UNIVERSIDADE DO ALGARVE

Instituto Superior de Engenharia

Algoritmos *Ant Colony Optimization* para o Encaminhamento de Informação em Redes IP com Fios

por

João Magona Mapisse

Mestrado em Engenharia Eléctrica e Electrónica

Área de Tecnologias de Informação e Telecomunicações

Dissertação submetida em cumprimento do

requisito para a obtenção do grau de

Mestre em Engenharia Eléctrica e Electrónica.

Departamento de Engenharia Electrotécnica

Outubro de 2010

Universidade do Algarve

Instituto Superior de Engenharia

Mestrado em Engenharia Eléctrica e Electrónica

Área de Tecnologias de Informação e Telecomunicações

Esta Dissertação, intitulada:

Algoritmos *Ant Colony Optimization* para o Encaminhamento de Informação em Redes IP com Fios

foi escrita por João Magona Mapisse ,

e orientada pelo

Professor Doutor Pedro Jorge Sequeira Cardoso e pelo

Professor Doutor Jânio Miguel E. Ferreira Monteiro

tendo sido aprovada pelo

Departamento de Engenharia Electrotécnica

do Instituto Superior de Engenharia.

Faro, Outubro de 2010

Algoritmos *Ant Colony Optimization* para o Encaminhamento de Informação em Redes IP com Fios
por
João Magona Mapisse

Os algoritmos *swarm intelligence* tem contribuido significativamente para a resolução de vários problemas na área de optimização. Estes emulam o comportamento dos seres sociais na busca de alimentos. O primeiro algoritmo criado, destinava-se à resolução de problemas discretos de optimização e foi designado por *Ant System* [Dorigo *e*t al. , 1991]. Em 1997, Di Caro e Dorigo desenvolveram o primeiro algoritmo de encaminhamento mais bem sucedido, para para redes com fios, inspirado no comportamento das colónias de formigas na busca de alimentos, tendo sido designado por *AntNet*. Os bons resultados obtidos por este algoritmo impulsionou uma intensa investigação nessa área, sendo que alguns dos algoritmos criados a posteriori, podem ser encontrados na Tabela 1.1 desta dissertação.

Este trabalho faz parte de um projecto de pesquisa na área de optimização, direccionada para redes de dados com fios, que procura utilizar o comportamento dos insectos sociais na busca de alimentos, associando novos métodos e parâmetro com vista à produzir melhores resultados em relação ao *AntNet*.

Nesse sentido foram criados os algoritmos ϵ-*DANTENet* e *CR-DANTENet*, que associam o método usado no *AntNet* com a pesquisa em profundidade. Por outro lado, são apresentados três novos algoritmos, nomeadamente: *AntNetBw*, ϵ-*DANTENetBw* e *CR-DANTENetBw*, que introduzem um novo parâmetro das redes com fios. Trata-se da *utilização da largura de banda* ou simplesmente *largura de banda disponível*, que não é tida em consideração no algoritmo *AntNet* original e em vários que usam esta heurística.

Palavras Chave: *Swarm Intelligence*, *Ant Colony Optimization*, Largura de Banda Disponível, Encaminhamento, Formiga Artificial, Pesquisa em Profundidade.

Ant Colony Optimization Algorithms for Data Routing in Internet Protocol Wired Networks
by
João Magona Mapisse

The Swarm intelligence algorithms (algorithms that emulate the behavior of social insects in search of food) have been contributed to give solution for many optimization problems. Ant System was the first algorithm to be created and aimed to solve discrete problems [Dorigo *e*t al. , 1991]. In 1997, Di Caro and Dorigo developed the first success routing algorithm for wired networks, inspired by the behavior of ant colonies when it seek for food. This algorithm was called AntNet.

Based on its good experimental results, many researchers started to develop new algorithms (based in principles of AntNet). Some of these algorithms we can find in Table 1.1 of this dissertation.

This work is part of a research project in optimization for data wired networks, which aims use the behavior of social insects as algorithm to adapt with other methods and parameters in order to produce better results than AntNet.

In this case we created ϵ-DANTENet and CR-DANTENet algorithms with both ants behavior and deep search method, and AntNetBw, ϵ-DANTENetBw and CR-DANTENetBw which introduce in above algorithms, a new parameter of wired networks called *available bandwidth*, not taken in account in the original AntNet algorithm and several who use this heuristic.

Index Terms: Swarm Intelligence, Ant Colony Optimization, Available Bandwidth, Routing, Agent, Deep Search.

Dedicatória

À minha mãe Maria Angelina

ao meu pai Tiago Mapisse

Agradecimentos

Nunca deixo de agradecer a Deus por iluminar os meus caminhos e fortalecer-me para que possa enfrentar os desafios que vão surgindo minuto a minuto. Aos meus pais, que se envolveram em esforços para a minha educação e na dos meus irmãos, por sempre incutirem o espírito académico em cada um de nós, que me permitiu chegar a esta fase. *Sei que fui o menimo que mostrava a sua rebeldia através de pontos de vista e acções, mas tudo no sentido de mudar alguma coisa, também foi difícil para mim mas acabei conseguindo alguma coisa; e apesar de tudo isso nunca deixei de vos amar.*

Queria também agradecer a Rosalina, minha noiva, que esteve presente activamente nestes dois anos da minha formação, apoiando-me e não me deixando desistir do curso. Não deixo de realçar que vários acontecimentos ocorreram neste período, tendo sido um deles, a possibilidade de me tornar pai pela primeira vez.

Os meus agradecimentos estendem-se ao meu irmão Lucas e irmã Irene, que estando do outro lado do oceano, na pérola do Índico, sempre entraram em contacto comigo, transmitindo aquele conforto que é característico, tendo este sido de grande ajuda emocional dando-me mais motivação. Afinal nestes momentos o apoio familiar

é de extrema importância.

Um agradecimento especial vai para os meus orientadores, os Professores Doutores Pedro Jorge Sequeira Cardoso e Jânio Miguel E. Ferreira Monteiro, pela atenção, compreensão, apoio e ajuda que têem sido característico e principalmente por abdicarem dos seus preciosos tempos para poderem responder com prontidão a todas questões que eu ia colocando; contribuindo para o enriquecimento e sucesso deste trabalho, que teve o seu início com uma simples instalação e configuração do *ns-2* no Linux, passando para a criação de scripts para a simulação até a configuração dos métodos usados neste trabalho.

Ao Instituto Português de Apoio ao Desenvolvimento - IPAD, vai o meu "muito obrigado" pelo apoio concedido para a realização deste curso.

Ao meu colega de Cabo Verde, Emanuel, vai um "obrigado", pelo contacto permanente que tivemos um com o outro durante o período de quase 1 ano; afinal começámos juntos alguns dos módulos constituíntes deste trabalho e tivemos no início os mesmos orientadores.

Aos demais, que de forma directa ou indirecta, tenham contribuido para o enriquecimento deste trabalho, seja de forma positiva ou negativa, vai o meu "muito obrigado".

Conteúdo

Lista de Tabelas

Tabela

Lista de Figuras

Figura

Siglas

AntNet	*Ant Network*
AntNetBw	*Ant Network with Bandwidth*
BGP	*Border Gateway Protocol*
CBQ	*Class Based Queuing*
CBR	*Constant Bitrate*
CR-DANTENet	*Congestion Reaction - Depth ANT Explorer Network*
CR-DANTENetBw	*Congestion Reaction - Depth ANT Explorer Network with Bandwidth*
DRR	*Deficit Round Robin*
FIFO	*First In First Out*
FQ	*Fair Queuing*
FTP	*File Transfer Protocol* ou Protocolo de Transferência de Ficheiros
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*

LAN	*Local Area Network* ou Rede Local
LSA	*Link-State Advertisement*
MAC	*Media Access Control*
MAN	*Metropolitan Area Network* ou Rede Metropolitana
NAM	*Network Animator*
ns-2	*Network Simulator - 2*
OSI	*Open Systems Interconnection*
OTcl	*Object- Oriented Tool Command Language*
PC	*Personal Computer* ou Computador Pessoal
QoS	*Quality of Service* ou Qualidade de Serviço
RED	*Random Early Detection*
RIP	*Routing Information Protocol*
SFQ	*Statistical Fair Queuing*
SI	*Swarm Intelligence*
TCP	*Transmission Control Protocol*
ToS	*Type of Service*
UDP	*User Datagram Protocol*
VBR	*Variable Bitrate*
WAN	*Wide Area Network* ou Rede de longa distância
ϵ-*DANTENet*	ϵ-*Depth ANT Explorer Network*
ϵ-*DANTENetBw*	ϵ-*Depth ANT Explorer Network with Bandwidth*

CAPÍTULO

1

Introdução

O sistema de rede de computadores que transmite a informação em pacotes[1] iniciou o seu desenvolvimento por Leonard Kleinrock, Paul Baran e Donald Davies no início da década de 1960. Com o surgimento dos primeiros PC's, durante a década de 1970, houve uma maior difusão destas redes (LAN's[2] , MAN's[3] , WAN's[4] e Internet, conhecida como rede mundial ou rede entre WAN's) [Tanenbaum, 2003]. Actualmente estas redes encontram-se integradas com as redes de telecomunicações.

A interligação entre estas redes é feita através de *routers* que tem como função efectuar o encaminhamento de pacotes entre estas. Esse encaminhamento é feito recorrendo à tabela de encaminhamento de cada *router*. Cada uma dessas tabelas

[1] Pacote (ou datagrama) é uma estrutura unitária de transmissão de informação ou uma sequência de dados transmitida por uma rede ou linha de comunicação que utilize a comutação de pacotes.

[2] LAN - é uma rede de computadores que abrange um espaço físico pequeno, como a casa, o escritório ou um pequeno grupo de edifícios de uma escola ou de um aeroporto.

[3] MAN - é uma rede de computadores com o perímetro de uma cidade. Usualmente esta rede interliga várias LAN's

[4] WAN - conhecida como rede geograficamente distribuída, é uma rede de computadores que abrange uma grande área geográfica, com frequência um país ou continente.

possui informações de como encaminhar para um dado destino os pacotes que chegam aos seus interfaces[5] .

Em geral o encaminhamento pode ser definido de forma estática ou dinâmica. Num sistema de encaminhamento estático, a informação de encaminhamento presente nas tabelas de encaminhamento de cada *router* é definida pelos administradores dos respectivos sistemas e mantidas sem quaisquer alterações; enquanto que, num sistema de encaminhamento dinâmico são usados protocolos de encaminhamento que permitem alterar automaticamente a informação de encaminhamento presente nas referidas tabelas.

Os protocolos de encaminhamento recorrem aos algoritmos de encaminhamento para encontrar o caminho com o menor custo para um dado destino. Esse custo pode ser determinado por diversos critérios dentre os quais a distância, o tempo, etc. Tais algoritmos usam processos matemáticos para tomar as suas decisões de encaminhamento e serão o objecto de estudo nesta dissertação.

Com o aumento do tráfego nas redes de dados, devido a transmissão de aplicações multimédia e de dados cada vez mais exigentes, surge a necessidade de criar algoritmos de encaminhamento que respondam não apenas à menor distância (como é o caso dos algoritmos de encaminhamento *Distance-Vector* e *Link-State* - ver Capítulo 3), mas que, de uma forma dinâmica, monitorizem e procurem o melhor caminho entre as várias possibilidades de encaminhamento existentes. Nessa sequência, surgem os algoritmos *Ant Colony Optimization* (ACO) [Dorigo & Stutzle, 2004] que são meta heurísticas[6] que usam colónias de formigas artificiais na busca da melhor solução para problemas discretos de optimização. Estes algoritmos são inspirados no

[5] Interface é definido genericamente como conjunto de meios físicos ou lógicos criados com objectivo de fazer a adaptação entre dois ou mais sistemas. No caso das redes, as interfaces permitem a interligação entre as redes.

[6] Meta heurísticas são métodos heurísticos que procuram soluções óptimas num conjunto de soluções em problemas de optimização [Glover, 1986].

comportamento das formigas reais na busca de alimentos. A Tabela 1.1 apresenta alguns dos algoritmos ACO usados para solucionar as questões de encaminhamento em redes de dados com fios, subdivididos em duas categorias:

- *best effort*[7] e
- *QoS*[8] .

Tabela 1.1: Algoritmos ACO para redes com fios existentes nas categorias *best effort e QoS*.

Problema: Best Effort	
Nome do algoritmo	**Referência**
AntNet, AntNet-FA	Di Caro & Dorigo [1997b]; Di Caro & Dorigo [1997a] Di Caro & Dorigo [1998b]
ABC Uniform ants	Subramanian *et* al. [1997]
CAF	Heusse *et* al. [1998]; Heusse *et* al. [1999]
ABC-backward	van der Put [1998]
DCY-AntNet, NFB-Ants	Oida & Kataoka [1999]
AntNet NetMngmt	Gallego-Schmid [1999]
BntNetL	Doi & Yamamura [2000]; Doi & Yamamura [2002]
Improved AntNet	Baran & Sosa [2000]
AntNet Single-path	Jain [2002]
AntNet security	Zhong & Evans [2002]
Adaptive-SDR	Kassabalidis *et* al. [2002]
Problema: QoS	
ABC	Schonderwoerd *et* al. [1996]; Hayzelden & Bigham [1999]
ASGA	White *et* al. [1998b]; White *et* al. [1998a]
AntNet-FS	Di Caro & Dorigo [1998a]
ABC Smart ants	Bonabeau *et* al. [1998]
ARS	Oida & Sekido [1999]; Oida & Sekido [2000]
AntNet+SELA	Caro & Vasilakos [2000]
Multi-swarm	Michalareas & Sacks [2001a]; Michalareas & Sacks [2001b]
Ant-based routing	Sandalidis *et* al. [2001]; Sandalidis *et* al. [2004]
Ant-QoS	Subing & Zemin [2001]
QColony	Tadrus & Bai [2003.]

[7] *Best effort* é um conceito adoptado em redes de dados onde todos os pacotes são tratados de igual forma, i.e., partilham a mesma largura de banda disponível com os mesmos critérios [Park, 2005].

[8] *QoS* - é um conceito adoptado nas redes de dados de forma a garantir determinado nível de qualidade de um serviço ou aplicação (voz, video ou dados) no receptor [Park, 2005].

O primeiro método mais bem sucedido apresentado como um algoritmo ACO para o encaminhamento de pacotes em redes com fios foi o *AntNet* [Di Caro & Dorigo, 1997b]. Ao nível experimental (ou simulação) em que foi testado, este algoritmo foi um dos poucos a atingir um alto nível de desempenho [Lawrence & Giles, 1998]. Em 1998 a versão do *AntNet* foi modificada, alterando a forma como eram feitas as actualizações nas tabelas de encaminhamento e no tratamento dos custos[9] .

No entanto este algoritmo não leva em consideração a utilização da largura de banda entre as ligações, não evitando situações eminentes de congestionamento entre as mesmas. Este assunto será alvo de análise no decorrer desta dissertação.

1.1 Motivação do trabalho

O aumento do tráfego através da Internet, levou muitos investigadores a procurarem e criarem novas soluções que pudessem garantir melhores condições de transmissão de pacotes através da Internet. Nesse sentido, para além dos algoritmos clássicos como o *Distance-Vector* e o *Link-State*, que encaminham pacotes com base nas tabelas de encaminhamento construídas a partir de um critério baseado em custos mínimos (sendo que as tabelas de encaminhamento apenas são actualizadas quando acorre uma alteração na topologia da rede), surge o *AntNet* [Di Caro & Dorigo, 1997b]. O *AntNet* possibilita o encaminhamento usando tabelas de encaminhamento construídas e actualizadas dinamicamente, sempre que se verifica uma alteração nas condições da rede, como sejam: a topologia, o tráfego ou o custo. A solução *AntNet* e outras que funcionam baseadas em princípios semelhantes, deram origem ao conceito de encaminhamento inteligente [Farooq, 2006].

Basicamente os algoritmos ACO possuem dois tipos de processos, que são ca-

[9] A abordagem acerca dos custos será feita mais adiante, durante a descrição do algoritmo *AntNet*.

racterísticos nos algoritmos utilizados para encaminhamento em redes sem fios, nomeadamente: o reactivo e o proactivo. O processo reactivo é usado para encontrar informações de encaminhamento, até ao momento indisponíveis, de um dado nó de origem para um nó destino e o processo proactivo é usado para melhorar as informações de encaminhamento encontradas no processo reactivo [Di Caro *e*t al. , 2004]; [Rajagopalan & Shen, 2006].

Um outro aspecto importante relaciona-se com os parâmetros da rede. Os algoritmos clássicos como o *Distance-Vector* e o *Link-State* constroem as suas tabelas com base nos custos existentes entre os nós adjacentes, enquanto que os algoritmos ACO, como é o caso do *AntNet*, constroem as suas tabelas de encaminhamento com base na quantidade de pacotes presentes nas filas de espera de cada nó e na qualidade do caminho. No algoritmo *AntNet* a qualidade de um caminho é designado por intensidade de feromonas (que será estudada posteriormente). Essa qualidade expressa a frequência com que o mesmo caminho é escolhido entre um nó de origem e destino.

Este trabalho introduz nos algoritmos ACO o conceito de pesquisa em profundidade através de dois algoritmos. Este conceito foi introduzido para problemas discretos, pela primeira vez por Cardoso *e*t al. [2010]. A ideia por detrás deste conceito é a de explorar as soluções encontradas pelo método *AntNet*, na busca de alternativas melhores sem que seja necessário reconstruir as soluções por completo.

Um outro conceito introduzido neste trabalho é o da utilização da largura de banda. Este conceito surgiu da necessidade de encaminhar os pacotes para ligações que possuem maior disponibilidade de largura de banda. Esta utilização permite reduzir consideravelmente a possibilidade de surgimento de congestionamento em determinadas ligações e consequente redução do número de pacotes perdidos, em redes com elevada complexidade. Estes métodos foram testados através da criação de três

algoritmos de encaminhamento ACO, que apresentaremos ao longo deste documento.

1.2 Objectivos do trabalho

Em termos matemáticos o conceito de pesquisa em profundidade, ao contrário dos processos reactivo e proactivos, visa a construção de soluções óptimas com base em soluções não consideradas óptimas. Esta ideia foi inicialmente apresentada, para a resolução de problemas discretos, no trabalho de Cardoso *e*t al. [2010], no qual soluções híbridas, que utilizam a pesquisa em profundidade com algoritmos ACO, apresentam bons resultados em redes com elevada complexidade.

O conceito de utilização da largura de banda que será introduzido nos algoritmos de encaminhamento ACO, poderá reduzir os níveis de congestionamento, evitando com isso grandes perdas de pacotes na rede.

Assim, formulamos as seguintes hipóteses para cada caso:

Hipótese para o problema da pesquisa em profundidade: Algoritmos de pesquisa em profundidade escaláveis deverão melhorar as médias de pacotes perdidos na rede em relação ao algoritmo *AntNet*.

Hipótese para o problema da integração da largura de banda: Algoritmos que usam a largura de banda escaláveis deverão melhorar as médias de pacotes perdidos na rede em relação ao algoritmo *AntNet*.

Ao logo da restante dissertação vamos tentar responder às hipóteses aqui colocadas.

1.3 Organização da dissertação

Este trabalho tem como objectivo apresentar novas soluções de encaminhamento para redes com fios, através da introdução do parâmetro de utilização da largura de banda bem como da criação de algoritmos híbridos que comportem o método *AntNet* e o comportamento exploratório. Assim, no Capítulo 2 definiremos e caracterizaremos os aspectos básicos que estão por detrás do encaminhamento, começando pelo conceito de rede, e dos elementos principais da mesma. Serão estudados os modelos de referências OSI e TCP/IP definidos para a comunicação entre elementos de uma mesma rede. Os aspectos principais relacionados com cada camada desses modelos serão mencionados, com o objectivo de conhecer o papel da c amada responsável pelo encaminhamento. Por fim mencionaremos os elementos constituíntes de uma tabela de encaminhamento.

No Capítulo 3 estudaremos os algoritmos clássicos como são os casos do *Distance-Vector* e do *Link-State* bem como o algoritmo *AntNet*. Descreveremos ainda a forma de funcionamento de cada um desses algoritmos, incidindo na forma como é efectuada a actualização das tabelas de encaminhamento; estudaremos o comportamento das colónias de formigas na busca de alimentos, para posterior percepção do mecanismo utilizado pelos algoritmos ACO, como é o caso do *AntNet*. Em suma, este capítulo tem como objectivo dar a conhecer os algoritmos de encaminhamento que estiveram na origem de vários protocolos de encaminhamento, para posteriormente fazer uma comparação destes com os algoritmos de encaminhamento por nós propostos nesta dissertação.

No Capítulo 4 apresentaremos o algoritmo de encaminhamento ϵ-*DANTENet* para redes com fios, que é baseado no algoritmo ϵ-*DANTE* criado para resolver problemas discretos de optimização. Neste capítulo será apresentado o princípio de fun-

cionamento deste algoritmo com especial incidência na forma como são criadas as formigas *DANTE's* e como é feita a actualização das tabelas de encaminhamento em cada nó. Apresentaremos também o algoritmo de encaminhamento *CR-DANTENet*, que surge na sequência de um melhoramento feito no algoritmo ϵ-*DANTENet*, enfatizando as principais características que diferenciam este do anterior.

No Capítulo 5 introduziremos um novo parâmetro aos algoritmos ACO, que tem como objectivo criar um equilíbrio entre as filas de espera juntamente com a intensidade de feromonas e o factor de largura de banda disponível entre as diversas ligações. Assim, começaremos por apresentar o pressuposto para a introdução deste parâmetro. Apresentaremos também as novas versões dos algoritmos de encaminhamento, nomeadamente o, *AntNetBw*, ϵ-*DANTENetBw* e *CR-DANTENetBw*, que recorrem à largura de banda disponível entre as ligações para efectuarem o encaminhamento das formigas nos nós intermédios, possibilitando a criação de novos e melhores caminhos para o encaminhamento de pacotes nas redes com fios.

No Capítulo 6 apresentaremos, para um conjunto mais alargado de redes, os resultados das simulações efectuadas com os algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw* e *CR-DANTENetBw* sendo feita a comparação entre estes. Para tal estudaremos a influência de um conjunto de parâmetros nos pacotes perdidos na rede. Neste capítulo feremos o estudo estatístico aplicando o teste t de modo a concluir acerca dos algoritmos apresentados, para as condições de rede consideradas.

No Capítulo 7 apresentaremos as considerações finais deste trabalho.

CAPÍTULO

2

Arquitectura protocolar das redes de comunicação

Com o presente capítulo pretendemos introduzir os conceitos básicos das redes de comunicação, através da caracterização de uma rede, descrição dos elementos constituintes, definição dos modelos que estão na base das comunicações entre os vários elementos de uma mesma plataforma de rede, bem como o processo que permite tal comunicação.

