolo DYMO, são de 11%, 8%, 13,4% e 12,3% e acontecem nos respectivos cenários: Figura 5.5-e e de pausa é de 0s, Figura 5.5-f e tempo de pausa de 200s, Figura 5.5-g e tempo de pausa de 100s e finalmente Figura 5.5-h e tempo pausa de 100s.

O protocolo A-DYMO2 novamente teve problemas com o grande overhead causado por seus parâmetros.

5.5.3 Taxa de Perda

A taxa de perda descreve o somatório dos pacotes descartados por erro, colisão, tempo excedido, descarte no final da simulação, dentre outros.

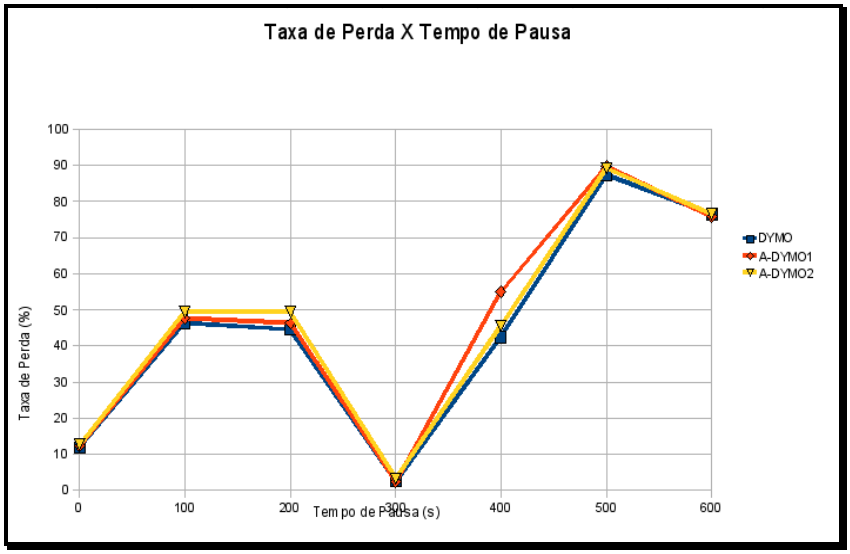


Figura 5.5-i: Perda 20n x 5c x 10m/s

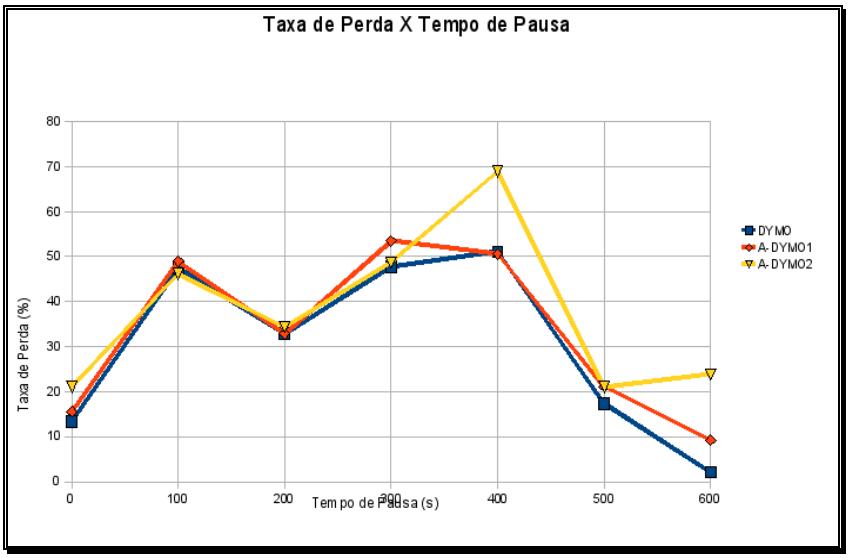


Figura 5.5-j: Perda 20n x 5c x 20m/s

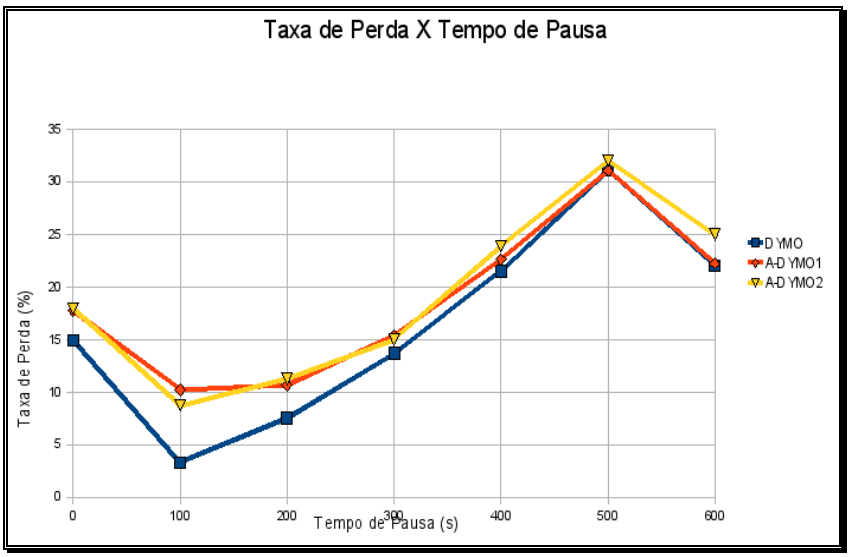


Figura 5.5-k: Perda 50n x 15c x 10m/s

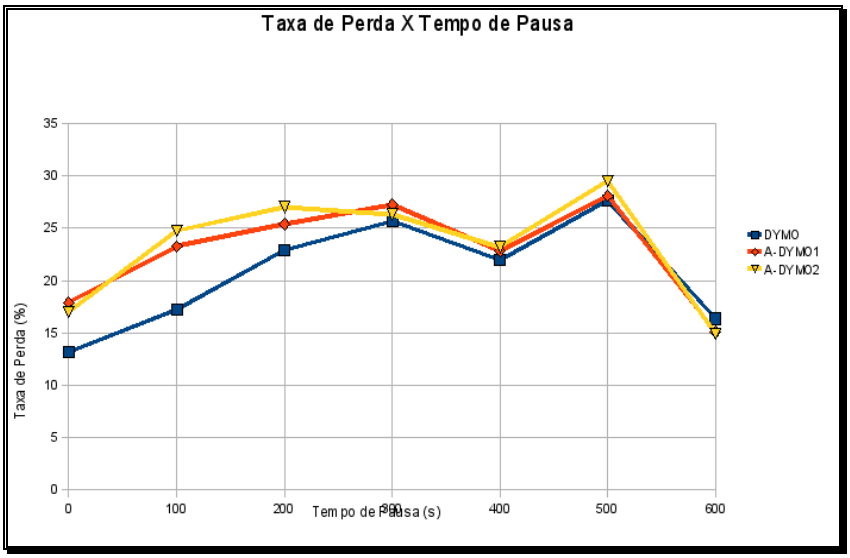


Figura 5.5-l: Perda 50n x 15c x 20m/s

O resultado do overhead extra criado pelo tráfego de controle dos protocolos A-DYMO1 e A-DYMO2 pode claramente ser observado nos gráficos 5.5-i até 5.5-l.

Quando o cenário tem menos nós, o que ocorre na Figura 5.5-i e Figura 5.5-j, a taxa de perda dos dois protocolos A-DYMO chega a ser igual a dos protocolo DYMO. Porém, quando a quantidade de nós é maior, caso da Figura 5.5-k e Figura 5.5-l, a taxa de perda aumenta na maioria absoluta dos casos, situação essa provocada pelo aumento do número de formigas e consequente aumento de overhead. As taxas de perda do protocolo A-DYMO1 em relação ao protocolo DYMO são em média 18,6%, 9,2%, 4,5% e 17,45, nas respectivas figuras: Figura 5.5-i, Figura 5.5-j, Figura 5.5-k e finalmente Figura 5.5-l. O desempenho do A-DYMO2 foi semelhante ao do A-DYMO1 na situação citada acima.