2.1 Visão geral

As redes de comunicação têm ocupado um papel cada vez mais importante no desenvolvimento da sociedade moderna. São utilizadas em larga escala nos mais variados organismos (empresarias e particulares), para a troca de informação. Um exemplo de uma entidade que se suporta em tais redes é a Sociedade Interbancária de Serviços (SIBS), que possui um sistema que interliga diversas instituições bancárias. Esse sistema permite efectuar operações bancárias tais como depósitos em contas, transferências, carregamentos de telefones, pagamentos de bens e serviços entre outras.

Inicialmente visto como um meio para interligar computadores, o seu domínio de actuação tem vindo a ser alargado às redes de telecomunicações, com a integração em diversas redes de acesso, permitindo desta forma uma nova gama de serviços, tais como voz e televisão sobre IP, para além do já habitual acesso à Internet.

Nesse sentido, e uma vez que este trabalho versa sobre algoritmos que são usados por protocolos de encaminhamento para transportar a informação para os diversos elementos de uma rede, torna-se necessário começar por definir os conceitos que estão por trás desses métodos. Por essa razão surge este capítulo que define o conceito de rede de comunicação e todos os elementos que a constituem, tais como, nós e ligações entre os mesmos. Apresentaremos e explicaremos as funções de todas as camadas do modelo TCP/IP, derivado do modelo OSI, que está na base das comunicações entre os diversos nós de uma rede.

Posteriormente definiremos e explicaremos o mecanismo de encaminhamento, recorrendo ao exemplo de uma rede. Para finalizar introduziremos o conceito de tabela de encaminhamento, o que nos vai permitir conhecer a sua estrutura e a dos elementos que a constituem.

2.2 Rede de comunicação

Para definirmos uma rede vamos recorrer ao conceito matemático de grafo. Assim sendo, um grafo é definido como uma estrutura representada por $G(V, A)$, onde o conjunto V representa os vértices (ou nós) e o conjunto A representa as arestas (ou ligações) desta estrutura. A Figura 2.1 apresenta um exemplo deste tipo de estrutura, onde os pontos *1, 2, 3, 4 e 5* representam os vértices e os segmentos de rectas *1-2, 1-3, 1-4, 2-3, 2-4, 3-5 e 4-5* representam as arestas.

Numa rede de dados, os nós e as ligações, representam os elementos da rede. O nó é uma unidade de processamento, transmissão e recepção. A ligação representa um sistema de transmissão caracterizado por possuir determinados parâmetros tais como a largura de banda, atrasos e tipo de ligação[1] .

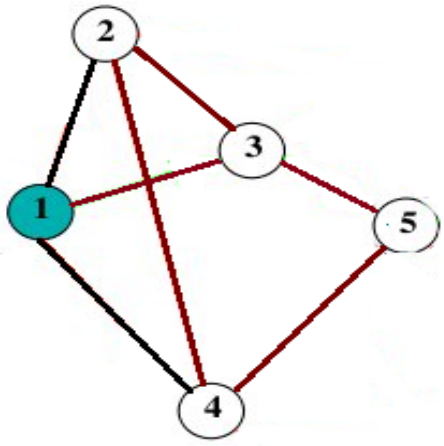


Figura 2.1: Exemplo de uma estrutura de grafo com 5 vértices e 7 arestas.

As redes de dados são criadas com o objectivo de trocar informações entre os

[1] O tipo de ligação refere-se ao sentido de transmissão da informação num dado canal.Existem ligações *simplex* onde a informação é transmitida no canal apenas num sentido, e as ligações *duplex* onde a informação pode ser transmitida nos dois sentidos. Ainda dentro das ligações *duplex* encontramos as classificadas como *half-duplex*, onde a transmissão da informação não é simultânea, e as *full-duplex* onde a transmissão da informação pode ser feita simultaneamente.

nós destas. Para que se efective essa troca de informação é necessário que haja um emissor, um canal e um receptor. Deste modo, define-se o nó de origem como emissor (o nó que envia dados); o nó de destino como receptor (o nó para onde os dados são enviados); a ligação entre os nós como canal de comunicação. Em muitos casos, a ligação entre os nós de origem e de destino é feita por intermédio de outros nós da rede. Assim, define-se o nó intermédio como sendo aquele que não sendo o nó de origem ou de destino, é responsável por determinar o caminho que os dados devem seguir para alcançar um dado destino, através do (re)encaminhamento da informação que lhes chega [Di Caro, 2004].

Quando um nó pretende enviar uma dada informação, este cria pacotes e envia pela rede para um determinado destino.

A sequência dos pacotes que apresentam determinada relação (por exemplo, devido ao facto de serem gerados pela mesma aplicação) é designada por fluxo. Os pacotes podem pertencer ao mesmo fluxo se forem classificados de acordo com alguns dos seguintes parâmetros: endereço de origem, endereço de destino, valor do *Type of Service* (ToS), o tipo de autenticação, o número do pacote e os parâmetros de segurança. Os fluxos podem ter como destino apenas um nó, um conjunto restrito de nós ou todos os nós da rede, sendo que desta forma teremos os fluxos *unicast*, *multicast* ou *broadcast*, respectivamente [Gai, 1998].

2.2.1 Nós de uma rede

Em geral, uma rede de comunicação é composta de nós e ligações. Os nós podem representar terminais como computadores, ou elementos intermédios tais como *bridges*[2] ou *routers*.

Numa rede Internet Protocol (IP) um nó é tipicamente designado por *router*,

[2] Ver a definição no 2º paragráfo da secção 2.4.

enquanto que numa rede telefónica é chamado de central ou comutador telefónico. Os *routers* encaminham pacotes entre distintas redes com base em endereços IP [Peterson & Davie, 2003] decidindo em qual das duas interfaces deve encaminhar um determinado pacote.

Actualmente os *routers* e *bridges* são construídos como dispositivos que executam funções específicas de encaminhamento e comutação (*switching*) respectivamente, de forma rápida em comparação com um computador no seu lugar.

Alguns dispositivos desempenham as funções de *routers* e de *bridges* de forma simultânea, trabalhando à nível de rede e de ligação de dados[3] .

2.3 O Modelo OSI e TCP/IP

O modelo OSI (Figura 2.2) estabelece a forma comum segundo o qual os computadores devem ser conectados e foi definido pela primeira vez pela organização ISO[4] [Kaufmann, 2007]. Este é constituído por 7 camadas, onde um ou mais protocolos executam as funcionalidades definidas por cada uma delas. O modelo OSI é uma referência para a estrutura de protocolos que definem as regras de como os processos devem efectuar a comunicação.

Descrevendo cada camada começando pela camada mais baixa, encontramos a [Peterson & Davie, 2003]:

- camada física que é responsável pela transmissão da sequência de bits (impulsos eléctricos ou ópticos) de informação através do canal;

- camada de ligação que recebe o fluxo de bits agrupados em tramas[5] ;

[3] Os conceitos de nível de rede e de ligação de dados encontram-se definidos na secção 2.3.

[4] *International Organization for Standardization.*

[5] Pacotes gerados pelos protocolos da camada de rede podem ser encapsulados dentro de um trama para serem transportados pela rede.

Figura 2.2: Modelo de referência OSI.

- camada de rede que é responsável pelo encaminhamento (ver secção 2.4);
- camada de transporte que é responsável pela comunicação entre os processos, ou seja, recebe os pacotes da camada de rede e agrupa-os formando os dados para o envio à camada de sessão. De modo semelhante quando a camada de transporte recebe os dados da camada de sessão fragmenta-os em pacotes enviando para a camada de rede;

- camada de sessão que oferece mecanismos para controlo e sincronização do diálogo entre as entidades de aplicação comunicantes.

- camada de apresentação que é utilizada para converter o formato dos dados num formato adequado para a aplicação, fornecendo meios para o estabelecimento e utilização de sintaxes abstractas que possibilitam essa troca de informação;

- camada de aplicação que providencia o acesso ao ambiente OSI para os utilizadores, fornecendo mecanismos de comunicação orientados às aplicações. Alguns exemplos de aplicações que actuam neste nível são o HTTP e o FTP.

No modelo TCP/IP (Figura 2.3) as três camadas superiores são agrupadas em uma única camada. Este modelo deriva do modelo OSI e é constituído apenas por quatro camadas, nomeadamente a:

- camada de acesso à rede que representa a interface da rede. Nesta camada encontramos diversos protocolos de rede. Na prática, estes protocolos são implementadas por uma combinação de hardware (por exemplo, um adaptador de rede) e software (por exemplo, um *driver* de dispositivo de rede) [Tanenbaum, 2003];

- camada de Internet deste modelo corresponde à terceira camada o modelo OSI (camada de rede). Neste nível é utilizado o protocolo IP que suporta a interligação entre as várias tecnologias de rede. Nesta camada também é efectuado o endereçamento e o encaminhamento;

- camada de transporte contém dois protocolos principais, nomeadamente o TCP e o UDP. Na Internet, o TCP e UDP são chamados protocolos extremo-

a-extremo, embora seja igualmente correcto referir-se a eles como protocolos de transporte [Peterson & Davie, 2003];

- camada de aplicação deste modelo resulta da integração das três últimas camadas do modelo OSI, conforme referido anteriormente, nomeadamente a de aplicação, a de apresentação e a de sessão.

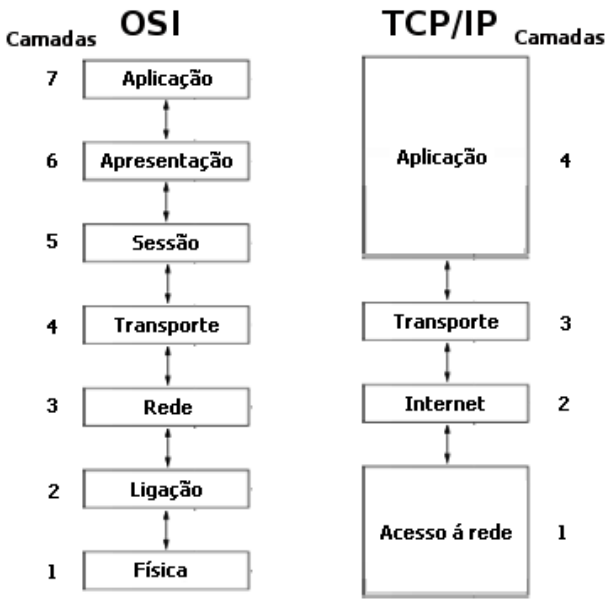


Figura 2.3: Modelo de referência OSI vs Modelo TCP/IP.

Nos modelos referenciados (Figuras 2.2 e 2.3), os *routers* trabalham na camada de rede e são responsáveis pelo encaminhamento de pacotes entre as redes ligadas à cada uma das suas interfaces. Isto é possível graças aos protocolos que os próprios *routers* utilizam em conjunto com os métodos ou algoritmos de encaminhamento, destinados à construção e actualização das tabelas de encaminhamento (Secção 2.5).

2.4 Encaminhamento

O encaminhamento é um processo através do qual um determinado nó intermédio, que executa as funções de um *router* ou *bridge*, recebe determinado pacote e envia para um outro nó, que pode ser o nó de destino ou um nó que tenha informações de como reencaminhar esse pacote para o nó ou nós de destino.

A *bridge* é um equipamento de rede que trabalha na camada de ligação de dados e efectua o encaminhamento baseando-se no endereço físico (endereço MAC). O *router*, ao contrário da *bridge*, trabalha ao nível de rede e faz o encaminhamento com base nos endereços do nível de rede [Kaufmann, 2007]. O nosso trabalho será baseado no encaminhamento ao nível de rede, por essa razão limitar-nos-emos à esta camada.

As redes distinguem-se umas das outras pelos seus endereços de rede[6] . A topologia da Figura 2.4 mostra um exemplo de duas LAN's e três MAN's que por sua vez interligam as LAN's. Os *routers* desta figura possuem três interfaces, sendo que cada uma delas identifica uma rede, através do seu endereço de rede.

Consideremos que foi definida uma máscara para toda a rede, em que os três primeiros bytes dos endereços desta rede correspondem ao segmento de rede e o último byte corresponde ao *host*. Deste modo, podemos afirmar que o *router* Maputo está ligado aos segmentos de rede *200.20.1.0, 200.20.2.0 e 192.168.4.0* através dos *hosts 3, 18 e 254* do mesmo, respectivamente. O *router* Beira está ligado aos segmentos de rede *200.20.2.0, 200.20.3.0 e 192.168.6.0* através dos *hosts 22, 27 e 61* do mesmo, respectivamente. Finalmente o *router* Nampula está ligado aos segmentos de

[6] Os endereços de rede do protocolo IP (versão 4) é composto por *4 bytes* identificando o segmento de rede (*Network ID*) e o terminal (*host*). A identificação do segmento da rede e do *host* é feito recorrendo a uma outra combinação de *4 bytes* denominada por máscara da rede. Na máscara de rede os bits a *1* identificam *Network ID* e o bits a *0* identificam o *host ID*. *Host ID's* a *0's* identificam o segmento e a *1's* identificam todos os terminais do segmento.

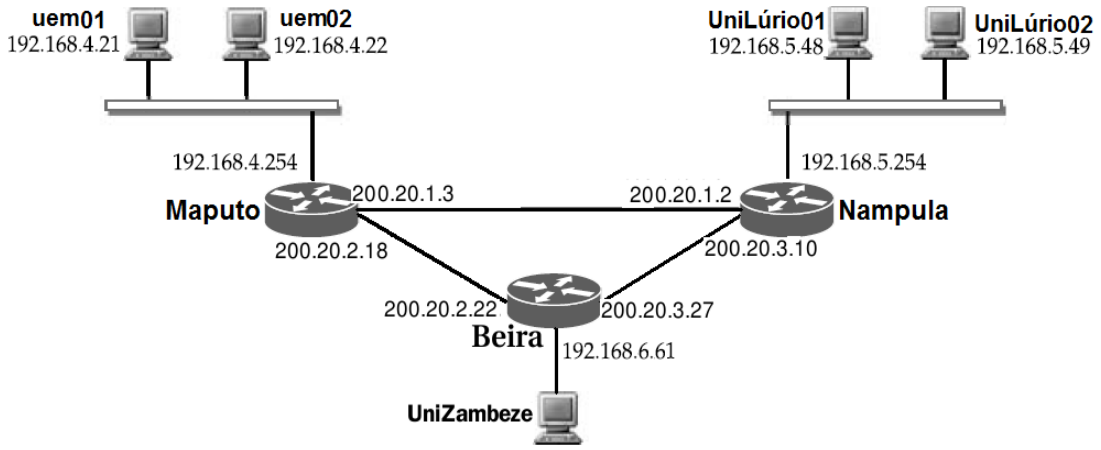


Figura 2.4: Exemplo de uma topologia de redes de dados, onde podemos ver três *routers* cada um fazendo a separação entre entre a LAN e a MAN.

rede *200.20.1.0, 200.20.3.0 e 192.168.5.0* através dos *hosts 2, 10 e 254* do mesmo, respectivamente.

Ainda na mesma rede, no segmento da rede *192.168.4.0* (à esquerda) temos a ligação entre o *router* Maputo, e o barramento que liga à dois computadores, nomeadamente a *uem01* e *uem02* com *hosts 21 e 22*, respectivamente. No segmento de rede *192.168.5.0* (à direita) temos a ligação entre o *router* Nampula, e o barramento que liga à dois computadores, nomeadamente a *UniLúrio01* e *UniLúrio02* com *hosts 48 e 49*, respectivamente. No segmento de rede *192.168.6.0* temos a ligação entre o *router* Beira e o computador *UniZambeze* com *host 61.*

Entretanto, o envio de pacotes entre *hosts* é feito mediante os endereços de rede, ou seja, caso queiramos enviar um pacote do computador *uem01* para *UniLúrio02* na rede da Figura 2.4, este será enviado, em primeiro lugar, para o endereço *192.168.4.254*, que corresponde a um dos interfaces do *router* Maputo que esta directamente ligado ao seu barramento. Por sua vez o *router* Maputo tem duas possibili-

dades de reencaminhar o pacote, quer seja pelo *router* Beira ou pelo *router* Nampula. Vamos supor que o *router* Maputo possui a informação de encaminhamento indicando que a rede que corresponde ao computador *UniLúrio02* é alcançavel através do *router* Nampula. Nesse sentido o *router* Maputo reencaminhará o pacote para o endereço *200.20.1.2*, através da sua interface *200.20.1.3*. O endereço *200.20.1.2* corresponde a uma das interfaces do *router* Nampula, que recebe este pacote e reencaminha-o para o endereço *192.168.5.254* que corresponde ao barramento ligado à sua interface. Neste barramento o pacote é entregue ao computador *UniLúrio02*.

Num outro cenário, em que pretendamos enviar um pacote do computador *uem01* para *UniZambeze*, este será enviado, em primeiro lugar, para o endereço *192.168.4.254*, que corresponde a um dos interfaces do *router* Maputo que está directamente ligado ao seu barramento. Supondo que o *router* Maputo possui a informação de encaminhamento indicando que a rede que corresponde ao computador *UniZambeze* é alcançavel através *router* Beira, o *router* Maputo reencaminhará o pacote para o endereço *200.20.2.22*, através da sua interface *200.20.2.18*. O endereço *200.20.2.22* corresponde a um dos interfaces do *router* Beira, que recebe este pacote e reencaminha através da sua interface *192.168.6.61* para o computador *UniZambeze*.

2.5 Tabelas de encaminhamento

Conforme descrito na Secção 2.4, a função do encaminhamento é a de enviar os pacotes de um dado nó (intermédio) para um nó ou nós de destino ou que possuam informações de como o nó de destino pode ser alcançado, através de uma das suas interfaces. A escolha da interface por onde um pacote deve seguir é feita procurando na tabela de encaminhamento do respectivo nó, o endereço de rede correspondente

ao destino pretendido. Estas tabelas são construídas pelos algoritmos de encaminhamento baseando-se nas mensagens trocadas entre os nós adjacentes.

Cada entrada na tabela de encaminhamento possui um *next hop* para um determinado destino na rede. Esse *next hop* identifica o interface de rede do *router* que deve ser usada para alcançar o destino pretendido. Dependendo do tipo de implementação, as entradas nas tabelas de encaminhamento podem ser feitas também com base no endereço de MAC. Muitas implementações tendem a manter apenas os endereços de rede devido ao facto das tabelas de encaminhamento serem optimizadas com base nestes. A tabela de encaminhamento do *router* Maputo (da Figura 2.4), apresentada na Tabela 2.1, ilustra um exemplo da composição das tabelas de encaminhamento. Nesta temos na primeira coluna a rede de destino, na segunda coluna a máscara da rede, na terceira coluna temos a interface por onde deve seguir o pacote para alcançar a rede de destino correspondente e na quarta coluna temos o *next hop*.

Tabela 2.1: Exemplo de uma tabela de encaminhamento do *router* Maputo.

***Router* Maputo**			
Rede de destino	Máscara	Interface	*Next hop*
192.168.4.0	255.255.255.0	192.168.4.254	-
192.168.5.0	255.255.255.0	200.20.1.3	200.20.1.2
192.168.6.0	255.255.255.0	200.20.2.18	200.20.2.22

Assim, pacotes que chegam ao *router* Maputo e tenham como destino o segmento de rede *192.168.4.0*, são enviados pela interface *192.168.4.254* que corresponde ao segmento de rede de destino; caso os pacotes tenham como destino o segmento de rede *192.168.5.0*, estes são reencaminhados através da interface *200.20.1.3* para o endereço *200.20.1.2*; por fim os pacotes que se destinam à rede *192.168.6.0* são reencaminhados através da interface *200.20.2.18* para o endereço *200.20.2.22*.

2.6 Sumário

Actualmente assiste-se a uma crescente procura pelas redes de comunicação e serviços. O seu estudo ao nível da investigação e desenvolvimento requer o conhecimento dos conceitos que estão por detrás dessas redes. Neste capítulo fizemos uma breve introdução aos conceitos básicos das redes de comunicação definindo os seus elementos principais, tais como nós e ligações.

Numa rede, os intervenientes na comunicação são geralmente os nós, que trocam mensagens entre si e fazem o encaminhamento da informação. A forma como essa comunicação se processa, é definida pelos protocolos do modelo OSI. Cada uma das camadas desse modelo é responsável pela execução de um conjunto de tarefas específicas. Por exemplo, a camada de rede deste modelo é responsável pelo encaminhamento de informação, ou seja, fazer chegar a informação ao destino.

O modelo TCP/IP utiliza habitualmente o protocolo IP na camada de rede. Os protocolos usados nesta camada utilizam algoritmos de encaminhamento que utilizam determinadas métricas para a construção das tabelas de encaminhamento.

No próximo capítulo vamos estudar os algoritmos clássicos de encaminhamento, tais como, o *Distance-Vector* e o *Link-State* bem como o método Dijkstra. No mesmo capítulo será estudado o primeiro algoritmo ACO criado para as redes com fios nomeadamente o *AntNet*.

CAPÍTULO

3

Algoritmos de Encaminhamento

Neste capítulo pretendemos descrever os algoritmos de encaminhamentos vulgarmente conhecidos como clássicos e analisar o mecanismo de funcionamento que permite efectuar o encaminhamento de pacotes na rede. Estudaremos também o mecanismo de funcionamento, que está por detrás do primeiro algoritmo ACO utilizado para redes de dados com fios.

3.1 Visão geral

O encaminhamento é tido como um mecanismo que permite que um pacote seja entregue ao seu destino. Para que haja encaminhamento é necessário que hajam as tabelas de encaminhamento que são construídas a partir dos algoritmos de encaminhamento. Assim, este capítulo tem como objectivo dar a conhecer alguns dos algoritmos de encaminhamento pioneiros, sendo que para isso descreveremos e classificaremos os algoritmos de encaminhamento nas suas diferentes categorias. Apresentaremos e descreveremos ainda alguns algoritmos de encaminhamento clássicos como o *Distance-Vector* e o *Link-State* bem como o método Dijkstra.

Para finalizar o capítulo estudaremos o algoritmo *AntNet*, começando por analisar o comportamento das colónias de formigas, para posterior percepção do funcionamento dos algoritmos ACO.

A descrição do mecanismo de funcionamento do algoritmo *AntNet* é feita sequencialmente, sendo primeiro apresentados os elementos principais que constituem este algoritmo; é também apresentada a estrutura constituinte dos nós deste algoritmo bem como a forma como são actualizadas as tabelas de encaminhamento.

3.2 Descrição e classificação

Um dos requisitos fundamentais de uma rede de comunicação é (re)encaminhar eficientemente os pacotes a apartir de um nó de origem até um ou mais nós de destino. Para isso é preciso criar ou determinar um caminho entre esses nós. O caminho pode ser definido manualmente (pelo administrador da rede) - encaminhamento estático, ou recorrendo aos algoritmos de encaminhamentos - encaminhamento dinâmico. Os objectivos dos algoritmos de encaminhamento, são definidos pelas exigências da rede

e pelo tipo de serviço que se pretende oferecer.