5.5.4 Sobrecarga de Roteamento – Overhead

O ponto mais vulnerável do protocolo A-DYMO (e já esperado por esta pesquisa) encontra-se nos gráficos 5.5-m até 5.5-p. São dois os motivos dessa vulnerabilidade: primeiro: o tráfego extra diretamente criado pelas EANTs; e segundo diz respeito às retransmissões criadas de forma indireta pelo mecanismo de memória que algumas vezes fornece caminhos desatualizados ou inexistentes.

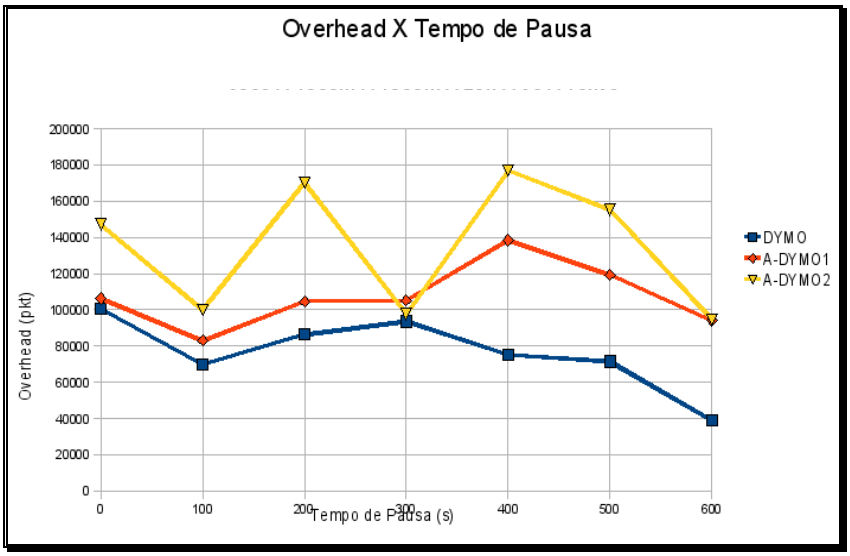


Figura 5.5-m: Overhead 20n x 5c x 10m/s

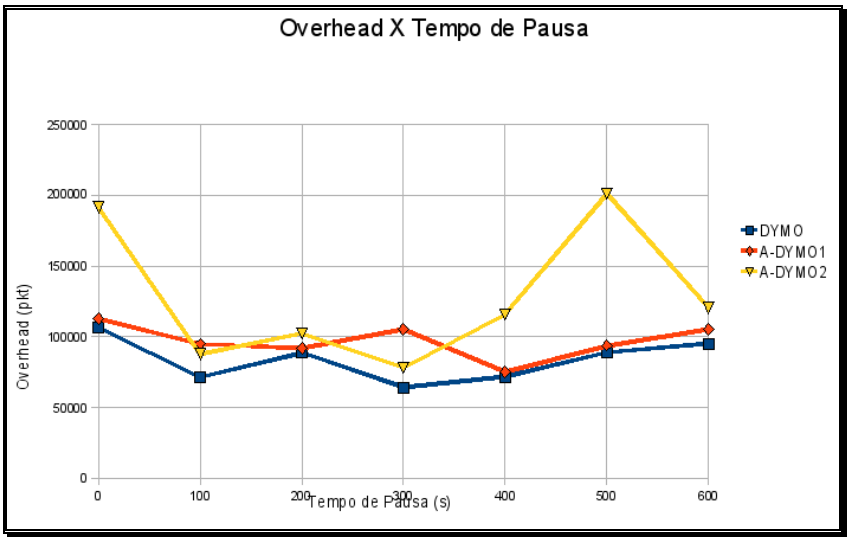


Figura 5.5-n: Overhead 20n x 5c x 20m/s

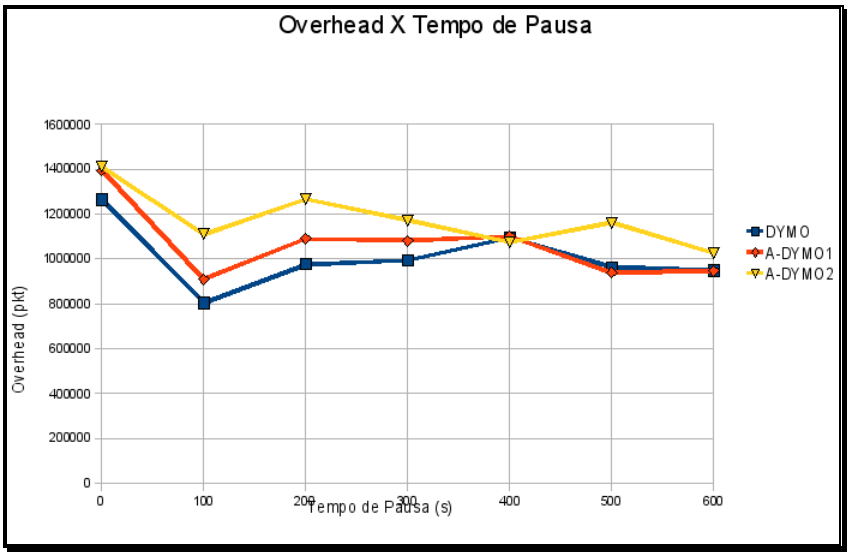


Figura 5.5-o: Overhead 20n x 5c x 10m/s

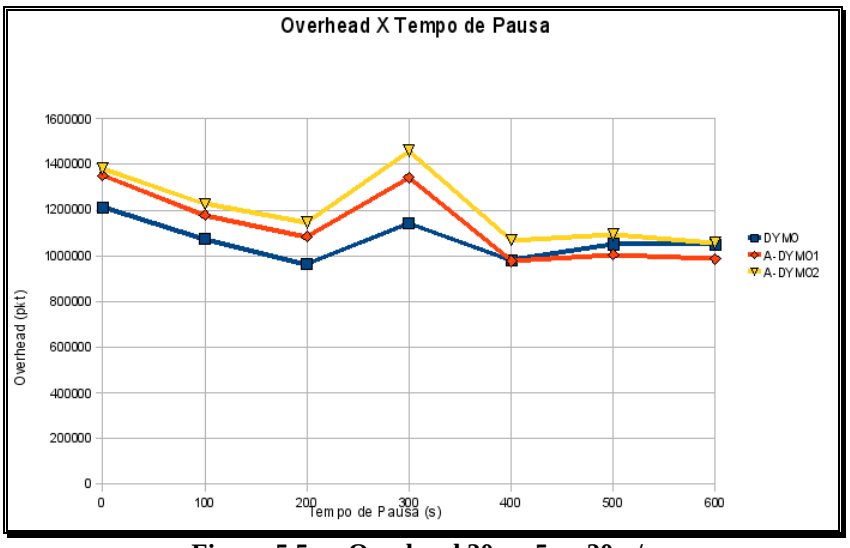


Figura 5.5-p: Overhead 20n x 5c x 20m/s

Como era de se esperar o protocolo A-DYMO2, por trabalhar com parâmetros máximos, tem as maiores taxas de overhead. Em relação ao protocolo DYMO o A-DYMO2 possui pouco mais de 14%, 10%, 29% e até 40% nas figuras: Figura 5.5-m, Figura 5.5-n, Figura 5.5-o e Figura 5.5-p.

O protocolo A-DYMO1 ficou com uma taxa de overhead em torno de 15% maior que o protocolo DYMO.

5.6 Conclusão

Os resultados obtidos com o protocolo A-DYMO1 mostram que ele é hábil em entregar os dados em tempo médio menor que o protocolo DYMO quanto ao A-DYMO2. Também fica claro que a taxa de entrega é superior na maioria dos cenários.

Nas métricas de perda e overhead já era esperado um pior desempenho do protocolo A-DYMO. A nova informação fica por conta do pior desempenho utilizando parâmetros máximos.

6 Conclusão e Trabalhos Futuros

6.1 Análise Final

No primeiro capítulo desta dissertação, mais especificamente na seção 1.1 os objetivos foram traçados:

"*O objetivo principal desta pesquisa foi a proposta e a implementação de um protocolo de roteamento para redes ad hoc chamado A-DYMO.* "

Os objetivos foram cumpridos.

Após um período de revisão bibliográfica e análise de alguns protocolos já existentes foi possível propor um protocolo para redes MANET que utilizasse a meta-heurística ACO, que foi chamado de A-DYMO. Em seguida o passo natural foi sua implementação.

A implementação foi bem sucedida, e o protocolo A-DYMO foi construído, onde no mesmo, um conjunto de parâmetros acessíveis através da arquitetura interpretada deu mais agilidade à fase de testes.

Depois da fase de testes, passou-se à simulação do protocolo A-DYMO e sua posterior avaliação tendo como base comparativa o protocolo DYMO. Deu-se, em seguida, sequência aos trabalhos de estudo e análise.

Os gráficos do capítulo anterior confirmaram os resultados esperados por nosso estudo. Assim podemos citar:

- O protocolo A-DYMO teve em geral, melhor taxa de entrega;

- O protocolo A-DYMO, na maioria dos cenários propostos, entregou os dados em um menor tempo.
- Um overhead extra era esperado no protocolo A-DYMO e seus efeitos também foram confirmados, tanto no que diz respeito a taxa de entrega quanto a quantidade nominal de fluxo de tráfego acima da criada pelo protocolo DYMO.
- O protocolo A-DYMO ajustado com parâmetros máximos obteve mais pontos negativos do que positivos, pelo menos nos cenários propostos.

Assim, o protocolo A-DYMO mostrou-se superior ao protocolo DYMO no que diz respeito a efetiva entrega de dados em um tempo inferior. Mostrou também, que é possível através de ajustes em seus parâmetros influenciar diretamente no seu desempenho.

6.2 Prosseguido com a Pesquisa

Vislumbrando trabalhos futuros, deve-se levar em consideração principalmente que o projeto e a implementação do protocolo A-DYMO tiveram como métrica de qualidade a distância, medida em número de saltos, devendo, portanto, o passo natural seguinte ser a inserção de uma métrica mais sofisticada.

Assim sugerimos como trabalhos futuros:

- Utilização de outras métricas, tais como, qualidade do link através de previsão nível de energia dos enlaces, tamanho do buffer, previsão de congestionamento, dentre outros.
- Adaptação do protocolo A-DYMO para trabalhar com função multiobjetivo.

6.3 Últimas Considerações

O problema de roteamento em redes *ad hoc* é largamente estudado, pois através dele decorre uma série de consequências para a rede. Logo uma estratégia de

roteamento bem proposta e implementada pode economizar recursos valiosos das redes sem fio. Assim, sempre que uma nova estratégia surge como opção para contribuir com esse problema, ela é rapidamente verificada. Da mesma forma como a meta-heurística ACO, outras técnicas baseadas na observação de elementos da natureza irão surgir apresentando-se como opção para a resolução de problemas computacionais. Desta forma, cabe a comunidade científica ficar atenta para não desperdiçar as oportunidades que certamente surgirão.

7 Referências Bibliográficas

[1] Perkins, C. E (Coord.). *Ad hoc Networking*. [S.l.]: Addison-Wesley, 2001.

[2] Liu, C.; Kaiser, J. *A Survey of Mobile Ad hoc network Routing Protocols*. Magdeburg: University of Magdeburg, 2005.

[3] Chakeres, I.; Perkins, C. E. *DYMO - Dynamic MANET On-demand Routing*. Internet draft. Draft-ietf-manet-dymo-05, 2006. Disponível em: http://tools.ietf.org/html/draft-ietf-manet-dymo-05. Acesso em: 10 dez. 2008.

[4] IETF Manet charter. Disponível em: http://www.ietf.org/html.charters/manet-charter.html. Acesso em: 15 dez. 2008.

[5] The Network Simulator – Ns2. Disponível em: http://www.isi.edu/nsnam/ns/. Acesso em 01 de Julho de 2009.

[6] Perkins, C. E.; Belding-Royer, E. *RFC 3561: AODV - Ad hoc On-Demand Distance Vector Routing*, 2003. Disponível em: http://tools.ietf.org/html/rfc3561. Acesso em: 10 dez. 2008.