Embora os objectivos dos algoritmos de encaminhamentos sejam diferentes para diferentes tipos de redes, estes podem ser classificados em duas categorias [Kaufmann, 2007]:

algoritmos orientados ao utilizador: a rede que utiliza estes tipos de algoritmos, deve fornecer um bom serviço ao utilizador, i.e., de tal forma que os pacotes devem ser enviados do nó de origem ao nó de destino de forma rápida;

algoritmos orientados à rede: a rede com este tipo de algoritmos, deve garantir o fornecimento de um serviço eficiente e um encaminhamento justo de tal modo que a maioria dos utilizadores possam desfrutar de um bom e aceitável serviço, em detrimento de um melhor serviço para apenas alguns utilizadores; como é o caso dos algoritmos orientados ao utilizador. Neste tipo de encaminhamentos tenta-se encontrar um equilíbrio entre o bom e o aceitável. Tal equilíbrio torna-se necessário, porque existem recursos de rede limitados, como é o caso da largura de banda.

A par desses objectivos, o encaminhamento possui o problema que resume-se em encontrar um caminho com o menor custo (distância, preço ou outro parâmetro) relativamente aos restantes caminhos disponíveis na rede, para um dado destino. As tabelas de encaminhamento são construídas com base nesse custo. O cálculo desse custo é feito recorrendo aos chamados algoritmos de encaminhamentos. Estes algoritmos podem ser classificados quanto [De Araújo, 1999]; [Di Caro, 2004]; [Thomas, 2006]:

- à localização: centralizados e distribuídos;
- ao tipo de encaminhamento: estáticos ou dinâmicos;

- à decisão de encaminhamento: *source* ou *hop-by-hop*[1] .

Nos **algoritmos centralizados**, um nó, tido como central, é responsável por actualizar as tabelas de encaminhamento de todos os nós da rede e por efectuar todas as decisões de encaminhamento. Estes tipos de algoritmos são usados em redes pequenas e em casos específicos, pois criam uma sobrecarga nas comunicações ao nível do nó central, além de que qualquer falha neste nó paralisa toda a rede.

Os **algoritmos distribuídos** possuem uma filosofia de funcionamento diferente dos algoritmos centralizados. Nestes, as decisões de encaminhamento são feitas ao nível de cada nó da rede, sendo cada nó responsável por actualizar a sua própria tabela de encaminhamento com base em informações obtidas da rede.

Nos **algoritmos estáticos** a definição das tabelas de encaminhamento é feita inicialmente ou quando ocorre alguma alteração da topologia da rede. Neste tipo de algoritmos não existe a adaptação do encaminhamento ao tráfego da rede.

Os **algoritmos dinâmicos** são adaptativos, i.e., o encaminhamento é alterado de acordo com as condições de tráfego bem como quando ocorre algumas alteração da topologia da rede.

Nos **algoritmos *source*** o caminho que um dado pacote deve seguir na rede é definido no nó de origem do mesmo e transportado no seu cabeçalho. Em cada nó intermédio, é lido esse cabeçalho e reencaminhado para o next hop de acordo o caminho definido pelo nó de origem.

Nos **algoritmos *hop-by-hop*** o caminho não é definido no nó de origem mas sim

[1] *Hop-by-hop* será considerado como nó-a-nó.

os nós intermédios, que vão definindo os *next hops* para os pacotes até que estes alcancem o destino.

3.3 Algoritmo de encaminhamento *Distance-Vector*

O algoritmo de encaminhamento *Distance-Vector* (DV), também conhecido como algoritmo de *Bellman-Ford* ou de *Ford-Fulkerson*, é um algoritmo que pode ser classificado quanto a localização como distribuído e pertence a categoria dos algoritmos orientados ao utilizador usado por diversos protocolos de encaminhamento [Thomas, 2006]. Quanto ao tipo e decisão de encaminhamento é um algoritmo estático e *hop-by-hop*, respectivamente.

Neste algoritmo os nós trocam informações com os seus vizinhos acerca dos custos, sem conhecimentos da forma como estes determinam esses custos, convergindo para uma situação em que cada nó conhece o caminho mais curto (critério de menor custo) para todos os nós da rede. Trata-se de um algoritmo assíncrono, pois as mensagens trocadas entre cada nó e os seus vizinhos, não são feitas simultaneamente.

O DV é uma algoritmo iterativo, pois o processo de convergência, relativo à construção das tabelas de encaminhamento para cada nó da rede, é feito em vários passos, até que cada nó conheça toda a informação de encaminhamento da rede.

No algoritmo de encaminhamento DV cada nó mantém uma tabela dos custos para os nós vizinhos, que são actualizadas com uma certa periodicidade.

Para exemplificar o funcionamento deste algoritmo, consideremos a rede representada na Figura 3.1 e vejamos como poderá ser calculado o caminho mais curto entre os nós A e F. Para tal, vamos introduzir as seguintes notações:

- d_{ik} custo da ligação entre o nó i e o nó vizinho k;

- D_{kj} custo mínimo (k, j) do caminho entre o nó k e j.

O Custo mínimo (i, j) é calculado de forma iteractiva de acordo com a expressão:

$$D_{ij} = \min_{(k \in N_i \wedge k \neq i \wedge i \neq j)} \{d_{ik} + D_{kj}\}, \tag{3.1}$$

onde N_i é o conjunto de nós vizinhos do nó i.

Consideremos ainda que os custos entre cada par de nós adjacentes são: $d_{AB} = d_{BA} = 1$, $d_{AD} = d_{DA} = 2$, $d_{AC} = d_{CA} = 1$, $d_{BC} = d_{CB} = 1$, $d_{BD} = d_{DB} = 3$, $d_{CE} = d_{EC} = 1$, $d_{DE} = d_{ED} = 2$, $d_{CF} = d_{FC} = 5$, e $d_{EF} = d_{FE} = 1$ (Figura 3.1).

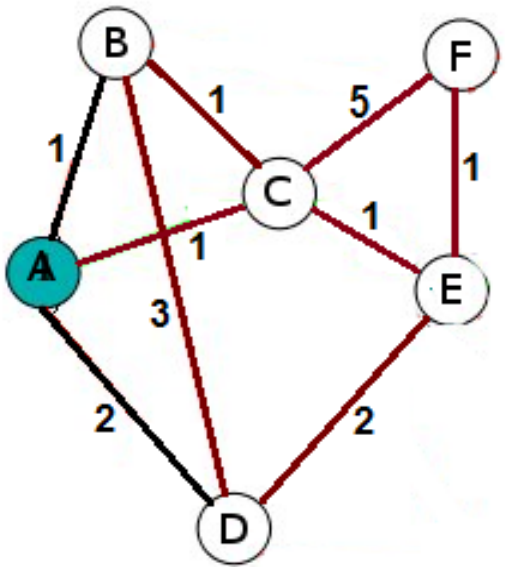


Figura 3.1: Rede com 6 nós, 9 ligações e os respectivos custos.

Considerando que no instante 0, nenhum nó da rede recebeu mensagens dos custos por parte dos nós vizinhos. Neste instante, os custos $D_{kj} = \infty$, excepto nos casos em que o nó vizinho é o nó de destino ($j = F$ sendo $D_{FF} = 0$).

Os custos mínimos para o nó F, nos diversos nós da rede são calculadas de acordo com a expressão (3.1), obtendo-se os valores presentes na Tabela 3.1. Pelo

Tabela 3.1: Cálculo das custos dos nós *A,B,C,D e E* para *F*, no instante incial (instante *0*).

Instante 0: nenhum nó trocou mensagens com o seu vizinho						
Nós	D_{kj}				Cálculo do D_{ij}	Caminho
Nó A	$D_{BF}=\infty$	$D_{DF}=\infty$	$D_{CF}=\infty$	-	$D_{AF}=\min\{d_{AB}+D_{BF}; d_{AD}+D_{DF}; d_{AC}+D_{CF}\}=$ $\min\{1+\infty; 2+\infty; 1+\infty\}=\infty$	-
Nó B	$D_{AF}=\infty$	$D_{CF}=\infty$	$D_{DF}=\infty$	-	$D_{BF}=\min\{d_{BC}+D_{CF}; d_{BD}+D_{DF}; d_{BA}+D_{AF}\}=$ $\min\{1+\infty; 3+\infty; 1+\infty\}=\infty$	-
Nó C	$D_{EF}=\infty$	$D_{BF}=\infty$	$D_{AF}=\infty$	$D_{FF}=0$	$D_{CF}=\min\{d_{CE}+D_{EF}; d_{CB}+D_{BF}; d_{CA}+D_{AF}; d_{CF}+D_{FF}\}=$ $\min\{1+\infty; 1+\infty; 1+\infty; 5+0\}=5$	C-F
Nó D	$D_{AF}=\infty$	$D_{BF}=\infty$	$D_{EF}=\infty$	-	$D_{DF}=\min\{d_{DE}+D_{EF}; d_{DB}+D_{BF}; d_{DA}+D_{AF}\}=$ $\min\{2+\infty; 3+\infty; 2+\infty\}=\infty$	-
Nó E	$D_{CF}=\infty$	$D_{DF}=\infty$	$D_{FF}=0$	-	$D_{EF}=\min\{d_{EF}+D_{FF}; d_{EC}+D_{CF}; d_{ED}+D_{DF}\}=$ $\min\{1+0; 1+\infty; 2+\infty\}=1$	E-F

cálculo, vemos que neste instante o nó A não tem o caminho para o nó F (pois $D_{AF} = \infty$). Nota-se que os nós C e E possuem o caminho para o nó F pois este é vizinho daqueles. Após a primeira troca de mensagens, os nós A, B e E passam a saber que C possui um caminho para F, uma vez que o $D_{CF} \neq \infty$. Da mesma forma, os nós C e D também passam a saber que E possui um caminho para F, pois o $D_{EF} \neq \infty$. Assim, no instante 1 são calculados os novos custos para o nó F a partir de cada nó da rede, com base na informação disponível em cada nó, tal como apresentado na Tabela 3.2.

Tabela 3.2: Cálculo dos custos dos nós *A,B,C,D e E* para *F*, no instante *1*.

Instante 1: primeira recepção de mensagens						
Nós	D_{kj}				Cálculo do D_{ij}	Caminho
Nó A	$D_{BF}=\infty$	$D_{DF}=\infty$	$D_{CF}=5$	-	$D_{AF}=\min\{d_{AB}+D_{BF}; d_{AD}+D_{DF}; d_{AC}+D_{CF}\}=$ $\min\{1+\infty; 2+\infty; 1+5\}=6$	A-C-F
Nó B	$D_{AF}=\infty$	$D_{CF}=5$	$D_{DF}=\infty$	-	$D_{BF}=\min\{d_{BC}+D_{CF}; d_{BD}+D_{DF}; d_{BA}+D_{AF}\}=$ $\min\{1+5; 3+\infty; 1+\infty\}=6$	B-C-F
Nó C	$D_{EF}=1$	$D_{BF}=\infty$	$D_{AF}=\infty$	$D_{FF}=0$	$D_{CF}=\min\{d_{CE}+D_{EF}; d_{CB}+D_{BF}; d_{CA}+D_{AF}; d_{CF}+D_{FF}\}=$ $\min\{1+1; 1+\infty; 1+\infty; 5+0\}=2$	C-E-F
Nó D	$D_{AF}=\infty$	$D_{BF}=\infty$	$D_{EF}=1$	-	$D_{DF}=\min\{d_{DE}+D_{EF}; d_{DB}+D_{BF}; d_{DA}+D_{AF}\}=$ $\min\{2+1; 3+\infty; 2+\infty\}=3$	D-E-F
Nó E	$D_{CF}=5$	$D_{DF}=\infty$	$D_{FF}=0$	-	$D_{EF}=\min\{d_{EF}+D_{FF}; d_{EC}+D_{CF}; d_{ED}+D_{DF}\}=$ $\min\{1+0; 1+5; 2+\infty\}=1$	E-F

Nesta iteração e resulado da simplicidade da rede, todos os nós da rede passam

a ter um caminho para o nó F. Comparando as Tabelas 3.1 e 3.2 notamos que inicialmente (instante 0), o nó C possuía um caminho para o nó F com um custo de 5 (Caminho $C - F$). Neste instante surge a possibilidade de alcançar o nó F através do E (a partir do C) com um custo menor que o anterior. No instante 2 (Tabela 3.3), calculam-se novamente os custos mínimas para o nó F, a partir de cada nó da rede.

Tabela 3.3: Cálculo dos custos dos nós *A, B, C, D e E* para *F*, no instante *2*.

Instante 2: segundo recepção de mensagens						
Nós	D_{kj}				Cálculo do D_{ij}	Caminho
Nó A	D_{BF}=6	D_{DF}=3	D_{CF}=2	-	D_{AF}=min{d_{AB} +D_{BF}; d_{AD} +D_{DF}; d_{AC} +D_{CF}}= min{1+6; 2+3; 1+2} = 3	A-C-E-F
Nó B	D_{AF}=6	D_{CF}=2	D_{DF}=3	-	D_{BF}=min{d_{BC} +D_{CF}; d_{BD} +D_{DF}; d_{BA} +D_{AF}}= min{1+2; 3+3; 1+6} = 3	B-C-E-F
Nó C	D_{EF}=1	D_{BF}=6	D_{AF}=6	D_{FF}=0	D_{CF}=min{d_{CE} +D_{EF}; d_{CB} +D_{BF}; d_{CA} +D_{AF}; d_{CF} +D_{FF} }= min{1+1; 1+6; 1+6; 5+0} = 2	C-E-F
Nó D	D_{AF}=6	D_{BF}=6	D_{EF}=1	-	D_{DF}=min{d_{DE} +D_{EF}; d_{DB} +D_{BF}; d_{DA} +D_{AF}}= min{2+1; 3+6; 2+6} = 3	D-E-F
Nó E	D_{CF}=2	D_{DF}=3	D_{FF}=0	-	D_{EF}=min{d_{EF} +D_{FF}; d_{EC} +D_{CF}; d_{ED} +D_{DF}}= min{1+0; 1+2; 2+3} = 1	E-F

Este cálculo conduz à escolha de um novo caminho entre A e F que possui um menor custo, se comparado com o calculado no instante 1.

No instante 3, efectuam-se novamente os cálculos dos custos mínimos para F (Tabela 3.4).

Tabela 3.4: Cálculo dos custos dos nós *A,B,C,D e E* para *F*, no instante *3*.

Instante 3: terceira recepção de mensagens						
Nós	D_{kj}				Cálculo do D_{ij}	Caminho
Nó A	D_{BF}=3	D_{DF}=3	D_{CF}=2	-	D_{AF}=min{d_{AB} +D_{BF}; d_{AD} +D_{DF}; d_{AC} +D_{CF}}= min{1+3; 2+3; 1+2} = 3	A-C-E-F
Nó B	D_{AF}=3	D_{CF}=2	D_{DF}=3	-	D_{BF}=min{d_{BC} +D_{CF}; d_{BD} +D_{DF}; d_{BA} +D_{AF}}= min{1+2; 3+3; 1+3} = 3	B-C-E-F
Nó C	D_{EF}=1	D_{BF}=3	D_{AF}=3	D_{FF}=0	D_{CF}=min{d_{CE} +D_{EF}; d_{CB} +D_{BF}; d_{CA} +D_{AF}; d_{CF} +D_{FF} }= min{1+1; 1+3; 1+3; 5+0} = 2	C-E-F
Nó D	D_{AF}=3	D_{BF}=3	D_{EF}=1	-	D_{DF}=min{d_{DE} +D_{EF}; d_{DB} +D_{BF}; d_{DA} +D_{AF}}= min{2+1; 3+3; 2+3} = 3	D-E-F
Nó E	D_{CF}=2	D_{DF}=3	D_{FF}=0	-	D_{EF}=min{d_{EF} +D_{FF}; d_{EC} +D_{CF}; d_{ED} +D_{DF}}= min{1+0; 1+2; 2+3} = 1	E-F

Estes cálculos não alteram os custos obtidas na Tabela 3.3. Em conformidade com o que foi dito no inicio desta secção, esta tabela é actualizada periodicamente, conduzindo à um cálculo, dos custos, a tender para o infinito. Assim, se os custos d_{ik} se mantiverem constantes e não houver alteração na topologia da rede (quebra de ligação ou entrada dum novo nó na rede), qualquer troca de mensagem bem como qualquer cálculo dos custos mínimos para F não alterará os resultados da Tabela 3.4 [Kaufmann, 2007].

O exemplo anterior mostrou-nos como é feita a construção das tabelas de encaminhamento no algoritmo DV através dos custos (D_{kj}).

O algoritmo 1 apresenta os passos essenciais do processo descrito.

Algoritmo 1 Pseudocódigo do algoritmo *Distance-Vector*

1: $D_{kj}(t = 0) \Leftarrow \infty$ {i-nó de origem, j-nó de destino}
2: $D_{jj}(t = 0) \Leftarrow 0$; {t-instante de tempo}
3: {ciclo infinito}
4: **while** $t > 0$ **do**
5: **for** $j = 1$ to N **do**
6: $D_{ij} \Leftarrow min_{(k \in N_i \wedge k \neq i \wedge i \neq j)}\{d_{ik} + D_{kj}\}$ {N-número de nós da rede e N_i é o conjunto de nós vizinhos do nó i}
7: $D_{kj} \Leftarrow D_{ij}$
8: **end for**
9: **end while**

Um factor a levar em consideração no algoritmo DV é o facto de cada nó da rede conhecer apenas o custo para os seus vizinhos (pois os outros custos são transmitidos pelos vizinhos) e não de todo o caminho por onde deve seguir o pacote na rede para alcançar um dado destino. A grande vantagem deste algoritmo é a sua facilidade de implementação, sendo muito úteis em redes bastante pequenas. Os protocolos de encaminhamento mais populares que usam estes algoritmos são o RIP e o BGP [Thomas, 2006].

3.4 Algoritmo de encaminhamento *Link-State*

No algoritmo de encaminhamento *Link-State*, criado por McQuillan *et* al. [1978] para resolver o problema do tempo de convergência do algoritmo DV [Ducatelle, 2007], cada nó da rede mantém os custos das ligações de toda a rede.

Neste algoritmo cada nó, primeiro identifica os vizinhos, calculando a seguir o custo da ligação entre si e os seus vizinhos. Utilizando mensagens enviadas em *floading*, chamadas LSA's [Rodriguez *et* al. , 2001], todos os nós criam uma base de dados com toda a informação da topologia de rede, facto que cria uma grande diferença entre este algoritmo e o DV. Como vimos na secção 3.3, os nós DV apenas contém informações relativas aos seus vizinhos e não de toda a topologia de rede.

A base de dados com a informação da topologia de rede criada pelos nós *Link-State* é utilizada para determinar o menor caminho para cada um dos destinos da rede. Para o efeito usa o método Dijkstra, para construir ou actualizar as tabelas de encaminhamento, conforme será descrito a seguir.

3.4.1 Método Dijkstra

O princípio de funcionamento do método Dijkstra é baseado no conceito de que todos os nós da rede trabalham em conjunto para encontrar o caminho com o menor custo para todos os nós destinos da rede. Este método calcula o caminho mais curto a partir de cada nó de origem, para todos os destinos da rede [Kaufmann, 2007].

Para exemplificar o seu funcionamento vamos considerar novamente a rede apresentada na Figura 3.1, supondo que o nó A deseja conhecer o caminho que possue o menor custo para cada um dos nós da rede. Tal como para o algoritmo DV, vamos definir três parâmetros:

- d_{kj} custo da ligação entre o nó k e j;

- D_{ij} custo mínimo do caminho entre o nó i e j,
- $\overline{D_{ij}}$ custo mínimo temporário do caminho entre o nó i e j.

Inicialmente considera-se que os nós vizinhos de A tem $\overline{D_{ij}} = d_{kj}$ (onde $k = i$) enquanto que os restantes destinos tem custo $\overline{D_{ij}} = \infty$. Como resultado temos: $\overline{D_{AB}} = 1$, $\overline{D_{AC}} = 1$, $\overline{D_{AD}} = 2$ enquanto que $\overline{D_{AE}} = \infty$, $\overline{D_{AF}} = \infty$.

Começando pelo nó A construimos duas tabelas, uma contendo as ligações e caminhos que vão surgindo ao longo da procura dos caminhos com o menor custo, $\overline{D_{ij}}$, na rede e outra contendo apenas os caminhos com o menor custo, D_{ij}. O objectivo da primeira tabela é de comparar os custos temporários para posterior criação da segunda tabela contendo os custos definitivos.

Assim, sendo que o nó A possui três ligações, uma para cada um dos nós vizinhos, surgem três possibilidades cada uma delas associada ao seu custo temporário, conforme podemos ver na Tabela 3.5

Tabela 3.5: Tabela de procura, usada para determinar o custo mínimo dos nós da rede Figura 3.1, contendo as ligações, caminhos e custos temporários, construída a partir do nó A.

Ligação	d_{kj}	Caminho a partir da origem A	$\overline{D_{ij}}$
$A-B$	$d_{AB}=1$	$A-B$	$\overline{D_{AB}} = d_{AB} = 1$
$A-C$	$d_{AC}=1$	$A-C$	$\overline{D_{AC}} = d_{AC} = 1$
$A-D$	$d_{AD}=2$	$A-D$	$\overline{D_{AD}} = d_{AD} = 2$

A escolha do caminho com o menor custo para um dos destinos é feita recorrendo a expressão:

$$D_{ij} = \min\{\overline{D_{ij}}\} \tag{3.2}$$

onde:

- D_{ij} corresponde ao custo mínimo a escolher do nó i para o destino j, calculado a partir dos valores de $\overline{D_{ij}}$ da Tabela 3.5;
- $\overline{D_{ij}} = D_{ik} + d_{kj}$ corresponde ao custo mínimo temporário. De salientar que inicialmente $\overline{D_{ij}} = d_{kj}$ $(k = i)$.

Assim sendo:

$$D_{ij} = \min\{\overline{D_{AB}}; \overline{D_{AC}}; \overline{D_{AD}}\} = min\{1; 1; 2\} = \{1; 1\} \tag{3.3}$$

Uma vez que existe um mínimo duplo associado aos nós B e C, escolhemos qualquer um deles. Suponhamos que é escolhido o nó B, ou seja, o caminho $A - B$ com um custo $D_{ij} = D_{AB} = \overline{D_{AB}} = 1$. Assim o nó A passa a ter o caminho para o nó B através da ligação $A - B$ (Figura 3.2).

Retiramos o caminho $A - B$ e o seu custo associado da Tabela 3.5 de procura para a tabela dos custos mínimos deste nó (Tabela 3.6). Eliminamos a linha $A - B$ da Tabela 3.5, de procura permanecendo as linhas ilustradas na Tabela 3.7.

Tabela 3.6: Custo mínimo e o caminho a partir do nó A para o nó B da rede da Figura 3.1, construída a partir da Tabela 3.5.

Caminho a partir da origem A	D_{ij}
$A - B$	$D_{AB} = 1$

Tabela 3.7: Linhas restantes da Tabela 3.5 de procura, depois de removida a linha com o menor custo.