[7] Johnson, D.; Hu, Y.; Maltz, D. *RFC 4728: DSR - The Dynamic Source Routing Protocol*, 2004. Disponível em: http://tools.ietf.org/html/rfc4728. Acesso em: 10 dez. 2008.

[8] Perkins, C. E.; Bhagwat, P. *DSDV - Highly dynamic Destination-Sequenced Distance-Vector routing for mobile computers*. London: ACM Computer, 1994.

[9] The SECAN-LAB. *Interoperability Laboratory for Security in Ad-Hoc Networks.* Luxembourg: University of Luxembourg, 2004-2006. Disponísvel em: http://wiki.uni.lu/secan-lab/SECAN-LAB.html Acesso em: 15 jan. 2009.

[10] Murthy, S.; Garcia-Luna-Aceves, J. J. *WRP - An Efficient Routing Protocol for Wireless Network*. Santa Cruz: University of California, 2006.

[11] Toh, C.-K. *ABR - Associativity-Based Routing For Ad-Hoc Mobile Networks*. United Kingdom: University of Cambridge Computer Laboratory, 1996.

[12] Zhou, H.; Singh, S. *CBM - Content based multicast in ad hoc networks.* Oregon: Oregon State University, 2000.

[13] Sivakumar, R.; Sinha, P.; Bharghavan, V. *CEDAR: Core-Extraction Distributed Ad hoc Routing algorithm.* Illinois: University of Illinois at Urbana-Champaign, 1999.

[14] Haas, Z. J.; Pearlman, M. R.; Samar, P. *ZRP - The Zone Routing Protocol for Ad hoc Networks.* Internet Draft. Draft-ietf-manet-zone-zrp-04.txt. Disponível em: http://tools.ietf.org/id/draft-ietf-manet-zone-zrp-04.txt. Acesso em: 26 jan. 2009.
[15] Basagni, S. et al. *DREAM - A Distance Routing Effect Algorithm for Mobility.* Dallas: The University of Texas at Dallas, 1998.
[16] Pei, G.; Gerla, M.; Chen T.-W. *FSR - Fisheye State Routing in Mobile Ad hoc Networks.* Los Angeles: University of California, Los Angeles, 2000.
[17] Nikaein, N.; Bonnet, C.; Nikaein, N. *HARP - Hybrid Ad hoc Routing Protocol.* France: Sophia Antipolis, Institut Eurécom, 2001.
[18] Ko, Y.-B.; Vaidya, N. H. *LAR - Location-Aided Routing in mobile ad hoc networks.* Texas: Texas A&M University, 2000.
[19] Dube, R. et al. *SSA - Signal Stability based Adaptive Routing for Ad-Hoc Mobile Networks.* Maryland: University of Maryland, 1996.
[20] Joa-Ng, M.; Lu, I.-T. *A Peer-to-Peer Zone-Based Two-Level Link State Routing for Mobile Ad hoc Networks.* [S.l], 1999.
[21] Ros, J. F.; Ruiz, M. P. DYMOUM. Disponível em: http://masimum.dif.um.es/?Software:DYMOUM. Acesso em: 07 jul. 2008.
[22] Blum, C.; Li, X. Swarm Intelligence in Optimization. In: Blum, C.; Merkle, D. (Ords.). Swarm Intelligence: Introduction and Applications. German: Springer, 2008. p. 43-86.
[23] Beekman, M.; Sword, G. A.; Simpson, S. J.; Biological Foundations os Swarm Intelligence. In: Blum, C.; Merkle, D. (Ords.). Swarm Intelligence: Introduction and Applications. German: Springer, 2008. p. 3-42.
[24] Dorigo, M.; Stutzle, T. Ant Colony Optimization. USA: The MIT Press, 2004. 321p.
[25] Dorigo, M.; Birattari, M.; Stutzle, T. Ant Colony Optimization: Artificial Ants as a Computational Intelligence Techniqu. IEEE Computational Intelligence Magazine. Belgium, Université Libre de Bruxelles, nov. 2006.
[26] Bonabeau, E.; Dorigo, M.; Theraulaz, G. Swarm Intelligence: From Natural to Artificial Systems. England: Oxford Univerity Press, 1999, 320p.
[27] Gunes, M.; Sorges, U.; Bouazizi, I. The Ant-Colony Based Routing Algorithm for MANETs. International Workshop on *Ad hoc* Networking (IWAHN 2002), Canada: British Columbia, Vancouver, August 18-21, 2002.
[28] Ducatelle, F.; Di Caro, G.; Gambardella, L.M. Ant Agents for Hybrid Multipath Routing in Mobile *Ad hoc* Networks. Switzerland: IDSIA, 2004.
[29] Marwaha, S.; Tham, C.K.; Srinivasan, D. Mobile Agents based Routing Protocol for Mobile *Ad hoc* Networks. Singapura: University of Singapore, [s.d.].
[30] Aissani, M.; Fenouche, M.; Sadour, H.; Mellouk, A. Ant-DSR: Cache Maintenance Based Routing Protocol for Mobile Ad-Hoc Networks. The Third Advanced Conference on Telecommunications(AICT'07), IEEE 2007.
[31] Yang, X.; Layuan, L.; Chuanhui, C. Application Research Based Ant Colony Optimization for MANET. China: Department of Computer Science, Wuhan University of Technology, 2006.