Ligação	d_{kj}	Caminho a partir da origem A	$\overline{D_{ij}}$
$A - C$	$d_{AC} = 1$	$A - C$	$\overline{D_{AC}} = d_{AC} = 1$
$A - D$	$d_{AD} = 2$	$A - D$	$\overline{D_{AD}} = d_{AD} = 2$

Uma vez que foi escolhido o caminho $A - B$, ou seja, o caminho do nó A até o nó B, então fixamos o nó B, e acrescentamos na Tabela 3.7 as ligações deste nó

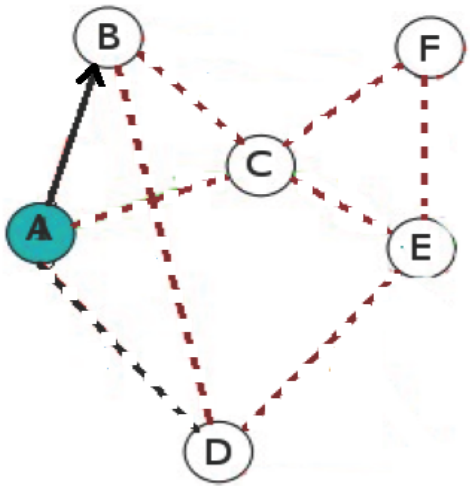


Figura 3.2: Caminho com o custo mínimo entre os nós A e B determinado usando o método Dijkstra.

aos seus vizinhos, excluindo o nó A e os nós cujos destinos já são conhecidos. Assim, surgem duas novas ligações, $B-C$ e $B-D$, que podem ser visualizadas na Tabela 3.8.

Tabela 3.8: Tabela de procura resultante da edição das linhas derivadas das ligações do nó B aos seus vizinhos (excepto nó de origem, A, e nós cujos destinos são conhecidos) à Tabela 3.8.

Ligação	d_{kj}	Caminho a partir da origem A	$\overline{D_{ij}}$
$A-C$	$d_{AC}=1$	$A-C$	$\overline{D_{AC}}=d_{AC}=1$
$A-D$	$d_{AD}=2$	$A-D$	$\overline{D_{AD}}=d_{AD}=2$
$B-C$	$d_{BC}=1$	$A-B-C$	$\overline{D_{BC}}=D_{AB}+d_{BC}=1+1=2$
$B-D$	$d_{BD}=3$	$A-B-D$	$\overline{D_{BD}}=D_{AB}+d_{BD}=1+3=4$

Aplicando a expressão (3.2) aos custos da Tabela 3.8, o menor custo estará associado ao caminho entre o nó A e C.

Retiramos o caminho $A-C$ e o seu custo associado da Tabela 3.6, para a tabela dos custos mínimos desde do nó A, obtendo-se a Tabela 3.9. Na sequência,

eliminamos a linha $A - C$ da Tabela 3.8 resultando na Tabela 3.10.

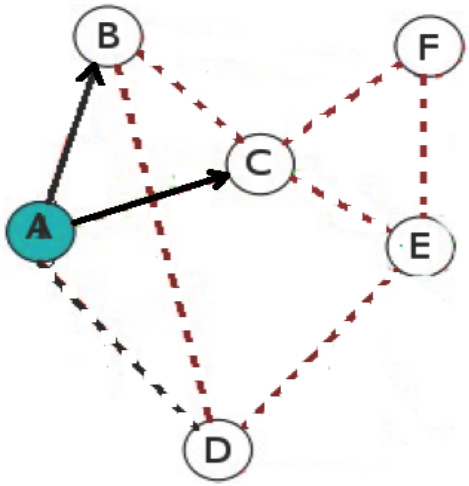


Figura 3.3: Caminhos com custos mínimos entre os nós A e B bem como A e C, determinado usando o método Dijkstra.

Tabela 3.9: Custos mínimos e o caminho a partir do nó A para os destinos B e C da rede Figura 3.1.

Caminho a partir da origem A	D_{ij}
$A - B$	$D_{AB} = 1$
$A - C$	$D_{AC} = 1$

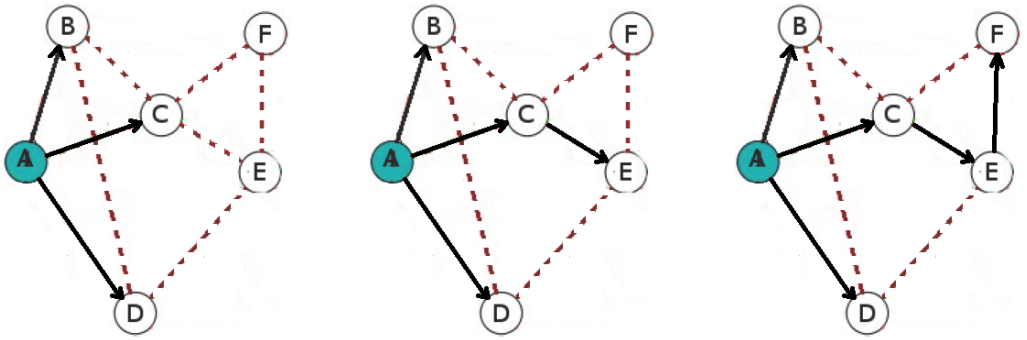
Tabela 3.10: Linhas restantes da Tabela 3.8, depois de removida a linha do caminho $A - C$

Ligação	d_{kj}	Caminho a partir da origem A	$\overline{D_{ij}}$
$A - D$	$d_{AD} = 2$	$A - D$	$\overline{D_{AD}} = d_{AD} = 2$
$B - C$	$d_{BC} = 1$	$A - B - C$	$\overline{D_{BC}} = D_{AB} + d_{BC} = 1 + 1 = 2$
$B - D$	$d_{BD} = 3$	$A - B - D$	$\overline{D_{BD}} = D_{AB} + d_{BD} = 1 + 3 = 4$

O procedimento repete-se até que sejam encontrados caminhos para todos os destinos da rede, conforme ilustrados na Tabela 3.11. Os passos subsequentes podem ser visualizados na Figura 3.4.

Tabela 3.11: Custos e o caminho a partir do nó A para todos os destinos dos nós da rede Figura 3.1.

Caminho a partir do nó de origem A	D_{ij}
$A - B$	$D_{AB} = 1$
$A - C$	$D_{AC} = 1$
$A - D$	$D_{AD} = 2$
$A - C - E$	$D_{AE} = 2$
$A - C - E - F$	$D_{AF} = 3$



(a) Criação do caminho entre o nó A e D.

(b) Criação do caminho entre o nó A e E.

(c) Criação do caminho entre o nó A e F.

Figura 3.4: Caminhos a partir do nó A para todos os nós da rede.

Este exemplo mostra como este algoritmo faz a procura dos caminhos com o menor custo a partir de cada nó para todos os destinos da rede. Deve-se ter em conta que, de acordo com a topologia da rede, o método Dijkstra pode consumir muitos recursos de memória em cada *router* (em redes densas).

O algoritmo 2 apresenta os passos assenciais do Dijkstra.

3.5 Algoritmo de encaminhamento *AntNet*

O algoritmo de encaminhamento *AntNet* foi concebido para o encaminhamento adaptativo de pacotes. Em redes com este algoritmo, os *routers* efectuam o (re)encaminhamento

Algoritmo 2 Pseudocódigo do método Dijkstra

1: Fixa o nó de origem i,
2: $\overline{D_{ij}} \Leftarrow d_{kj}$ {i-nó de origem, k-nó adjacente do i e j-nó de destino; $k = i$ (inicialmente)}
3: colocando as ligações $i - k's$ juntamente com os custos mínimos temporários $\overline{D_{ij}}$ na tabela S {S : tabela de caminhos que surgem na rede}
4: **for** todos os nós k **do**
5: Calcula o custo mínimo D_{ij} das ligações $i - k's$ da tabela S, usando a expressão (3.2)
6: Mover o $i - k$ do custo mínimo calculado da tabela S para tabela S' {S' : correspondente aos caminhos com custos mínimos}
7: Fixa o nó k e calcular o $\overline{D_{kk_m}} = D_{ik} + d_{kk_m}$, onde k_m são vizinhos de k
8: **end for**

de pacotes de forma inteligente, pois fazem um balanço entre as filas de espera e a preferência dos caminhos. A restante dissertação será dedicada ao estudo de algoritmos de encaminhamento que utilizam o método do algoritmo *AntNet*.

3.5.1 A Meta heurística *Ant Colony Optimization*

Descrição do Comportamento das Colónias de formigas - *Ant Colony* (AC)

Entre os finais de 1989 e inícios de 1990 um grupo de investigadores da Universidade Livre de Bruxelas fez sucessivas experiências com formigas reais, obtendo o fundamento teórico sobre a influência das feromonas[2] no comportamento das colónias de formigas. Os resultados obtidos levaram a concluir que as feromonas actuavam como uma memória dinâmica e colectiva da colónia. Este facto é explicado da seguinte forma:

- quando uma formiga sai do seu formigueiro à procura de alimentos, vai depo-

[2] Feromonas são substâncias químicas segregadas por animais com o intuito de induzir determinadas reacções noutros membros da mesma espécie.

sitando determinada quantidade de feromonas pelo caminho por onde passa. A feromona deixada traduz-se em informações da experiência que esta formiga teve;

- quando a formiga encontra uma fonte de alimentos, volta para a colónia pelo mesmo caminho, mas em sentido contrário, deixando mais feromonas que actualizam de forma permanente a informação acerca do caminho para a fonte de alimentos encontrada. Essa actualização comunica e/ou influencia, de forma indirecta e através do ambiente, as outras formigas (propriedade de estigmergia). Assim, as formigas que encontrarem aquele rasto de feromonas seguem aquele caminho.

Este tipo de comunicação, associada à resposta positiva por parte das outras formigas, podem ser suficientes para permitir que os elementos do formigueiro descubram o caminho mais curto, entre o formigueiro e a fonte de alimentos, dentre vários caminhos existentes.

Para exemplificar o comportamento de uma colónia de formigas, vamos considerar os triângulos equiláteros da Figura 3.5, onde um dos vértices da base representa o formigueiro e o adjacente, na mesma base, representa a fonte de alimentos.

Do formigueiro partem duas formigas em direcção à fonte de alimentos. Vamos considerar que não há evaporação das feromonas. Deste modo, podemos considerar os seguintes passos:

(1) no instante $t = 0$, não existe nenhum rasto de feromonas nos dois caminhos (A e B) que separam o formigueiro da fonte de alimentos. Neste caso a probabilidade de qualquer um dos caminhos ser escolhido é a mesma. Vamos supor que neste instante uma formiga escolhe aleatoriamente o caminho A e a outra o caminho B.

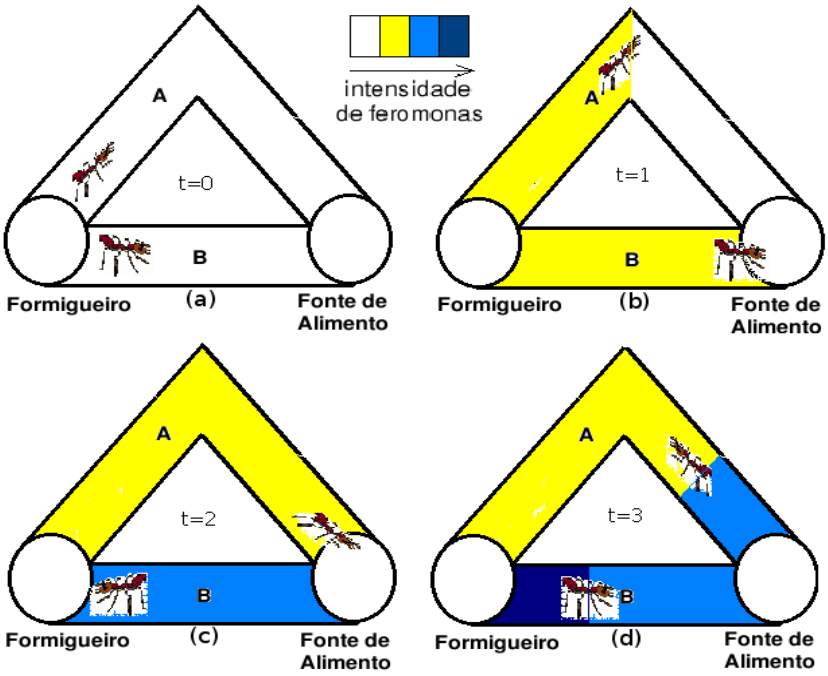


Figura 3.5: Ilustração do depósito de feromonas por duas formigas que seguem dois caminhos distintos do formigueiro à fonte de alimentos (instante $t = 0$). No instante $t = 1$, a formiga que segue pelo caminho B alcança a fonte de alimentos, enquanto que a formiga que seguiu pelo caminho A encontra-se a metade do caminho. No seu regresso ao formigueiro, a formiga que segue pelo caminho B deposita mais feromonas aumentando a intensidade da mesma nesse caminho (instante $t = 2$). Quando a formiga que segue pelo caminho B chega ao formigueiro a outra alcança a fonte de alimentos. Supondo que a formiga que seguiu por A volta a escolher o mesmo caminho para regressar ao formigueiro e que a formiga que se encontra no formigueiro escolhe o caminho B, a formiga que seguiu por B volta a depositar rastos de feromonas, aumentando ainda mais a sua intensidade sobre B e reduzindo a probabilidade do caminho A ser escolhido por outras formigas (instante $t = 3$), no percurso entre o formigueiro e a fonte de alimentos e vice versa.

(2) partindo do pressuposto que as formigas vão deixando rastos de feromonas pelo caminho, no instante $t = 1$, a formiga que escolheu o caminho B alcança a fonte de alimentos deixando o primeiro rasto de feromonas por todo o caminho, enquanto que a formiga que seguiu pelo caminho A ainda não alcançou a mesma e o rasto de feromonas ainda esta pela metade do caminho. Nesse

mesmo instante a formiga que seguiu pelo caminho B volta ao formigueiro e a outra continua o seu percurso até a fonte de alimentos.

(3) no instante $t = 2$, a formiga que seguiu pelo caminho A alcança finalmente a fonte de alimentos preenchendo o rasto de feromonas no mesmo. Em simultâneo, a formiga que seguiu pelo caminho B, já se encontra de volta ao formigueiro e intensificou a quantidade de feromonas existentes por esse mesmo caminho. Esta intensidade é vista pelo gráfico de intensidade de feromonas representada na parte superior da Figura 3.5.

(4) ainda no instante $t = 2$, a formiga que se encontra no formigueiro prepara-se para voltar à fonte de alimentos, possuindo duas possibilidades de escolha do caminho por onde seguir (A ou B). No caminho B temos uma maior intensidade de feromonas em relação ao caminho A, em termos de relação, temos uma relação de 2 para 1, respectivamente; o que significa que o caminho B possui maior probabilidade de ser escolhido pela formiga que se encontra no formigueiro, em relação ao caminho A. Suponhamos que a formiga que se encontra no formigueiro escolhe de novo o caminho B, e que a formiga que se encontra na fonte de alimentos escolhe o caminho A. Durante os seus percursos aumentarão ainda mais a quantidade de feromonas presentes nos dois caminhos (instante $t = 3$). No caminho B temos uma maior intensidade de feromonas, se comparado com o caminho A, o que reduz significativamente a probabilidade deste ser escolhido pelas formigas que vão do formigueiro à fonte de alimentos e vice versa. Isto se deve ao facto da taxa de chegada das formigas à fonte de alimentos pelo caminho B ser maior se comparada com a do caminho A [Di Caro, 2004]; [Ferreira *et* al. , 2008].

Um facto importante a reter é de que os insectos sociais, como é o caso das formigas, estão distribuídos em sistemas, que apesar da sua simplicidade individual, apresentam uma grande estrutura de organização social, permitindo-lhes ter comportamentos sociais iguais ao descrito.

O *Ant Colony Optimization* - ACO

Os algoritmos baseados em colónias de formigas foram introduzidos nos princípios da década de 1990. Estes, estudam um modelo computacional derivado da observação do comportamento das formigas reais. Este comportamento tem servido de fonte de inspiração para a concepção dos algoritmos *Ant Colony* (AC), que objectivam de solucionar problemas de optimização [Dorigo & Stutzle, 2004].

Os algoritmos AC mais bem sucedidos são conhecidos como *Ant Colony Optimization* (ACO). Considerando a modelação para problemas de redes, os algoritmos ACO usam formigas artificias que, em comparação com as formigas reais, possuem a seguinte filosofia de funcionamento [Blum & Merkle, 2008]:

- as formigas artificiais movem-se da origem a um destino determinado, enquanto que as formigas reais partem do formigueiro à procura de uma fonte de alimento desconhecida;
- as formigas artificiais apenas deixam o rasto de feromonas quando fazem o caminho de volta para a origem, enquanto que as formigas reais deixam-no em ambos sentidos;
- a quantidade de feromonas usada para fazer a actualização do caminho, por parte das formigas artificiais, depende da qualidade da solução obtida, enquanto que as formigas reais depositam a mesma quantidade de feromonas pelo caminho por onde passam.

O primeiro algoritmo ACO criado e destinado a problemas discretos foi o *Ant System* (AS) [Dorigo *et* al. , 1991]; [Dorigo., 1992]. Foi aplicado pela primeira vez ao problema do caixeiro viajante (*Traveling Salesman Problem* - TSP) [Lawler *et* al. , 1985]; [Dorigo & Stutzle, 2004]; [Di Caro, 2004]. No AS cada formiga artificial é responsável por construir uma solução [Blum & Merkle, 2008]; [Da Silva, 2008]. Para descrever resumidamente este algoritmo vamos supor que temos um grafo $G(V, A)$ e pretendemos calcular o caminho mais curto entre um nó de origem, s, e de destino, d. Em cada intervalo de tempo Δt, uma formiga artificial inicia o seu percurso movendo-se do nó de origem para o de destino. Em cada instante t, a probabilidade da formiga artificial ser encaminhada a partir do nó k para o seu vizinho n ($n \in N_k$) é dada pela expressão [Dorigo *et* al. , 1999]:

$$P_{kn}(t) = \frac{[\tau_{kn}(t)]^{\alpha}[\eta_{kn}]^{\beta}}{\sum_{x \in N_k}[\tau_{kx}(t)]^{\alpha}[\eta_{kx}]^{\beta}} \tag{3.4}$$

onde:

τ_{kn} $(\sum_{n \in N_k} \tau_{kn} = 1)$ representa a intensidade de feromonas correspondente ao caminho entre os nós k e n;

$\eta_{kn} = \frac{1}{d_{kn}}$, representa o valor heurístico entre k e n, onde d_{kn} corresponde a distância entre os nós k e n;

α **e** β representam os pesos exponenciais dos parametros heurísticos τ_{kn} e η_{kn}, respectivamente.

Cada nó visitado pela formiga artificial, é armazenado numa memória, S, desta. Esta memória contém os registos de todos os nós por onde a formiga passa.

Quando a formiga alcança o nó de destino, volta ao nó de origem, pelo mesmo caminho mas em sentido contrário, actualizando a intensidade de feromonas existentes

pelo caminho. Esta actualização é feita depositando a quantidade de feromonas, $\Delta\tau(t) = \frac{1}{L}$ (L-comprimento do caminho percorrido pela formiga de s para d), em cada aresta do caminho que esta percorreu até d, de acordo com a seguinte expressão [Dorigo & Stutzle, 2004]:

$$\tau_{kn}(t) = \tau_{kn}(t) + \Delta\tau(t), \tag{3.5}$$

Após a actualização, é feita a evaporação de feromonas de acordo com a expressão:

$$\tau_{kn}(t) = (1 - \rho)\tau_{kn}(t) \tag{3.6}$$

onde: ρ representa o coeficiente de evaporação no intervalo durante o tempo t. Inicialmente $\tau_{kn}(0) = \tau_0$, $\tau_0 > 0$.

Ao nível das redes de dados com fios, foi criado o algoritmo ACO chamado *AntNet*, com o objectivo de procurar o melhor caminho a partir dos nós de origem para os vários destinos, baseando-se no principio acabado de descrever.

3.5.2 Descrição e caracterização do *AntNet*

Devido ao grande aumento do tráfego nas redes de dados, motivado pelo aumento de aplicações que transmitem informações através da Internet, houve a necessidade de melhorar a performance dessas redes de forma a tentar evitar grandes perdas de dados e proporcionar um bom serviço aos utilizadores. Até certo ponto, isto é possível graças ao melhoramento do mecanismo de encaminhamento da informação nessas redes, com o recurso aos algoritmos de encaminhamentos, responsáveis pela determinação dos caminhos por onde os pacotes devem seguir.

Neste sentido o *AntNet* procura encontrar o melhor caminho para o encaminhamento de pacotes, diferindo dos clássicos pelo facto de ser inspirado no comportamento de colónias de formigas na busca de alimentos; o que torna-o num algoritmo activo,

que dependendo das condições de tráfego procura com uma certa frequência novas soluções (caminhos) para o encaminhamento de pacotes.

Para descrever este algoritmo, vamos considerar que pretendemos procurar um caminho para o envio de pacotes a partir do nó de origem (s) para o nó de destino (d) da rede da Figura 3.6.

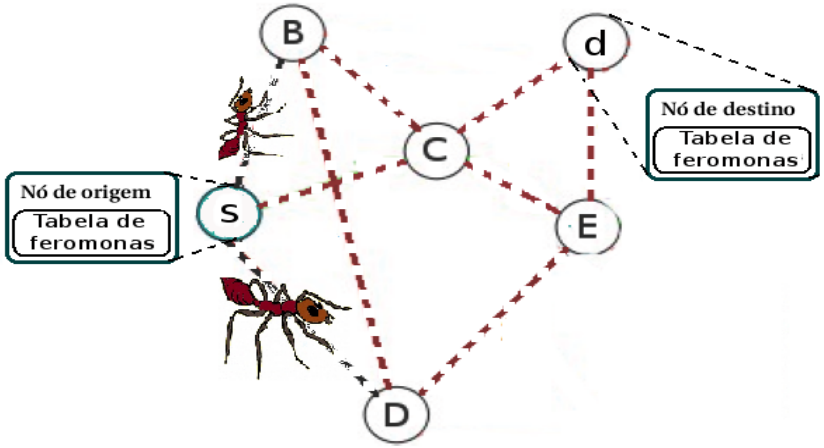


Figura 3.6: Rede com 6 nós e 9 arestas.

Para este algoritmo são definidas dois tipos de formigas (Figura 3.7):

- a ***formiga Forward***, representada por $F_{s \to d}$, que viaja desde o nó s até ao nó d, com o objectivo de encontrar um caminho para o envio de pacotes a partir do nó s para o d;

- a ***formiga Backward***, representada por $B_{s \to d}$, é gerada no nó d quando a formiga $F_{s \to d}$ chega a este, e tem como objectivo utilizar as informações recolhidas por esta formiga durante a caminho de s para d, para actualizar as tabelas de feromonas (e posteriormente as de encaminhamento) em todos

os nós por onde passou, incluindo o nó de origem.

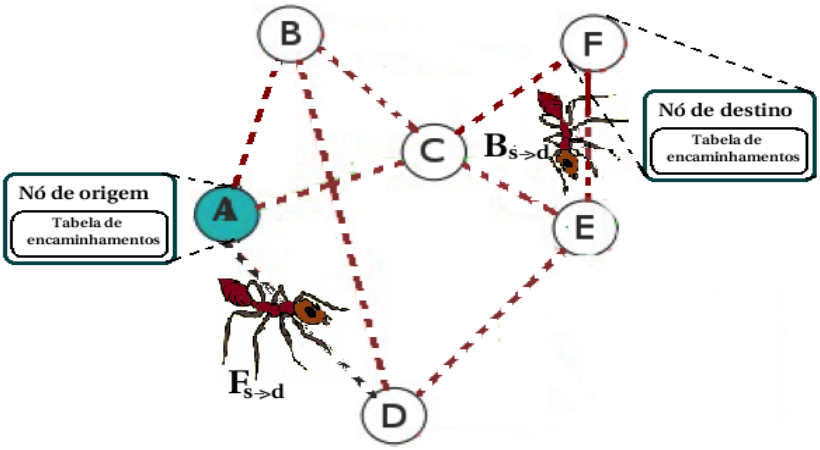


Figura 3.7: Ilustração das formigas $F_{s\to d}$ saindo do nó de origem e $B_{s\to d}$ voltando para o nó de origem eminstantes distintos.

Cada uma destas formigas representa um pacote de dados e transporta um conjunto de informações que permitem pseudo-aleatoriamente descobrir caminhos entre os nós definidos, como é o caso da formiga $F_{s\to d}$, e actualizar as tabelas de feromonas e as de encaminhamentos, como é o caso da formiga $B_{s\to d}$. A informação que estas formigas carregam está organizada numa matriz de memória representada por $S_{s\to d}$. Nesta matriz são armazenados os endereços dos nós e os custos associados às ligações entre esses nós, que são recolhidas por $F_{s\to d}$ (Figura 3.8).

Assim, em determinados intervalos de tempo, Δt, cada nó da rede envia uma formiga $F_{s\to d}$, com o objectivo de actualizar a informação de encaminhamento ou descobrir novos caminhos para todos os outros nós da rede. O destino, d, desta formiga é determinado pela expressão:

$$P_d = \frac{bits_{sd}}{\sum_{d' \in N} bits_{sd'}}, \tag{3.7}$$

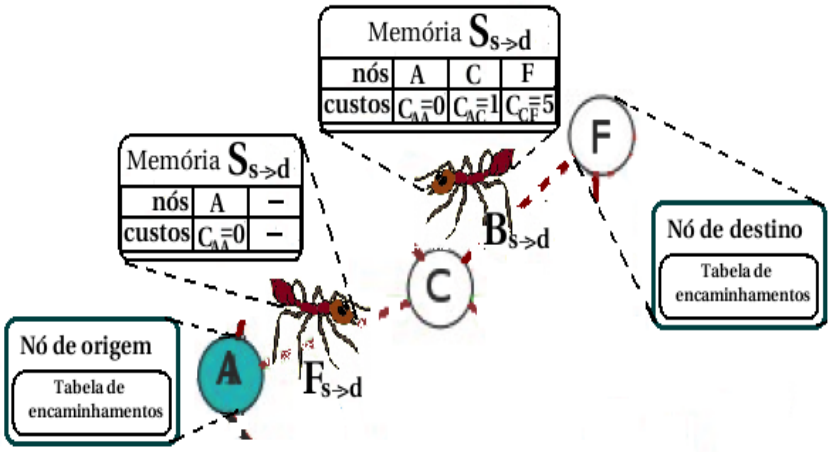


Figura 3.8: Ilustração da memória das formigas $F_{s\to d}$ e $B_{s\to d}$, onde constam os endereços dos nós bem como os custos associados às ligações entre os nós.

onde $bits_{sd}$ é o número de bits que foram transmitidos a partir do nó de origem, s, para o nó de destino d.

A expressão (3.7) dá-nos a probabilidade, P_d, de ser escolhido o nó d como destino para a formiga $F_{s\to d}$. O valor de P_d depende da quantidade de tráfego gerado a partir do nó s para os outros nós. Os nós para os quais está a ser enviado mais tráfego possuem maior probabilidade de serem escolhidos como destino da $F_{s\to d}$.

Determinado o destino, a $F_{s\to d}$ é lançada na rede. Durante o seu percurso os recursos que a rede disponibiliza para esta formiga são iguais aos dos outros pacotes da rede, ou seja, a formiga $F_{s\to d}$ partilha os mesmos critérios de atribuição de prioridades com os restantes pacotes da rede de modo a conhecer a situação real da rede. Salienta-se o facto de tanto a formiga artificial $F_{s\to d}$ como a $B_{s\to d}$ serem pacotes de dados.

A formiga *Forward*, $F_{s\to d}$, mantém em memória ($S_{s\to d}$) dois vectores ordenados onde armazena informação acerca do caminho que está a percorrer, nomeadamente o endereço dos nós por onde passa, $v_{v_1\to v_n} = [v_1, v_2, ..., v_n]$ ($s = v_1$ e $d = v_n$), e os custos associados a cada aresta $c_{v_1\to v_n} = [c_{v_1\to v_2}, c_{v_2\to v_3}, ..., c_{v_{n-1}\to v_n}]$.

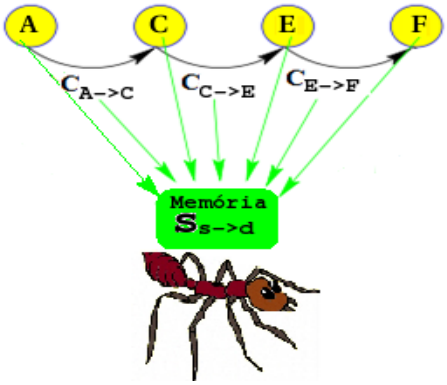


Figura 3.9: Exemplo do armazenamento dos endereços dos nós, v_i, bem como dos custos das ligações entre esses mesmos nós, $c_{v_s \to v_d}$, na memória $S_{s \to d}$ da formiga $F_{s \to d}$.

Em cada nó intermédio $v_i = k$, a probabilidade da formiga $F_{s \to d}$ ser encaminhada para qualquer um dos seus vizinhos $v_{i+1} = n$ é dada pela expressão:.

$$P_{nd} = \frac{\tau_{nd}^k + \alpha l_n}{1 + \alpha(|N_k| - 1)}, \qquad \forall n \in N_k \tag{3.8}$$

onde:

- $\tau_{nd}^k (\sum_{n \in N_k} \tau_{nd}^k = 1)$ representa a intensidade de feromonas na aresta (k, n), quando o destino é d;

- l_n representa o factor de correcção heurístico calculado com base na quantidade de pacotes presentes nas filas de espera do nó k que destinam-se ao nó vizinho n. l_n toma valores no intervalo $[0, 1]$ e é calculado de acordo com a expressão:

$$l_n = 1 - \frac{q_n}{\sum_{n'=1}^{|N_k|} q_{n'}} \tag{3.9}$$

onde q_n representa o número de pacotes presentes na fila de espera entre os nós k e n. Se o número de pacotes em espera na fila $k \to n$ for grande, comparativamente com a quantidade total de pacotes nas restantes filas, então l_n toma um valor próximo de 0, tomando um valor próximo de 1, caso contrário.

- $\alpha \in [0,1]$ representa o coeficiente de l_n, que atribui menor ou maior peso a este factor;

- N_k é o conjunto dos nós vizinhos de k.

Usando os valores de P_{nd}, $n \in N_k$, é escolhido pseudo-aleatoriamente um nó n, vizinho de k. Caso esse nó tenha sido visitado pela formiga, então esta é descartada; o que significa que para além dos factores apresentados na expressão (3.8), $S_{s \to d}$, a memória ocupa um papel importante no encaminhamento das formigas (Figura 3.10).

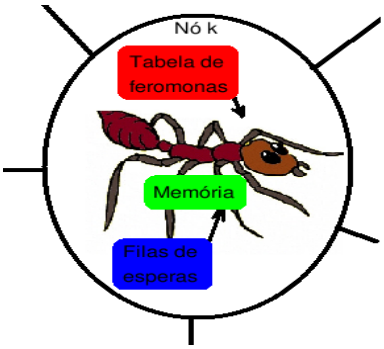


Figura 3.10: Factores de decisão de escolha do nó n por onde deve seguir a formiga $F_{s \to d}$ para alcançar o destino d.

Chegado ao nó de destino d, a formiga $F_{s \to d}$ é substituída pela formiga $B_{s \to d}$, criada neste nó, transferindo toda a sua memória, $S_{s \to d}$, para a última, sendo apagada de seguida.

A formiga $B_{s \to d}$ regressa até nó s pelo caminho realizado pela formiga $F_{s \to d}$, mas em sentido contrário; sendo encaminhada com a prioridade máxima[3] , de forma a

[3] Em redes que exijam determinado QoS, são definidos critérios para o encaminhamento dos pacotes que chegam a um dado nó (*router*). Assim quando os pacotes entram em determinado nó da rede, são marcados e colocados nas filas de espera correspondentes à marcação feita. A marcação desses pacotes depende do tipo de pacote que está a ser encaminhado e as filas de espera

actualizar com a maior rapidez possível as tabelas de feromonas e de encaminhamento dos nós nesse caminho, bem como a do nó de origem s. Esta actualização é feita comparando os custos da memória da formiga $B_{s \to d}$ com os custos existentes na tabela de encaminhamento do nó à actualizar, de acordo com a descrição que será feita a seguir [Dorigo & Stutzle, 2004].

Estrutura de dados dos nós que usam o *AntNet*

A estrutura de dados mantida nos nós que usam algoritmos de encaminhamento *AntNet* (Figura 3.11) é constituída por:

- $M_k(k = 1, 2, 3, 4, ..., N)$ que é um modelo estatístico local da rede. É utilizado para comparar os custos das diversas formigas *Forwards* que passaram pelo respectivo nó. Este modelo possui uma estrutura de dados com os parâmetros da média (μ_{kd}) e da variância (σ^2_{kd}) dos custos para alcançar o nó d a partir do nó k. Os parâmetros μ_{kd} e σ^2_{kd} são calculados de acordo com as expressões:

$$\mu_{kd} \leftarrow \mu_{kd} + \eta(c_{k \to d} - \mu_{kd}) \tag{3.10}$$

e

$$\sigma^2_{kd} \leftarrow \sigma^2_{kd} + \eta[(c_{k \to d} - \mu_{kd})^2 - \sigma^2_{kd}], \tag{3.11}$$

onde o $\eta \in [0, 1]$ representa o parâmetro heurístico que mede o peso de μ_{kd}, σ^2_{kd} e $c_{k \to d}$ sendo que o $c_{k \to d}$ representa o custo da ligação $k \to d$. Nota-se que a comparação dos custos neste modelo, é feita comparando os valores de μ_{kd} e σ^2_{kd} presentes no nó em causa, com os valores anunciados pela formiga que efectua a actualização .

são organizadas em função da largura de banda. As filas para pacotes com maior prioridade possuem maior largura de banda e com menor prioridade menor largura de banda.

- tabela de feromonas onde são armazenadas as intensidades de feromonas para os diversos destinos;
- tabela de encaminhamentos onde são armazenadas as informações de encaminhamentos para os diversos destinos da rede; e
- tamanho das filas de espera que são *buffers*[4] que servem para guardar os pacotes, temporariamente, antes de serem reencaminhados.

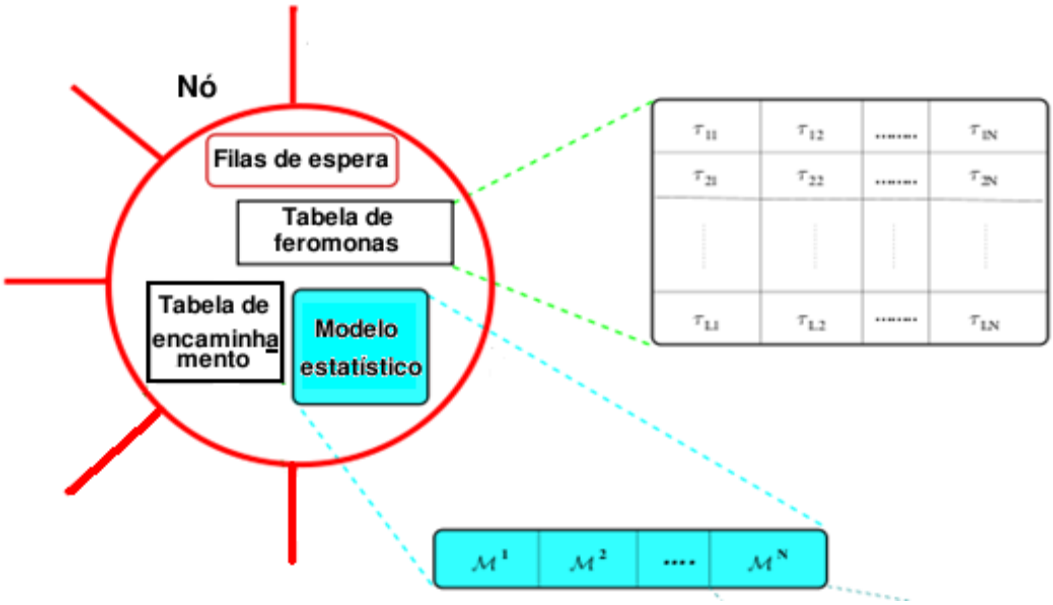


Figura 3.11: Estrutura de dados mantida nos nós que usam o algoritmo *AntNet*. O $M_k(k = 1, 2, 3, 4, ..., N)$ representa o modelo estatístico local da rede para os custos entre nós, possuindo uma estrutura de dados com os parâmetros da média (μ_{kd}) e da variância (σ^2_{kd}) dos custos para alcançar o nó d a partir do nó k. Temos também a tabela de feromonas, a tabela de encaminhamentos e as filas de espera.

Deste modo, em cada nó k, a formiga $B_{s\to d}$ faz a actualização do modelo estatístico, M_k, actualizando os parâmetros μ_{kd} e σ^2_{kd} (expressões (3.10) e (3.11)), de acordo com a informação $c_{k\to d}$ existente na memória $S_{k\to d}$.

[4] *buffer* é uma região de memória temporária utilizada para escrita e leitura de dados.

A actualização da tabela de feromonas, τ_k, é feita após a actualização do M_k, da seguinte forma:

- se o nó k possuir como vizinhos os nós m, f e n (Figura 3.12) e tendo a formiga $F_{s\to d}$ escolhido o nó f como *next hop*, quando esta tinha como destino o nó d, então a intensidade de feromonas τ_{fd}^k é actualizada de acordo com a expressão:

$$\tau_{fd}^k \leftarrow \tau_{fd}^k + r(1 - \tau_{fd}^k) \tag{3.12}$$

  onde τ_{fd}^k representa a intensidade de feromonas na aresta $(k,\ f)$, quando o destino é d e $r \in [0, 1]$ é um parâmetro que caracteriza o caminho ($r \equiv r(\tau, M)$).

- no caso dos nós vizinhos do nó k (Figura 3.12), que não tendo sido escolhidos pela formiga $F_{s\to d}$ (nós n e m) como *next hops*, quando esta tinha como destino o nó d, a actualização de feromonas é feita de acordo com a expressão:

$$\tau_{xd}^k \leftarrow \tau_{xd}^k - r\tau_{xd}^k, \qquad x \in \{n, m\} \tag{3.13}$$

Valores mais elevados do parâmetro r, favorecem o melhor caminho em detrimento dos outros, ou seja, o r é responsável pela diferença das intensidades de feromonas calculadas quando a formiga *Backward* chega ao nó k. As expressões que se seguem demonstram esse facto:

$$\begin{cases} \tau_{fd}^k \leftarrow \tau_{fd}^k + r(1 - \tau_{fd}^k) \equiv \tau_{fd}^k + r - r\tau_{fd}^k \equiv (1 - r)\tau_{fd}^k + r; \\ \qquad\qquad\qquad\qquad \tau_{xd}^k \leftarrow \tau_{xd}^k - r\tau_{xd}^k \equiv (1 - r)\tau_{xd}^k. \end{cases} \tag{3.14}$$

Por último, faz-se a actualização da tabela de encaminhamento, R^k. Para esta actualização calcula-se a probabilidade de cada decisão de encaminhamento referente a cada ligação de acordo com a expressão:

$$R_{nd}^k = \frac{\left(\tau_{nd}^k\right)^{\varepsilon}}{\sum_{i\in N_k} \tau_{id}^k}, \tag{3.15}$$

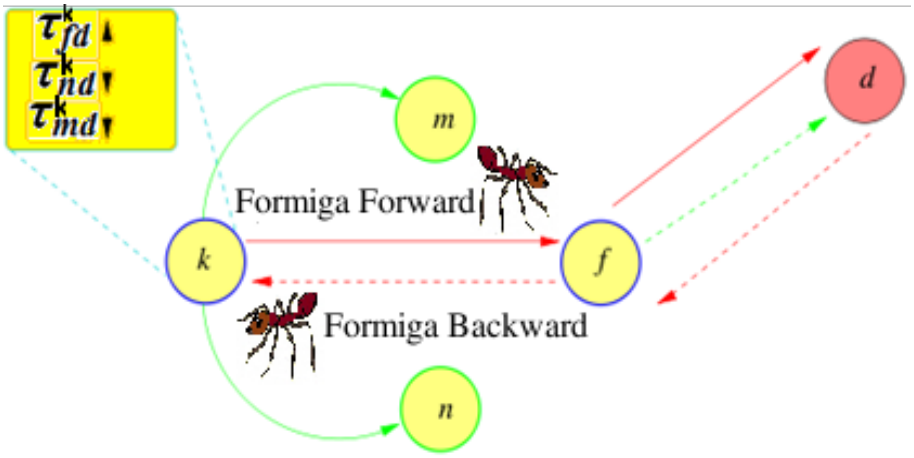


Figura 3.12: Processo de actualização da tabela de feromonas no nó k. τ^k_{fd}, τ^k_{nd} e τ^k_{md} são elementos da tabela de feromonas do nó k que representam as intensidades de feromonas existentes a partir do nó k aos nós f, n e m, respectivamente, com destino a d. Durante o processo de actualização de feromonas o τ^k_{fd} será aumentado visto que o nó f foi escolhido pela formiga $F_{s\to d}$ quando tinha como destino o nó d, e os valores de τ^k_{nd} e τ^k_{md} serão reduzidos de acordo com a expressão 3.13, satisfazendo a condição $\sum_{i=1}^{N_k} \tau^k_{id} = 1$ (i=f, n e d).

onde o ε define o nível de da aleatoriedade nas decisões de encaminhamento.

O objectivo deste cálculo é favorecer as opções de encaminhamento com elevada intensidade de feromonas (maior probabilidade) em detrimento das que possuem baixa intensidade de feromonas (menor probabilidade).

A expressão (3.15) é usada para o encaminhamento de pacotes da rede, onde R^k_{nd} representa os elementos da tabela de encaminhamento. Segundo as experiências realizadas por Di Caro [2004], o valor de $\varepsilon = 1.4$ revela-se ser bom porque assume um bom compromisso entre a necessidade de redução do risco de haver más decisões de encaminhamento e da possibilidade de que os pacotes sejam encaminhados por múltiplos caminhos.

O Algoritmo 3 apresenta os passos assenciais do processo que acabamos de descrever.

Algoritmo 3 Pseudocódigo do algoritmo *AntNet*

Require: Definem-se os parâmetros:
t - instante de tempo actual,
T - Tempo de simulação,
Δt - intervalo de tempo entre lançamento das formigas

```
1: foreach Nó de origem (s) do
2:   Inicia o modelo estatístico M
3:   Inicia a tabela de feromonas τ
4:   Inicia a tabela de encaminhamento R
5: end foreach
6: while não verificar a condição de paragem do
7:   foreach intervalo de tempo, Δt do
8:     Cria a formiga Forward ant
9:     d = CalculaDestino(tráfego da rede) {d-nó de destino, expressão (3.7)}
10:    EnviaFormigaForward (d, s)
11:    foreach FormigaForward do
12:      if d ≠ k then
13:        NextHop=CalculaNextHop(k, d, τ, FilasEspera) {k-nó intermédio, expressão (3.8)}
14:        EnviaFormigaForward (d, s, NextHop) {Envia a formiga Forward para o NextHop}
15:      else
16:        CriaBackwardAnt(d, s)
17:        Apaga(FormigaForwardAnt)
18:      end if
19:    end foreach
20:    foreach FormigaBackward do
21:      if s ≠ k then
22:        Actualiza M, τ e R {expressões (3.10), (3.11), (3.12), (3.12) e (3.13)}
23:        NextHop=EscolheNextHop(PilhaDeDados) {Escolhe o NextHop a partir da pilha de dados}
24:        EnviaFormigaBackward (s, d, NextHop) {Envia a formiga Backward para o NextHop}
25:      else
26:        Actualiza(M, τ, TabelaFeromonas)
27:        Actualiza M, τ e R {expressões (3.10), (3.11), (3.12), (3.12) e (3.13)}
28:      end if
29:    end foreach
30:  end foreach
31: end while
```

3.6 Sumário

Os algoritmos de encaminhamento tem um papel muito importante na transmissão de informação. Estes são usados nos protocolos de encaminhamentos por permitirem a criação e actualização das tabelas de encaminhamento nos nós de uma rede.

O algoritmo de encaminhamento *Distance-Vector* constrói as tabelas de encaminhamento num nó, com base nos custos que são comunicadas a esse nó pelos seus vizinhos.

A diferença entre o *Distance-Vector* e o *Link-State* reside no facto deste conhecer toda a topologia de rede, pois cada nó mede os custos aos vizinhos e envia essa informação em *flooding* (inundação) a todos os nós da rede. Assim, todos ficam a conhecer a rede na totalidade e podem assim calcular o Dijkstra; enquanto que no *Distance-Vector* cada nó troca informação com os vizinhos sobre o que cada um julga ser a sua distância a todos os nós da rede.

O algoritmo *AntNet* possui um principio de funcionamento bastante diferente dos anteriores, pois este é baseado no comportamento dos insectos sociais. A construção e actualização das tabelas de encaminhamento é feita sempre que as formigas conseguem alcançar um dado destino. Essa actualização baseia-se nos seguintes factores: quantidade de pacotes presentes nas filas de espera e intensidade de feromonas presentes nas tabelas de feromonas de um nó para um dado destino. A quantidade de feromonas reflecte a preferência do caminho. Estes factores tornam este, um algoritmo activo que reage às mudanças ao nível do tráfego da rede.