[32] Hussein, O.H.; Saadawi, T.N.; Lee, M.J. Ant Routing Algorithm for Mobile Ad-hoc networks (ARAMA). IEEE Journal on Selected Areas in Communications, New York, Vol. 23, N.12, dez 2005.
[33] Liu, Z.; Kwiatkowska M. Z.; Contantinou, C. A Swarm Intelligence Routing Algorithm for Manets. England: University of Birmingham, Birmingham, [s. d.].
[34] Liu, Z.; Kwiatkowska M. Z.; Contantinou, C. A Biologically Inspired QoS Routing Algorithm for Mobile *Ad hoc* Networks. England: University of Birmingham, Birmingham, [s. d.].
[35] Wang, J. et al. HOPNET: A hybrid Ant Colony Optimization Routing Algorithm for Mobile *Ad hoc* Networks. *Ad hoc* Networks. Canada: University Winnipeg, Winnipeg, 2008.
[36] Liu, M.; Sun, Y.; Liu, R.; Huang, X. An Improved Ant Colony QoS Routing Algorithm Applied to Mobile *Ad hoc* Networks. China: Central China Normal University, Wuhan, 2007.
[37] Braun, T.; Heissenbuttel. Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks.Switzerland: University of Berne, Neubruckstr, [s. d.].
[38] Osagie, E.; Thulasiraman P.; Thulasiram, R.K. PACONET: imProved Ant Colony Optimization routing algorithm for mobile *ad hoc* NETworks. 22nd Internacional Conference on Advanced Information Networking an Applications. IEEE, 2008.
[39] Baras, J.S.; Mehta, H. A Probabilistic Emergent Routing Algorithm for Mobile *Ad hoc* Networks. USA: University of Maryland, College park, Department of Elecrical and Computer Enginnering and the Intitute for Systems Research, [s. d.].
[40] Kamali, S.; Opatrny, J. POSANT: A Position Based Ant Colony Routing Algorithm for Mobile Ad-hoc Networks. Third International Conference on Wireless and Mobile Communications (ICWMC'07), IEEE, 2007.
[41] Dorigo, M.; Stutzle, T. An Experimental Study of the Simple Ant Colony Optimization Algorithm. Belgium: Univeristé Libre de Bruxelles, Bruxelles, [s. d.].
[42] Liu, Z.; Kwiatkowska, M.Z.; Constantinou, C. A Biologically Inspired Optimisation to AODV Routing Protocol. England: University of Birmingham, Birmingham, [s. d.].
[43] du Pessis, j. Colony Optimisation Distance Vector Routing in *Ad hoc* Networks. 2005. 167f. Dissertação (Mestrado em Ciência da Computação). Faculty of Engineering, Built-Enviroment and Information Technology, University of Pretoria, South Africa, nov. 2005.
[44] Gernhardt, G. Disponível em: CONFINT - http://search.cpan.org/~cgernhar/ConfInt-1.0.1/. Acesso em: 30/04/2009.