3.7 Conclusões

De uma forma geral podemos afirmar que a diferença entre os algoritmos clássicos, como o *Distance-Vector* e o *Link-State* para o *AntNet* reside no facto dos dois primeiros preocuparem-se apenas com os custos para o destino e serem estáticos, pois não reagem às mudanças de tráfego; enquanto que o último é um algoritmo dinâmico. Além do custo este preocupa-se também com a qualidade do caminho, que é ditada pelas condições da rede, como o congestionamento, ou seja, o congestionamento reduz a qualidade do caminho na medida em que quando a formiga não chega ao destino, devido ao descarte por congestionamento, perde-se a possibilidade de aumentar a intensidade de feromonas por parte dessa formiga.

No próximo capítulo serão apresentados as novas propostas de algoritmos ACO com pesquisa em profundidade, que visam explorar as soluções encontradas pelo algoritmo *AntNet*.

CAPÍTULO

4

Algoritmos ACO com pesquisa aleatória em profundidade

Com este capítulo pretendemos introduzir o conceito de pesquisa em profundidade associado aos algoritmos ACO. Este conceito visa dotar estes algoritmos de um comportamento mais exploratório, procurando dentre várias soluções existentes, aquelas que revelam-se melhores. Essas soluções são definidas como caminhos para um dado destino. Para tal serão estudados dois algoritmos diferentes entre si pelo seu princípio de funcionamento, mas semelhantes na forma como fazem a procura de soluções.

4.1 Visão geral

O constante aumento de aplicações multimédia nas redes IP, entre outras, conjugado com o crescimento do número de utilizadores de Internet, tem levado à criação de soluções de encaminhamento que proporcionem ao cliente um serviço cómodo, i.e., um bom serviço sem muitos atrasos ou perdas na transmissão de pacotes. Como já referimos, os algoritmos de encaminhamento são usados para determinar caminhos entre nós emissores e receptores. Nos algoritmos de encaminhamento clássicos, as tabelas de encaminhamento são actualizadas através da troca de mensagens entre os nós da rede e são baseadas em geral, nas distâncias. Depois que é escolhido um caminho, a diferenciação em classes de QoS é assegurada pela prioritização de nível IP.

Paralelamente a isso, os algoritmos de encaminhamento ACO, surgiram como uma possível alternativa para um encaminhamento de forma equilibrada, pois empreende uma dinâmica na rede de forma a proporcionar uma mesma satisfação para todos os utilizadores, oferecendo por igual os mesmos recursos [Kebria *e*t al. , 2009].

Nos algoritmos ACO podemos utilizar diversas características de acordo com o tipo de problema. Assim sendo, podemos considerar um caso de uma ou múltiplas colónias de formigas, uma ou múltiplas matrizes de feromonas e regras de actualização das matrizes de feromonas que permitem ou não às formigas actualizarem os rastos de feromonas de outras colónias [García *e*t al. , 2008]. Estes algoritmos, normalmente conhecidos como algoritmos de inteligência de enxame (Swarm Intelligence - SI), são conhecidos por produzir resultados com boas aproximações, precisando no entanto e em geral de bastante tempo e recursos ao nível de processamento, principalmente quando a aproximação obtida precisa de ser melhorada [Dorigo & Stutzle, 2004]. Todas estas circunstâncias fazem dos algoritmos híbridos uma possível solução para

a obtenção de resultados mais refinados com os métodos SI.

Nesse sentido, apresentamos neste capítulo dois algoritmos híbridos desenvolvidos para os problemas de encaminhamento em rede com fios. Estes algoritmos têm como base o método utilizado pelo algoritmo *AntNet* sendo que na próxima secção é apresentado o algoritmo ϵ-*DANTENet* que é uma adaptação do algoritmo ϵ-*DANTE* (ϵ-Depth ANT Explorer) [Cardoso *e*t al. , 2010]. Nessa secção far-se-á menção ao seu princípio de funcionamento desde a criação de uma formiga a partir de um nó origem até à forma como se faz a actualização das tabelas de encaminhamento por parte das mesmas.

Na secção 4.4 será apresentado o algoritmo *CR-DANTENet*, também para problemas de redes com fios, que surge na sequência de um melhoramento feito ao algoritmo ϵ-*DANTENet* com o objectivo de reduzir o tráfego gerado pelas formigas na rede. A base de funcionamento deste algoritmo será explicada nessa secção e faz-se ênfase às principais características que diferenciam este do seu predecessor. Nesta mesma secção, recorre-se a alguns exemplos para fazer comparação entre o *AntNet*, ϵ-*DANTENet* e o *CR-DANTENet*, de modo a avaliar as vantagens e desvantagens entre eles.

4.2 Algoritmo de encaminhamento ϵ-*DANTENet*

O algoritmo ϵ-*DANTENet* é um algoritmo híbrido que combina o método utilizado nos algoritmos *ACO* para o encaminhamento em redes com fios, com a pesquisa em profundidade. Trata-se de uma primeira adaptação, para problemas de redes de dados, do algoritmoϵ-*DANTE* apresentado por Cardoso *e*t al. [2010].
De forma a sabermos como funciona o ϵ-*DANTE*, vamos resumidamente descreve-lo nos seguintes passos:

- suponhamos que uma *Forward Ant*, pelo principio do AS, alcançou o nó de destino e cria a *Backward Ant*. Esta volta para o nó de origem actualizando as tabelas de feromonas presentes nos nós intermédios do caminho determinado pela *Forward Ant*, mas em sentido contrário;

- em cada nó intermédio por onde a *Backward Ant* passa, são criadas *Forward DANTE's* e enviadas para o mesmo destino da *Forward Ant*. Estas formigas têm como objectivo explorar novos caminhos, diferentes dos da *Forward Ant* original;

- quando a *Forward DANTE* alcança o destino, compara-se a solução (ou caminho) encontrada por esta formiga com a *Forward Ant*. Se esta nova solução não for boa, esta formiga é descartada. Caso contrário cria-se a *Backward Ant*. Esta volta para o nó de origem da *Forward Ant* original actualizando as tabelas de feromonas presentes nos nós intermédios.

O algoritmo 4 apresenta os passos processo ϵ-*DANTE*.

Algoritmo 4 Pseudocódigo do algoritmo ϵ-*DANTE*

Ensure: soluções aproximadas, P {possivelmente melhoradas}
1: Inicializa os rastos de feromonas
2: **while** não se verifica o critério de paragem **do**
3: **forall** ForwardAnts **do**
4: Usa o rasto de feromonas para construir uma nova solução, S
5: Verifica a qualidade S comparando com as soluções em P
6: Dependendo da qualidade de S, faz pesquisa em profundidade baseada na solução
7: **end forall**
8: Actualiza a tabela de feromonas
9: **end while**

O método do algoritmo ACO utilizado para o efeito de combinação com a pesquisa em profundidade foi o do *AntNet* apresentado no Capítulo 3. O algoritmo

ϵ-*DANTENet* foi desenvolvido com o objectivo de explorar as soluções obtidas pelo algoritmo *AntNet*, ou seja, todas as soluções consideradas boas em relação às suas antecessoras, são utilizadas para fazer a actualização das tabelas de feromonas, numa espécie de pesquisa em profundidade. Para uma melhor descrição deste algoritmo, começaremos por definir as seguintes formigas:

- ***formiga Forward***, representada por $F_{s\to d}^{Ant}$, viaja do nó de origem s até ao nó d, com objectivo de encontrar um caminho para o envio de pacotes entre estes dois nós.

- ***formiga Backward***, representada por $B_{s\to d}^{Ant}$, criada com objectivo de actualizar as tabelas de encaminhamento baseando-se nas informações recolhidas pela formiga $F_{s\to d}^{Ant}$ em todos os nós por onde esta passou, bem como criar a formiga $F_{k\to d}^{DANTE}$ (definida a seguir) nos nós intermédios;

- ***formiga Forward DANTE***, representada por $F_{k\to d}^{DANTE}$, esta formiga é gerada pela formiga $B_{s\to d}^{Ant}$ nos nós onde esta última passa e tem como objectivo de fazer a procura em profundidade, criando um comportamento exploratório. Viaja desde o nó intermédio k, onde é criada, até ao nó d, procurando caminhos alternativos diferente do encontrado pela formiga $F_{s\to d}^{Ant}$ (desde o nó k até ao nó d);

- ***formiga Backward DANTE***, representada $B_{s\to d}^{DANTE}$, é gerada quando a formiga $F_{k\to d}^{DANTE}$ chega ao nó de destino. Ela tem como objectivo utilizar as informações recolhidas pela formiga $F_{k\to d}^{DANTE}$ em todos os nós por onde passou, para actualizar as tabelas de encaminhamento no percurso $s \to d$ ($s \to k$ e $k \to d$).

A diferença entre as formigas $F_{s\to d}^{Ant}$ e $F_{k\to d}^{DANTE}$ está no facto desta última possuir uma estrutura da memória que inclui o custo total da primeira. No entanto a $B_{s\to d}^{DANTE}$ possui um estrutura de memória igual a $B_{s\to d}^{Ant}$.

Assim, e tal como no algoritmo *AntNet*, são enviadas formigas $F_{s\to d}^{Ant}$ a partir do nó de origem s para o nó de destino d (escolhido no nó s de acordo com a probabilidade dada pela expressão (3.7)), em intervalos de tempo, Δt, com o objectivo de melhorar a informação de encaminhamento ou de descobrir novos caminhos.

A probabilidade desta formiga ser encaminhada para qualquer um dos seus vizinhos n, a partir dos nós intermédios k é dada pela expressão (3.8); e durante o seu percurso ela armazena na sua memória $S_{s\to d}$ o endereço dos nós que visitou (v_i) e o custo das ligações ($c_{v_i\to v_{i+1}}$) (Figura 4.1).

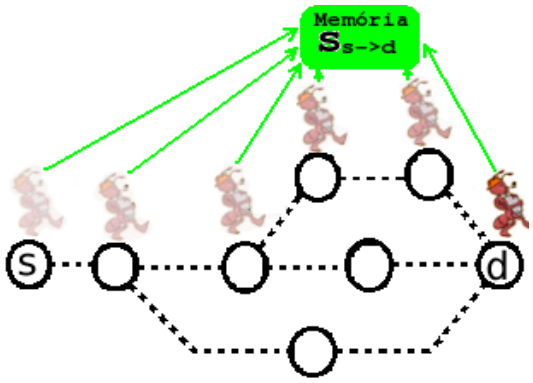


Figura 4.1: Processo de armazenamento dos nós e custos na memória $S_{s\to d}$, a partir do nó de origem s até o nó de destino d.

No nó de destino d, a formiga $F_{s\to d}^{Ant}$ é substituída pela formiga $B_{s\to d}^{Ant}$, criada nesse mesmo nó, transferindo toda a sua a sua memória para esta e, sendo destruída de seguida.

De volta ao nó s (em sentido contrário), a formiga $B_{s\to d}^{Ant}$, que possui prioridade máxima, passa por todos os nós visitados pela formiga $F_{s\to d}^{Ant}$, actualizando as tabelas

de feromonas e de encaminhamentos. Em cada nó intermédio k a formiga $B_{s\to d}^{Ant}$ cria a $F_{k\to d}^{DANTE}$ e copia parcialmente para esta, o vector dos nós e custos da memória, i.e., copia os nós $v_{v_1\to v_k}$ e os custos $c_{v_1\to v_k}$. É também armazenada, na memória da formiga criada, o custo total da formiga $F_{s\to d}^{Ant}$ (transportado pela $B_{s\to d}^{Ant}$).

A condição para a criação da formiga $F_{k\to d}^{DANTE}$ em k é o seu *next hop*, calculado pela expressão (3.8), ser diferente dos nós antecessor $(k-1)$ e predecessor $(k+1)$ da formiga $F_{s\to d}^{Ant}$.

Esta condição garante uma exploração por novos caminhos (ou soluções) por parte da formiga $F_{k\to d}^{DANTE}$ a partir do nó k. Caso seja verificada a referida condição, a formiga $F_{k\to d}^{DANTE}$ é enviada para o mesmo destino d, comportando-se como a formiga $F_{s\to d}^{Ant}$. O encaminhamento em cada nó intermédio é feito da mesma forma que a formiga $F_{s\to d}^{Ant}$ (Figura 4.1), e com o recurso a expressão (3.8).

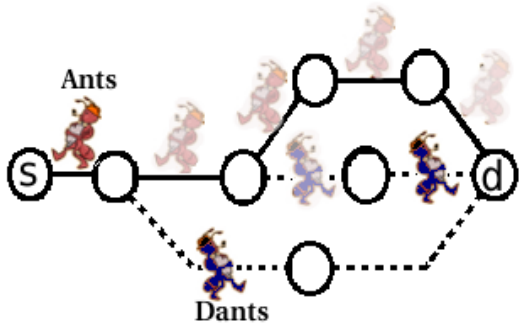


Figura 4.2: Criação das formigas $F_{k\to d}^{DANTE}$ nos nós por onde a formiga $B_{s\to d}^{Ant}$ passa. Para a criação desta formiga o *next hop* calculado pela expressão (3.8), deve ser diferente dos nós $k-1$ e $k+1$ do vector que guarda os nós visitados pela $F_{s\to d}^{Ant}$.

Quando a $F_{k\to d}^{DANTE}$ chega ao nó de destino cria a formiga $B_{s\to d}^{DANTE}$ copiando apenas as informações dos vectores dos nós e dos custos para a memória desta, caso

se verifique a condição:

$$Atraso_{F_{k\to d}^{DANTE}} < \epsilon \cdot Atraso_{F_{k\to d}^{Ant}}, \tag{4.1}$$

onde $\epsilon \in [0, 1]$.

Criada a formiga $B_{s\to d}^{DANTE}$, esta é enviada para o nó s pelo caminho de d a k e k a s, ou seja, por uma parte do caminho, de k à d percorrido pela formiga $F_{k\to d}^{DANTE}$ e por outra parte de s à k percorrido pela formiga $F_{s\to d}^{Ant}$, mas em sentido contrário (Figura 4.3).

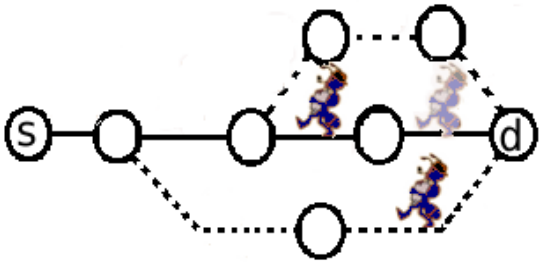


Figura 4.3: Formigas $B_{s\to d}^{DANTE}$ enviadas a partir do nó d para o nó s, actualizando as tabelas de encaminhamento dos nós intermédios k.

Esta formiga é encaminhada com a mesma prioridade que as $B_{s\to d}^{Ant}$, actualizando as tabelas de feromonas e de encaminhamento em cada nó intermédio, k, e no nó de origem, s.

Esta actualização é feita comparando os custos presentes na memória $S_{s\to d}$ com os custos existentes na tabela de feromonas do nó à actualizar, da mesma forma que no algoritmo *AntNet* descrito no capítulo 3. Este processo permite que as melhores soluções de encaminhamento, sejam inseridas nas tabelas de encaminhamento com mais rapidez. Ou seja, caso seja encontrada uma boa solução por parte da formiga $F_{k\to d}^{DANTE}$, esta solução é usada para efectuar a actualização das tabelas de feromonas

e de encaminhamento, antes de serem lançadas novas formigas na rede a partir dos nós de origem s. Este processo permite aumentar o rasto de feromonas nesse caminho minimizando o tempo para que um dado caminho, considerado bom, seja escolhido para o encaminhamento de pacotes.

A estrutura de dados nos nós que utilizam este algoritmo é mantida igual à do algoritmo *AntNet*.

Devido a sua filosofia de funcionamento, o algoritmo ϵ-*DANTENet* gera mais tráfego na rede se comparado com o *AntNet*, devido as formigas $F_{k\to d}^{DANTE}$ e $B_{s\to d}^{DANTE}$.

O processo ϵ-*DANTENet* encontra-se resumido no Algoritmo 5 (continuando no 6

4.3 Implementaçao do algoritmo ACO no Network Simulator - versão 2.34 (*ns-2*)

Com o intuito de testarmos os algoritmos ACO propostos nesta dissertação, foi feita a sua implementação no *ns-2* (Apêndice A). Como base foi utilizada a implementação do *AntNet* desenvolvida por Lima [2009]. Esta implementação gera as tabelas de feromonas, não procedendo no entanto à actualização das tabelas de encaminhamento, num procedimento estático, onde todo o tráfego não pertecente ao algoritmo *AntNet* é encaminhado segundo os parâmetros do algoritmo utilizado por omissão no *ns-2* (*Link-State*). Ou seja, o tráfego gerado pelos nós não é encaminhado de acordo com as tabelas de feromonas geradas pelo algoritmos *AntNet*.

Na realidade, que fosse do conhecimento do autor, não existia até ao momento nenhuma implementação completa no *ns-2* que permitisse dinamicamente responder às soluções de encaminhamento geradas pelos algoritmos ACO.

Em suma, a implementação base do *AntNet* no *ns-2* possuía métodos que per-

mitiam, entre outros:

- criar e lançar as formigas *Forward* para um dado destino;
- reencaminhar nos nós intermédios as formigas *Forward* de acordo com os valores de probabilidade escolhidas pseudo-aleatóriamente a partir da expressão (3.8);
- criar as formigas *Backward* a partir das formigas *Forward*;
- criação e actualização das tabelas de feromonas.

Na versão modificada foi introduzida a capacidade de criar e actualizar as tabelas de encaminhamento a partir das tabelas de feromonas, o que permitiu verificar a dinâmica resultante da interacção entre o algoritmo de encaminhamento e os fluxos de tráfego que se introduziam ao longo das simulações.

Com o intuito de facultar o acesso aos recursos criados, os mesmos estão disponíveis em Mapisse [2010], com uma explicação dos passos essenciais para a sua utilização. O objectivo de tais recursos é de permitir futuros desenvolvimentos por outros investigadores dos algoritmos testados.

4.4 Algoritmo de encaminhamento *CR-DANTENet*

Devido as formigas $F_{k\to d}^{DANTE}$ e $B_{s\to d}^{DANTE}$ do algoritmo ϵ-*DANTENet*, um número elevado de pacotes circula pela rede. A demonstração disso é feita considerando a rede com 7 nós, da Figura 4.4, onde temos os algoritmos *AntNet* e ϵ-*DANTENet* que criam e actualizam as tabelas de feromonas e de encaminhamentos. A simulação foi feita no *ns-2* durante 50 segundos. Os restantes dados desta simulação, tais como nós emissor e receptor, tipo de pacotes transmitidos bem como o ritmo e tempos de transmissão podem ser encontrados na Tabela 4.1.

Tabela 4.1: Dados inerentes a simulação da rede da Figura 4.4, com os algoritmos *AntNet* e ϵ-*DANTENet*.

Emissor	Receptor	Tipo de pacote	Ritmo (kbps)	Início da transmissão(s)	Fim da Transmissão(s)
0	6	UDP	900	0.0	50.0
0	6	UDP	900	1.0	50.0
3	6	UDP	900	0.75	50.0
6	1	UDP	900	1.0	30.0

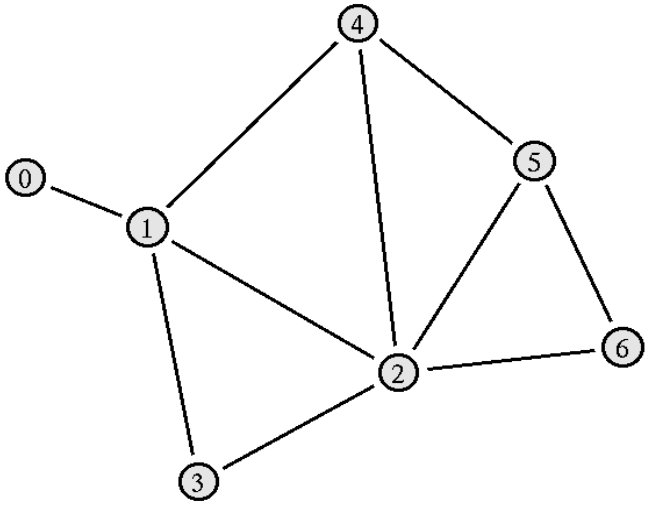


Figura 4.4: Rede com 7 nós utilizada para analisar o comportamento dos algoritmos *AntNet* e ϵ-*DANTENet*.

A escolha deste tipo de pacote relaciona-se com o facto do mesmo manter o seu ritmo de transmissão independentemente das condições impostas pela rede, tal como o tráfego; o que requer uma maior adaptabilidade dos algoritmos de encaminhamento de modo a reduzir ou mesmo evitar a perda de pacotes. Os resultados dessa simulação podem ser visualizados na Figura 4.5. Nota-se que o número de formigas geradas na rede em cada segundo, usando o algoritmo ϵ-*DANTENet* é cerca do dobro em comparação com o *AntNet*, apresentando uma pequena diminuição nas perdas de pacotes na rede (Figura 4.6). Nota-se que nos primeiros instantes, o algoritmo *AntNet*

apresenta uma subida galopante de perdas de pacotes na rede. Segundo os dados de transmissão ilustrados na Tabela 4.1, nesses instantes é injectado o tráfego UDP na rede, e a resposta do algoritmo *AntNet* é mais lenta se comparada com a do algoritmo ϵ-*DANTENet*. Próximo do segundo 50 nota-se um aumento das perdas deste algoritmo, que segundo a análise do ficheiro NAM (feita em conformidade com a descrição constante no Apêndice A), provavelmente possa ser devido a sequência de más decisões de encaminhamento dos pacotes na rede.

Note-se que se trata de uma simulação que tende a ser genérica, podendo haver variações de acordo a semente escolhida pelo gerador aleatório de números.

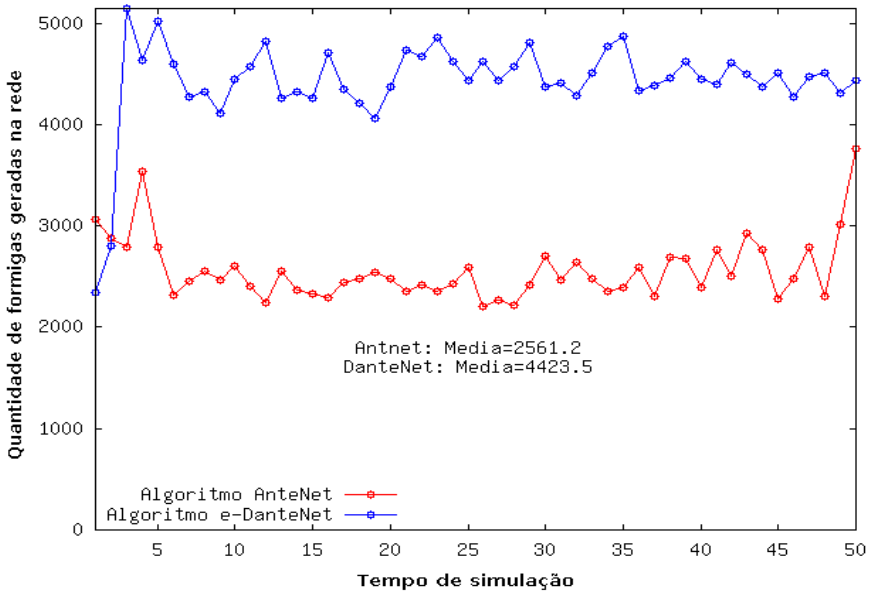


Figura 4.5: Quantidade de formigas geradas na rede da Figura 4.4, durante cada segundo da simulação feita, sendo que a rede utiliza os algoritmos *AntNet* e ϵ-*DANTENet*, e transmite pacotes UDP.

Neste sentido, o algoritmo *CR-DANTENet* que propomos, foi desenvolvido com o objectivo de melhorar o ϵ-*DANTENet*, através da diminuição do número de formigas

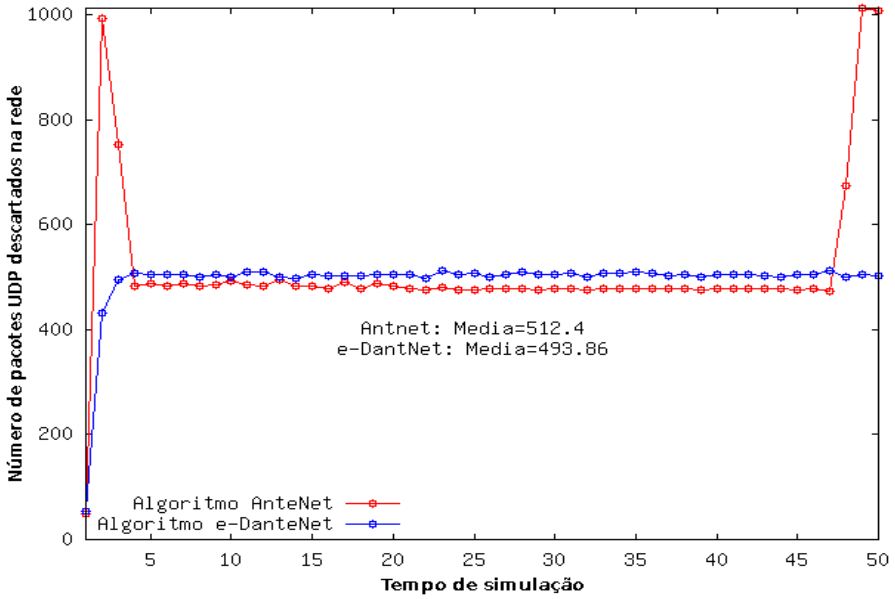


Figura 4.6: Perdas de pacotes na rede da Figura 4.4, durante cada segundo da simulação feita, com os algoritmos *AntNet* e ϵ-*DANTENet*.

$F_{k\to d}^{DANTE}$.

Assim, as partes comuns entre os algoritmos ϵ-*DANTENet* e o *CR-DANTENet* são:

- a escolha do nó de destino d;
- o encaminhamento da formiga $F_{s\to d}^{Ant}$, nos nós intermédios e no nó de origem e a informação que estas armazenam;
- a criação da formiga $B_{s\to d}^{Ant}$, pela $F_{s\to d}^{Ant}$ no destino d e a transferência da memória $S_{s\to d}$;
- a actualização das tabelas de feromonas pelas formigas $B_{s\to d}^{Ant}$; e
- a estrutura de dados mantida nos nós que usam este algoritmo de encaminhamento;

- a estrutura da memória é igual para todas as formigas, i.e., é organizada numa matriz com dois vectores: um que armazena os endereços dos nós e o outro que armazena os custos das ligações entre esses mesmos nós.

As diferenças entre o algoritmo ϵ-*DANTENet* e o *CR-DANTENet* são:

- em cada nó intermédio k, a formiga $B_{s\to d}^{Ant}$ cria a formiga $F_{k\to d}^{DANTE}$ e copia a informação da memória a partir do nó s ao k, mediante a condição do nó k apresentar a fila de espera cheia, de k para o *next hop* que permite alcançar o nó d, e o mesmo *next hop* for diferente do nó $k+1$.

- Quando a formiga $F_{k\to d}^{DANTE}$ chega ao nó de destino cria a formiga $B_{s\to d}^{DANTE}$ copiando para esta a sua memória.

A formiga $B_{s\to d}^{DANTE}$ é enviada para o nó s pelo caminho $d \to k$ e $k \to s$. O tipo de prioridade que esta formiga tem é o mesma que do algoritmo ϵ-*DANTENet* de modo a efectuar a actualização das tabelas de encaminhamento com a maior rapidez nos nós por onde passa.

Considerando as mesmas condições de simulação do exemplo prévio, o número de formigas geradas por este algoritmo em comparação com o *AntNet* e o ϵ-*DANTENet*, podem ser visualizados no gráfico da Figura 4.7 e as perdas dos UDP podem ser visualizadas na Figura 4.8.

O processo *CR-DANTENet* está resumido no algoritmo 7 (continua no 8).

Comparando o algoritmo ϵ-*DANTENet* com o *CR-DANTENet* notamos que o número de formigas geradas quando se utiliza o algoritmo *CR-DANTENet* foi reduzido em cerca de 28% (Figura 4.7) registando um ligeiro aumento das perdas UDP (cerca de 7%).

As perdas de pacotes dos algoritmos ϵ-*DANTENet* e *CR-DANTENet* relativamente ao *AntNet* representam uma diminuição em média de cerca de $3,8\%$ e um

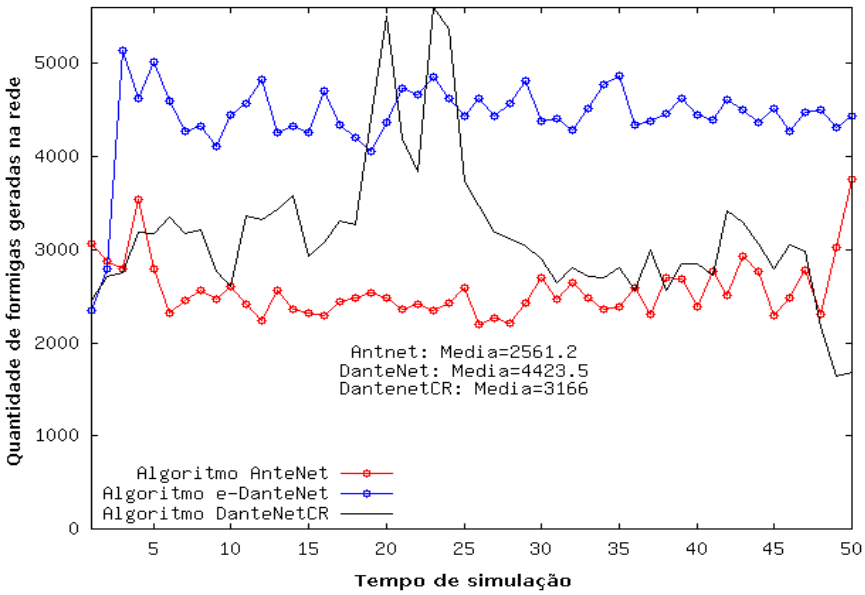


Figura 4.7: Quantidade de formigas geradas na rede da Figura 4.4, durante cada segundo da simulação feita, sendo que a rede utiliza os algoritmos *AntNet*, ϵ-*DANTENet* e *CR-DANTENet*, e transmite pacotes UDP.

aumento de $3,9\%$, respectivamente. Aparentemente, o aumento das perdas dos pacotes, foi originado por uma sequência de más decisões por parte das formigas $F_{k\to d}^{DANTE}$. Na realidade, para o caso em analise, não foram verificadas grandes melhorias nos resultados pelo que iremos propôr no Capítulo 5, novas soluções para o encaminhamento das formigas utilizando a largura de banda disponível.

4.5 Sumário

A pesquisa em profundidade utilizando algoritmos ACO foi introduzida visando melhorar as soluções encontradas pelo algoritmo *AntNet*. Para tal, foram criados dois algoritmos, nomeadamente o ϵ-*DANTENet* e o *CR-DANTENet*.

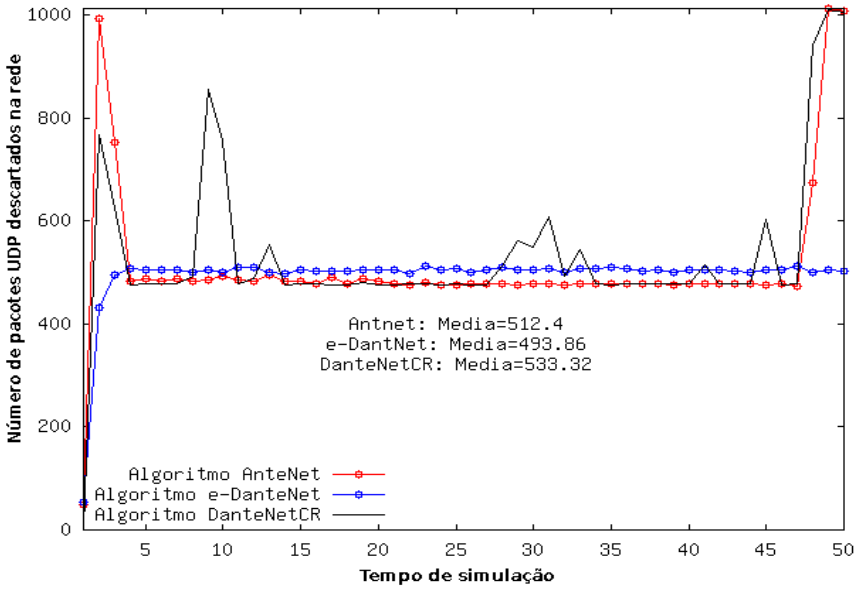


Figura 4.8: Perdas de pacotes na rede da Figura 4.4, durante cada segundo da simulação feita, com os algoritmos *AntNet*, ϵ-*DANTENet* e *CR-DANTENet*.

O ϵ-*DANTENet* foi uma adaptação para redes de dados do algoritmo ϵ-DANTE. Este algoritmo apresenta um novo tipo de formesta formiga é encaminhada iga artificial que é responsável por implementar a pesquisa em profundidade, limitada no número de ramos. Uma das formigas, designada por *Forward* DANTE, F^{DANTE}, é criada a partir das formigas *Backwards*, nos nós intermédios entre a origem e o destino. São criadas para descobrir novos caminhos para o destino, a partir do nó em questão.

No destino a formiga *Forward* DANTE cria a formiga *Backward* apenas quando se verificar a condição 4.1. Devido ao surgimento destas novas formigas, há um aumento do tráfego na rede. O *CR-DANTENet* aparece como uma tentativa para a diminuir o número de formigas na rede. Neste as formigas DANTE's são criadas quando existe uma situação de congestionamento numa dada ligação para o qual o

nó emissor pretende enviar os pacotes. Este comportamento limita a busca em profundidade, mas não inunda a rede de pacotes, que podem gerar perdas consideráveis. Um outro aspecto que diferencia estes dois algoritmos esta relacionado com o seu comportamento exploratório. Enquanto que no ϵ-*DANTENet* as formigas DANTE's, quando criadas, são enviadas apenas para frente, no *CR-DANTENet* estas podem ser enviadas para trás, para o nó $k-1$, mas todas com o propósito de atingir o destino.

4.6 Conclusão

Os algoritmos ϵ-*DANTENet* e *CR-DANTENet* juntam o método utilizado no *AntNet* com a pesquisa em profundidade. O objectivo da solução adoptada é o de melhorar as soluções obtidas pelo método utilizado no *AntNet*.

A solução adoptada pelo ϵ-*DANTENet*, descrita na Secção 4.2, introduziu algumas melhorias em relação ao *AntNet*, conforme pode ser verificado no gráfico da Figura 4.8, provocando simultâneamente um aumento da quantidade de formigas na rede, conforme ilustrado na Figura 4.7.

Como forma de minimizar o número de formigas geradas pelo ϵ-*DANTENet*, mantendo o mesmo nível de perda de pacotes na rede, foi criado o algoritmo *CR-DANTENet*. O gráfico da Figura 4.8 mostra uma redução na quantidade de formigas geradas por este algoritmo; mas, provavelmente, a sequência de más decisões no encaminhamento de pacotes fez com que, para o caso em analise, aumentasse o número de pacotes perdidos na rede, facto que se pode constatar no gráfico da Figura 4.7.

A melhoria em relação ao *AntNet*, no que diz respeito aos pacotes perdidos é de cerca de $3,8\%$ para 70% de formigas a mais geradas pelo ϵ-*DANTENet*. Paralelamente a este, o *CR-DANTENet* registou um aumento em cerca de $3,9\%$ no número de pacotes perdidos em relação ao *AntNet* apesar de reduzir o número de formigas

geradas na rede pelo ϵ-*DANTENet*.

Este exemplo, meramente ilustrativo e que não pode ser generalizado, serviu de base para o seguimento dos nossos estudos, explorando outras caracteristicas presentes numa rede de dados com fios.

Da mesma forma, os resultados alcançados levou-nos a procurar novas soluções para uma redução considerável do número de pacotes perdidos em relação ao *AntNet*. Assim, no próximo capítulo iremos apresentar uma nova proposta para os algoritmos estudados, através da introdução da largura de banda na escolha do *next hop*. O objectivo desta introdução é de tirar proveito da largura de banda disponível nas diversas ligações entre os nós.

Algoritmo 5 Pseudocódigo do algoritmo *ϵ-DANTENet*

Require: Definem-se os parâmetros:
t - instante de tempo actual,
T - Tempo de simulação,
Δt - intervalo de tempo entre lançamento das formigas

```
1: foreach Nó de origem (s) do
Require: cria-se:
    o modelo estatístico M
    a tabela de feromonas τ
    a tabela de encaminhamento
2:  while não verificar a condição de paragem do
3:    foreach intervalo de tempo, Δt do
4:      Cria a formiga Forward Ant
5:      Escolhe aleatóriamente o nó de destino d, em função do tráfego da rede
6:      EnviaFormigaForwardAnt (d, s)
7:      foreach FormigaForwardAnt do
8:        if d ≠ k then
9:          NextHop=Função(k, d, τ, FilasEspera) {através da expressão (3.8),
            onde k é o nó intermédio}
10:         EnviaFormigaForwardAnt (d, s) {Envia a formiga Forward Ant para
            o NextHop}
11:       else
12:         CriaBackwardAnt(d, s, PilhaDeDados)
13:         Apaga(FormigaForwardAnt)
14:       end if
15:     end foreach
16:     foreach FormigaBackwardAnt do
17:       if s ≠ k then
18:         Actualiza(M, τ, TabelaFeromonas)
19:         CriaForwardDANTE(d, s, PilhaDeDados, AtrasoForwardAnt)
20:         NextHop=Função(PilhaDeDados) {Escolhe o NextHop a partir da pi-
            lha de dados}
21:         EnviaFormigaBackwardAnt (s, d) {Envia a formiga Backward Ant
            para o NextHop}
22:       else
23:         Actualiza(M, τ, TabelaFeromonas)
24:         Apaga(FormigaForwardAnt) {Apaga a Formiga Backward Ant}
25:       end if
26:     end foreach
27:   end foreach
28:  end while
29: end foreach
```

Algoritmo 6 Pseudocódigo do algoritmo ϵ-*DANTENet* (cont.)

```
1: foreach FormigaForwardDANTE do
2:   if d ≠ k then
3:     NextHopDANTE=Função(k, d, τ, FilasEspera)
4:     if o next hop da DANTE não for o mesmo que fora escolhido pela Forward
       Ant then
5:       EnviaFormigaForwardDANTE (d, s)
6:     end if
7:   else
8:     if AtrasoForwardDANTE < ε· AtrasoForwardAnt then
9:       CriaBackwardDANTE(d, s, PilhaDeDados)
10:    end if
11:    Apaga(FormigaForwardDANTE)
12:  end if
13: end foreach
14: foreach FormigaBackwardDANTE do
15:   if s ≠ k then
16:     Actualiza(M, τ, TabelaFeromonas)
17:     NextHop=Função(PilhaDeDados)
18:     EnviaFormigaBackwardDANTE (s, d)
19:   else
20:     Actualiza(M, τ, TabelaFeromonas)
21:     Apaga(FormigaBackwardDANTE)
22:   end if
23: end foreach
```

Algoritmo 7 Pseudocódigo do algoritmo *CR-DANTENet*

Require: Definem-se os parâmetros:
t - instante de tempo actual,
T - Tempo de simulação,
Δt - intervalo de tempo entre lançamento das formigas
1: **foreach** Nó de origem (s) **do**
Require: cria-se:
o modelo estatístico M
a tabela de feromonas τ
a tabela de encaminhamento
2: **while** não verificar a condição de paragem **do**
3: **foreach** intervalo de tempo, Δt **do**
4: Cria a formiga *Forward Ant*
5: Escolhe aleatóriamente o nó de destino d, em função do tráfego da rede
6: EnviaFormigaForwardAnt (d, s)
7: **foreach** FormigaForwardAnt **do**
8: **if** d $\neq$ k **then**
9: NextHop=Função(k, d, τ, FilasEspera) {através da expressão (), onde k é o nó intermédio}
10: EnviaFormigaForwardAnt (d, s) {Envia a formiga Forward Ant para o NextHop}
11: **else**
12: CriaBackwardAnt(d, s, PilhaDeDados)
13: Apaga(FormigaForwardAnt)
14: **end if**
15: **end foreach**
16: **foreach** FormigaBackwardAnt **do**
17: **if** s $\neq$ k **then**
18: Actualiza(M, τ, TabelaFeromonas)
19: **if** Fila de espera, para o nó de destino, estiver cheia **then**
20: CriaForwardDANTE(d, s, PilhaDeDados)
21: **end if**
22: CriaForwardDANTE(d, s, PilhaDeDados, AtrasoForwardAnt)
23: NextHop=Função(PilhaDeDados) {Escolhe o NextHop a partir da pilha de dados}
24: EnviaFormigaBackwardAnt (s, d) {Envia a formiga Backward Ant para o NextHop}
25: **else**
26: Actualiza(M, τ, TabelaFeromonas)
27: Apaga(FormigaForwardAnt) {Apaga a *Formiga Backward Ant*}
28: **end if**
29: **end foreach**
30: **end foreach**
31: **end while**
32: **end foreach**

Algoritmo 8 Pseudocódigo do algoritmo *CR-DANTENet* (cont.)

```
1: foreach FormigaForwardDANTE do
2:   if d ≠ k then
3:     NextHopDANTE=Função(k, d, τ, FilasEspera)
4:     if o next hop da DANTE não for o mesmo que fora escolhido pela Forward
       Ant then
5:       EnviaFormigaForwardDANTE (d, s)
6:     end if
7:   else
8:     CriaBackwardDANTE(d, s, PilhaDeDados)
9:     Apaga(FormigaForwardDANTE)
10:  end if
11: end foreach
12: foreach FormigaBackwardDANTE do
13:  if s ≠ k then
14:    Actualiza(M, τ, TabelaFeromonas)
15:    NextHop=Função(PilhaDeDados)
16:    EnviaFormigaBackwardDANTE (s, d)
17:  else
18:    Actualiza(M, τ, TabelaFeromonas)
19:    Apaga(FormigaBackwardDANTE)
20:  end if
21: end foreach
```

CAPÍTULO

5

Introdução do factor de largura de banda nos algoritmos ACO

Neste capítulo pretendemos estudar um novo conceito que tem como objectivo explorar os recursos disponibilizados por uma rede de dados, para a transmissão de pacotes, de forma a melhorar as condições de encaminhamento. Nesse sentido testa-se a solução de utilização da largura de banda entre as ligações. Este conceito é aplicado em todos os algoritmos ACO até aqui estudados, nesta dissertação.

5.1 Visão geral

Os algoritmos ACO são conhecidos pela forma como emulam o comportamento dos insectos sociais. Estes algoritmos foram desenvolvidos para que os pacotes imitem o processo de procura de alimentos das formigas reais, deixando feromonas pelo caminho que percorrem. Estes pacotes (formigas artificiais) são utilizados para encontrar um melhor caminho entre os nós de origem e destino.

O *AntNet* (apresentado no Capítulo 3), foi o primeiro algoritmo ACO criado para resolver problemas de encaminhamento em redes de dados, mais bem sucedido e apresenta-se como um algoritmo *hop-by-hop* baseado na propriedade de *estigmergia*[1] levadas a cabo pelas formigas [Kebria *e*t al. , 2009]. Este algoritmo tem duas partes principais: a exploração para a descoberta dos melhores caminhos (feita com o recurso a expressão (3.8)) e a actualização das tabelas de feromonas e de encaminhamentos dos nós.

Neste capítulo introduzimos um novo factor de exploração, com o objectivo de criar um equilíbrio entre as filas de espera juntamente com as feromonas e a largura de banda disponível entre as diversas ligações. Este factor é introduzido em três algoritmos ACO dando origem à outros três, que nos propomos a apresentar, de forma a melhorarem as soluções apresentadas pelos algoritmos anteriores (estudados nos Capitulos 3 e 4)

Desta forma na Secção 5.2 será apresentado o pressuposto para a introdução do factor largura de banda disponível e o princípio de funcionamento dos algoritmos que usam este factor. Nas Secções 5.3, 5.4 e 5.5 serão apresentadas as versões dos algoritmos *AntNet*, ϵ*-DANTENet* e *CR-DANTENet*, respectivamente, que utilizam a largura de banda disponível entre as ligações para efectuarem o encaminhamento das

[1] Definido no Capitulo 3 como comunicação indirecta entre os elementos de um sistema através da modificação do ambiente.

formigas *Forwards* nos nós intermédios.

5.2 Pressupostos para a utilização da largura da banda

Os algoritmos ACO apresentados, destinados a redes com fios, apresentam a desvantagem de escolherem um caminho para o encaminhamento das formigas baseados apenas no tamanho das filas de espera e na intensidade de feromonas existentes. A proposta de utilizar a largura de banda surgiu da necessidade de aumentar a probabilidade de uma formiga ser encaminhada por uma ligação que possui uma grande parte da sua largura de banda disponível, reduzindo deste modo a probabilidade de se criar-se congestionamento em outras ligações (ligações com maior ocupação de largura de banda) e consequente redução das perdas de pacotes.

O estudo apresentado a seguir, serviu de inspiração para a introdução deste parâmetro nos algoritmos ACO estudados, visto que notou-se haver diferenças entre a condição de encaminhamento com e sem a largura de banda disponível. Assim sendo, vamos supor que num determinado instante de tempo, um determinado pacote UDP está a ser transmitido com o ritmo de $1Mbps$ conforme ilustrado na rede da Figura 5.1. Cada cor da fila de espera do nó 1 representa um pacote que se pretende enviar para o nó com a mesma cor. Na tabela de feromonas temos as intensidades de feromonas entre este nó e os seus vizinhos correspondente à um destino específico (representado com a côr do nó de destino em causa).

Considerando que a formiga *Forward* foi gerada no nó de origem 0 e encontra-se no nó 1, pretendendo encontrar um caminho para o nó de destino 6. Recorrendo à expressão (3.8) calculamos a probabilidade desta formiga ser encaminhada por qualquer um dos seus vizinhos, determinando em primeiro plano o factor heurístico l_n (expressões em (5.1)) relactivo aos seus vizinhos, de acordo com a expressão (3.9):

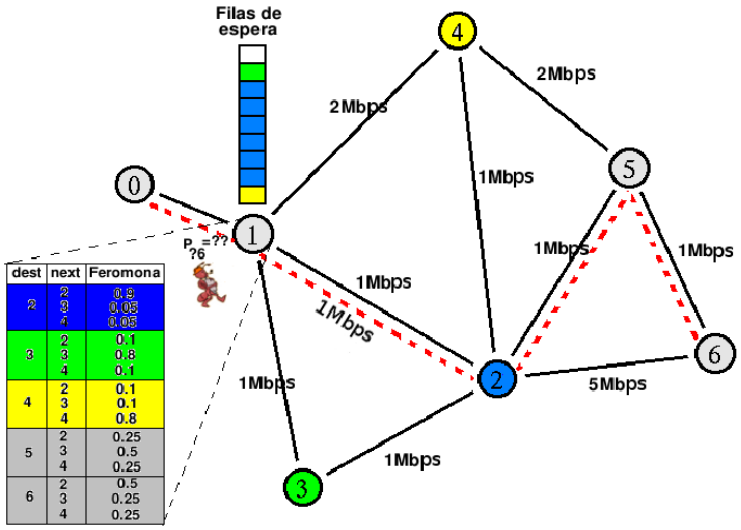


Figura 5.1: Transmissão de um pacote UDP a um ritmo de $1Mbps$ entre os nós 0 e 6 seguindo o caminho $0-1-2-5-6$. Nesta rede encontramos a fila de espera do nó 1 em que cada cor representa o pacote a ser enviado para o nó correspondente a essa cor.

$$\begin{cases} l_2 = 1 - \frac{q_2}{\sum_{n'=1}^{4} q_{n'}} = 1 - \frac{6}{1+6+1} = 0.75 \\ l_3 = 1 - \frac{q_3}{\sum_{n'=1}^{4} q_{n'}} = 1 - \frac{1}{1+6+1} = 0.125 \\ l_4 = 1 - \frac{q_4}{\sum_{n'=1}^{4} q_{n'}} = 1 - \frac{1}{1+6+1} = 0.125 \end{cases} \tag{5.1}$$

Deste modo,a probabilidade, considerando $\alpha = 0.5$, será dada por:

$$\begin{cases} P_{26} = \frac{\tau_{26}^1 + \alpha l_2}{1+\alpha(4-1)} = \frac{0.5+0.5\cdot 0.75}{1+0.5\cdot(4-1)} = 0.35 \\ P_{36} = \frac{\tau_{36}^1 + \alpha l_3}{1+\alpha(4-1)} = \frac{0.25+0.5\cdot 0.125}{1+0.5\cdot(4-1)} = 0.125 \\ P_{46} = \frac{\tau_{46}^1 + \alpha l_4}{1+\alpha(4-1)} = \frac{0.25+0.5\cdot 0.125}{1+0.5\cdot(4-1)} = 0.125 \end{cases} \tag{5.2}$$

Conforme os resultados das probabilidades das expressões em (5.2), a ligação entre os nós 1 e 2, possui maior valor de probabilidade, ou seja, esta ligação possui maior probabilidade de ser escolhida para o encaminhamento da formiga *Forward* devido ao valor elevado da intensidade de feromonas, apesar de apresentar maior quantidade de pacotes em fila de espera (Figura 5.1). O inconveniente de escolher

esta ligação relaciona-se com o facto dela possuir uma largura de banda de $1Mbps$, haver um tráfego UDP a $1Mbps$ e elevado número de pacotes em fila de espera. Isto significa que esta ligação está a ser utilizada a 100% (apenas pelo tráfego UDP) e qualquer tráfego que seja adicionado a esta ligação dará origem à perda de pacotes, em consequência do enchimento das filas de espera, pelo menos até que a rede responda a esse enchimento.

De modo a melhorar esta situação propõe-se uma alteração à expressão (3.8) de forma a incluir o factor de utilização da largura de banda.

Este factor é aqui definido por $Util_n^k$ e representa o valor relativo do factor de utilização da largura de banda, que indica o quanto a ligação $k-n$ está a ser utilizada, analisamos três possibilidades de colocação do factor $Util_n^k$, nomeadamente:

(1) em conjunto com o l_n - neste caso tenta-se não beneficiar as decisões de encaminhamentos derivadas do maior peso das feromonas:

$$P'_{nd} = \frac{\tau_{nd}^k + \alpha(l_n + (1 - Util_n^k))}{1 + \alpha(|N_k| - 1)}, \qquad \forall n \in N_k \tag{5.3}$$

(2) repesenta o peso do τ_{nd}^k - neste caso tenta-se equilibrar as decisões de encaminhamentos das feromonas e largura de banda disponível, ou seja, supondo um valor fixo de feromonas, quando maior for o $(1 - Util_n^k)$ maior será o produto $\tau_{nd}^k(1 - Util_n^k)$:

$$P'_{nd} = \frac{\tau_{nd}^k(1 - Util_n^k) + \alpha l_n}{1 + \alpha(|N_k| - 1)}, \qquad \forall n \in N_k \tag{5.4}$$

(3) "repesenta" o peso do P_{nd} (da expressão (3.8)) - neste caso tenta-se criar um equilibrio de decisões de encaminhamentos entre feromonas mais as filas de espera e a largura de banda disponível:

$$P'_{nd} = \left(\frac{\tau_{nd}^k + \alpha l_n}{1 + \alpha(|N_k| - 1)}\right) \cdot (1 - Util_n^k), \qquad \forall n \in N_k \tag{5.5}$$

Destas três expressões, escolhemos a (5.5) por equilibrar as decisões de encaminhamento entre feromonas, tamanho das filas de espera e a largura de banda disponível.

5.3 Algoritmo de encaminhamento *AntNetBw*

De acordo com a descrição da secção anterior criou-se uma variante do algoritmo *AntNet*: *AntNetBw*. O *AntNetBw* utiliza uma heurística associada à largura de banda disponível para fazer a escolha dos caminhos das formigas *Forward*. É uma versão criada com objectivos de reduzir o congestionamento entre as ligações, através da utilização das que possuem maior disponibilidade de largura de banda. Como resultado disso espera-se reduzir também as perdas devido a congestionamentos. Foi feita alteração ao algoritmo *AntNet* ao nível da expressão (3.8), sendo a nova expressão para o cálculo de P'_{nd} dada por:

$$P'_{nd} = \left(\frac{\tau^k_{nd} + \alpha l_n}{1 + \alpha(|N_k| - 1)} \right) \cdot (1 - Util^k_n), \qquad \forall n \in N_k \tag{5.6}$$

Deste modo aos factores de decisão para o encaminhamento da formiga, como a intensidade de feromonas, as filas de espera e a memória acresce o parâmetro de largura de banda (Figura 5.2).

Na próxima secção vamos aplicar a mesma ideia ao algoritmo ϵ-*DANTENetBw*.

5.4 Algoritmo de encaminhamento ϵ-*DANTENetBw*

O algoritmo ϵ-*DANTENetBw* é uma variante do algoritmo ϵ-*DANTENet*, que associa além do comportamento exploratório, a largura de banda para efectuar o encaminhamento das formigas *Forward* ($Ants$ e $DANTE's$), nos nós intermédios.

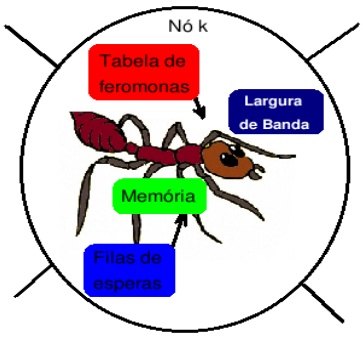


Figura 5.2: Factores de decisão para a escolha do nó n por onde deve seguir a formiga $F_{s \to d}$ para alcançar o destino d quando se leva em consideração a largura de banda.

A semelhança do *AntNetBw*, este algoritmo foi criado com objectivos de reduzir o congestionamento entre as ligações, através do aumento da probabilidade de escolha de ligações que possuem maior disponíbilidade de largura de banda conjugado com os critérios de escolha do método ϵ-*DANTENet*. Foi feita a alteração do algoritmo ao nível da expressão (3.8) sendo modificada para a expressão (5.6). A largura de banda disponível foi acrescentada aos factores de decisão para o encaminhamento das formigas *Forwards* (*Ants* e $DANTE's$) (Figura 5.2).

Na próxima secção apresentamos uma última versão que introduz o factor de largura de banda no algoritmo *CR-DANTENet*.

5.5 Algoritmo de encaminhamento *CR-DANTENetBw*

O algoritmo *CR-DANTENetBw* é uma variante do algoritmo *CR-DANTENet*, este último também variante do ϵ-*DANTENet*. Utiliza a largura de banda para influenciar o encaminhamento das formigas *Forward Ants* e $DANTE's$), nos nós intermédios. Foi também criada com objectivos de reduzir o congestionamento entre as ligações e espera-se, a semelhança dos anteriores, uma redução nas perdas devido ao congestionamento. A alteração efectuada ao *CR-DANTENet* foi ao nível da expressão

(3.8) sendo substituida pela (5.6).

Deste modo a largura de banda passa a fazer parte dos factores de decisão para o encaminhamento das formigas *Forwards*, como a intensidade de feromonas, as filas de espera e a memória.

5.6 Sumário

A utilização da largura de banda surge como um parâmetro que tenta distribuir os pacotes na rede de forma uniforme de acordo com a disponibilidade da largura de banda, feromonas e filas de espera. Esta solução atribui igual peso ao parâmetro utilização da largura de banda e ao conjunto feromonas e filas de espera da expressão (3.8). Assim, este parâmetro foi adicionado aos factores de decisão de encaminhamento das formigas *Forwards*, dando origem aos novos algoritmos: *AntNetBw*, ϵ-*DANTENetBw* e *CR-DANTENetBw*.

No próximo capítulo veremos qual foi o impacto que este parâmetro teve nas perdas de pacotes nos cenários de redes que serão apresentados.

CAPÍTULO

6

Resultados experimentais

Neste capítulo pretendemos analisar os resultados obtidos pelos algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw* e *CR-DANTENetBw*, para tentar determinar o algoritmo que melhor se comporta em determinadas condições de rede. O estudo basear-se-á nas médias de perdas de pacotes TCP e UDP. Neste mesmo capítulo estudamos a influência dos parâmetros Δt (intervalo de lançamento de formigas na rede), r (factor que determina a diferença entre as intensidades de feromonas aquando da sua actualização, favorecendo o caminho encontrado em detrimento dos restantes - ver expressão (3.14)) e α (coeficiente de correcção heurístico l_n - ver expressão (3.9)), na perda de pacotes na rede.

6.1 Breves considerações

Os algoritmos ACO utilizados para redes com fios, apresentam um bom desempenho, pelo menos ao nível de simulação, em comparação com os algoritmos clássicos, principalmente nos casos de redes muito densas. Assim sendo, de forma a analisar os algoritmos apresentados nos Capítulos 3, 4 e 5. Começaremos por analisar a influência dos factores Δt, α e r, na quantidade de pacotes perdidos na rede. Para isso vamos definir uma topologia de rede que será vista na secção seguinte.

A topologia de rede (*REDE 1*) apresenta 14 nós com 15 ligações do tipo duplex com um atraso de $50ms$. Nesta rede criamos duas situações com diferentes larguras de banda entre as ligações:

- na primeira situação (Figura 6.1(a)) as ligações entre os nós (*routers*) internos possuem a mesma largura de banda (*REDE 1A*); e
- na segunda situação (Figura 6.1(b)) as ligações entre os nós (*routers*) internos (centrais) possuem diferentes larguras de banda (*REDE 1B*).

(a) *REDE 1A* - As ligações entre os nós internos apresentam a mesma largura de banda.

(b) *REDE 1B* - Algumas ligações entre os nós internos apresetam diferentes larguras de banda.

Figura 6.1: *REDE 1* - Cenários de rede com 14 nós e 15 ligações.

Em cada situação criada definimos dois conjuntos de trafégos, que estão identificados como *TRÁFEGO I* (Tabela 6.1) e *TRÁFEGO II* (6.2).

Tabela 6.1: *TRÁFEGO I* - Tipo de tráfego, nós de origem e destino dos pacotes e os tempos de transmissão definidos para *REDE 1A* e *REDE 1B*.

Tráfego TCP e UDP					
Emissor	Receptor	Tipo de pacote	Ritmo (kbps)	Inicio da transmissão(s)	Fim da Transmissão(s)
0	8	UDP	900	0.0	30.0
1	9	UDP	900	1.5	30.0
11	10	UDP	900	0.75	30.0
13	1	UDP	900	1.0	30.0
9	3	UDP	900	0.0	30.0
12	3	UDP	900	0.5	30.0
9	0	TCP	-	0.0	30.0
13	11	TCP	-	1.0	30.0

Tabela 6.2: *TRÁFEGO II* - Tipo de tráfego, nós de origem e destino dos pacotes e os tempos de transmissão definidos para *REDE 1A* e *REDE 1B*.

Tráfego UDP					
Emissor	Receptor	Tipo de pacote	Ritmo (kbps)	Início da transmissão(s)	Fim da Transmissão(s)
0	8	UDP	900	0.0	30.0
1	9	UDP	900	1.5	30.0
11	10	UDP	900	0.75	30.0
13	1	UDP	900	1.0	30.0
9	3	UDP	900	0.0	30.0
12	3	UDP	900	0.5	30.0
9	0	UDP	900	0.0	30.0
13	11	UDP	900	1.0	30.0

Baseando-se no estudo apresentado por García *e*t al. [2008], efectuamos 25 simulações para cada tráfego de cada topologia, sob as mesmas condições, durante um período de tempo de 30 segundos; variando os parâmetros Δt, r e α, introduzidos no Capítulo 3. Tal como no Capítulo 4, o simulador utilizado para o efeito foi o $ns-2$ descrito no Apêndice A.

6.2 Influência dos parâmetros Δt, r e α nos pacotes perdidos na rede

Para estudar a influência dos parâmetros Δt, r e α no número de pacotes úteis perdidos na rede, vamos utilizar como referência o algoritmo *AntNet* na REDE 1B com o TRÁFEGO I.

Os valores dos parâmetros r, Δt, e α podem ser visualizados na Tabela 6.3. Os valores do parâmetro r foram escolhidos com base nos valores das primeiras simulações efectuadas com o algoritmo *AntNet*, quando este foi implementado e testado, descrito por Lima [2009], sendo o valor de $r = 0.01$ escolhido como um valor intermédio entre $r = 0.001$ e $r = 0.05$. Os valores de $\Delta t = 0.03$ e $\alpha = 0.45$ também foram escolhidos com base nas primeiras simulações efectuadas com o algoritmo *AntNet*, sendo que para verificar o comportamento da rede, quando esses parâmetros alteram os seus valores, foi escolhido um valor inferior e outro superior para cada parâmetro.

Tabela 6.3: Valores dos parâmetros Δt, r e α, utilizados para analisar a sua influência no número de pacotes úteis perdidos na REDE 1B.

Δt	r	α
0.01	0.001	0.1
0.03	0.01	0.45
0.05	0.03	0.9

Comecemos por analisar a influência do parâmetro Δt, quando variamos os valores de r e mantemos o parâmetro α fixo. A Tabela 6.4, mostra os resultados das médias percentuais de pacotes perdidos com $\alpha = 0.1$. Com $(r, \alpha) = (0.001, 0.1)$, a média percentual do número dos pacotes úteis perdidos atinge o seu valor mais baixo quando $\Delta t = 0.05$. Nos casos em que $(r, \alpha) = (0.01, 0.1)$ e $(r, \alpha) = (0.05, 0.1)$ o cenário repete-se, ou seja, a média dos valores relativos percentuais, do número de pacotes úteis perdidos, é mais baixo quando $\Delta t = 0.05$. Isto remete-nos para a

seguinte conclusão, para o caso em estudo (*REDE 1B* com *TRÁFEGO I*): quando os parâmetros $(r, \Delta t) = (0.01, 0.05)$, a média dos valores relativos percentuais dos pacotes úteis perdidos apresenta um valor mais baixo.

Tabela 6.4: Valores médios, μ, e desvios padrão, s, dos valores relativos percentuais entre os pacotes úteis perdidos e os pacotes úteis que circulam na *REDE 1B*, com *TRÁFEGO I* e diferentes intervalos de lançamento das formigas (Δt) e r, fixando $\alpha = 0.1$, para o algoritmo *AntNet*.

$\alpha = 0.1$			
$r \setminus \Delta t$	0.01	0.03	0.05
0.001	$\mu = 9.1087$ s=0.570	$\mu = 9.072$ s=0.540	$\mu = 8.621$ s=0.546
0.01	$\mu = 8.400$ s=2.981	$\mu = 7.217$ s=2.713	$\mu =$ **6.625** s=1.207
0.05	$\mu = 8.865$ s=2.985	$\mu = 8.742$ s=2.567	$\mu = 8.370$ s=3.003

A restante análise da influência do parâmetro α foi feita com o recurso às Tabelas 6.5 e 6.6, que representam os valores médios das percentagens de pacotes úteis perdidos, na *REDE 1B*, quando o parâmetro α toma os valores de 0.03 e 0.05, para além do valor de 0.01 apresentado na Tabela 6.4.

Tabela 6.5: Valores médios, μ, e desvios padrão, s, dos valores relativos percentuais entre os pacotes úteis perdidos e os pacotes úteis que circulam na *REDE 1B*, com *TRÁFEGO I* e diferentes intervalos de lançamento das formigas (Δt) e r fixando $\alpha = 0.45$, para o algoritmo *AntNet*.

$\alpha = 0.45$			
$r \setminus \Delta t$	0.01	0.03	0.05
0.001	$\mu = 9.1087$ s=0.570	$\mu = 9.072$ s=0.540	$\mu = 8.621$ s=0.546
0.01	$\mu = 8.400$ s=2.981	$\mu = 7.217$ s=2.713	$\mu =$ **6.625** s=1.207
0.05	$\mu = 8.865$ s=2.985	$\mu = 8.742$ s=2.567	$\mu = 8.370$ s=3.003

Da análise feita aos resultados das Tabelas 6.4, 6.5 e 6.6, para o parâmetro α, concluímos que os valores médios das percentagens de perda de pacotes para $\alpha = 0.1$

Tabela 6.6: Valores médios, μ, e desvios padrão, s, dos valores relativos percentuais entre os pacotes úteis perdidos e os pacotes úteis que circulam na *REDE 1B*, com *TRÁFEGO I* e diferentes intervalos de lançamento das formigas (Δt) e r fixando $\alpha = 0.9$, para o algoritmo *AntNet*.

$\alpha = 0.9$			
$r \setminus \Delta t$	0.01	0.03	0.05
0.001	$\mu = 8.300$ s=0.566	$\mu = 8.641$ s=0.534	$\mu = 8.646$ s=0.596
0.01	$\mu = 8.055$ s=2.340	$\mu =$ **7.280** s=1.009	$\mu = 7.520$ s=1.296
0.05	$\mu = 8.431$ s=2.780	$\mu = 9.079$ s=4.010	$\mu = 7.990$ s=2.132

e $\alpha = 0.45$ não apresentam diferenças. Para $\alpha = 0.9$ o valor médio percentual das perdas ocorre quando $(r, \Delta t) = (0.1, 0.03)$.

Das observações anteriores, podemos retirar algumas possíveis ilações:

parâmetro α: Uma vez que este parâmetro, α, está ligado à expressão de determinação do *next hop* para as formigas, mais concretamente é um peso associado ao factor l_n da expressão (3.8), um valor baixo deste parâmetro, atribui menor valor a este factor e consequentemente às filas de espera, valorizando mais as intensidades de feromonas presentes e conduzindo a um valor reduzido da probabilidade de escolher as ligações que apresentam maior número de pacotes em filas de espera. Um valor mais elevado de α conduz a uma maior valorização do factor heurístico l_n, aumentando deste modo a probabilidade de escolher ligações com maior número de pacotes em filas de espera. Provavelmente, esta valorização origina uma sequência de decisões não muito boas, pois as formigas são levadas a seguir determinados caminhos, em grande medida, devido ao enchimento das filas de espera e não às condições dos caminhos (feromonas). Para o valor de $\alpha = 0.9$ os resultados encontrados, indicam a média mais elevada do número de pacotes perdidos. Um valor

aproximadamente médio deste,revela-se ser ideal;

parâmetro r: Este parâmetro está ligado à actualização das tabelas de feromonas. Encontra-se na expressão (3.12) e determina a forma como são incrementadas ou decrementadas as intensidades das feromonas. Se este factor for baixo (valor próximo de zero), o incremento das intensidades das feromonas será feito mais lentamente à medida que as formigas *Backwards* passam por cada nó. Um incremento lento possui a desvantagem da escolha do melhor caminho convergir também de forma lenta. Em contrapartida se este factor for bastante elevado (próximo de um), ficamos perante uma situação que podemos classificar como instável na medida em que uma actualização feita pela formiga *Backward* pode induzir a uma situação de um bom caminho num instante, e de um mau caminho, no instante de actualização seguinte; i.e., ficamos perante uma situação de convergência rápida para bons e maus caminhos. Isto poderá ser justificado pelo facto deste factor assumir um bom compromisso com um valor intermédio de 0.01 onde a convergência é feita de forma "normal" (ou regular), dando origem a uma baixa média da percentagem de perdas de pacotes úteis;

parâmetro Δt: Este parâmetro, define o intervalo entre lançamentos das formigas *Forwards*, numa situação que podemos chamar de vazio na rede, ou seja, numa situação em que não existe tráfego útil na rede, permite a obtenção de melhores caminhos num curto espaço de tempo, quando possui um valor baixo (próximo de zero). Em situações de tráfego o cenário é diferente pois, é necessário encontrar um valor ideal dado que a rapidez de convergência das soluções é condicionada pelo tráfego existente na rede, uma vez que as formigas *Forwards* são encaminhadas como se fossem pacotes comuns. Dos

resultados das simulações realizados, pode-se ver que entre os valores estudados, o que apresenta melhores resultados é $\Delta t = 0.05$.

6.3 Comparação entre os algoritmos

Na secção 6.2 fizemos uma análise da influência dos factores Δt, r e α, com o objectivo de encontrar um conjunto de valores, dentre os ilustrados na Tabela 6.3, que não nos conduzem a situações de elevadas perdas numa rede de dados. Nesta secção utilizaremos os valores que assumiram o compromisso de baixas perdas para fazer a comparação entre os algoritmos *AntNet*, *ϵ-DANTENet*, *CR-DANTENet*, *AntNetBw*, *ϵ-DANTENetBw*, *CR-DANTENetBw*.

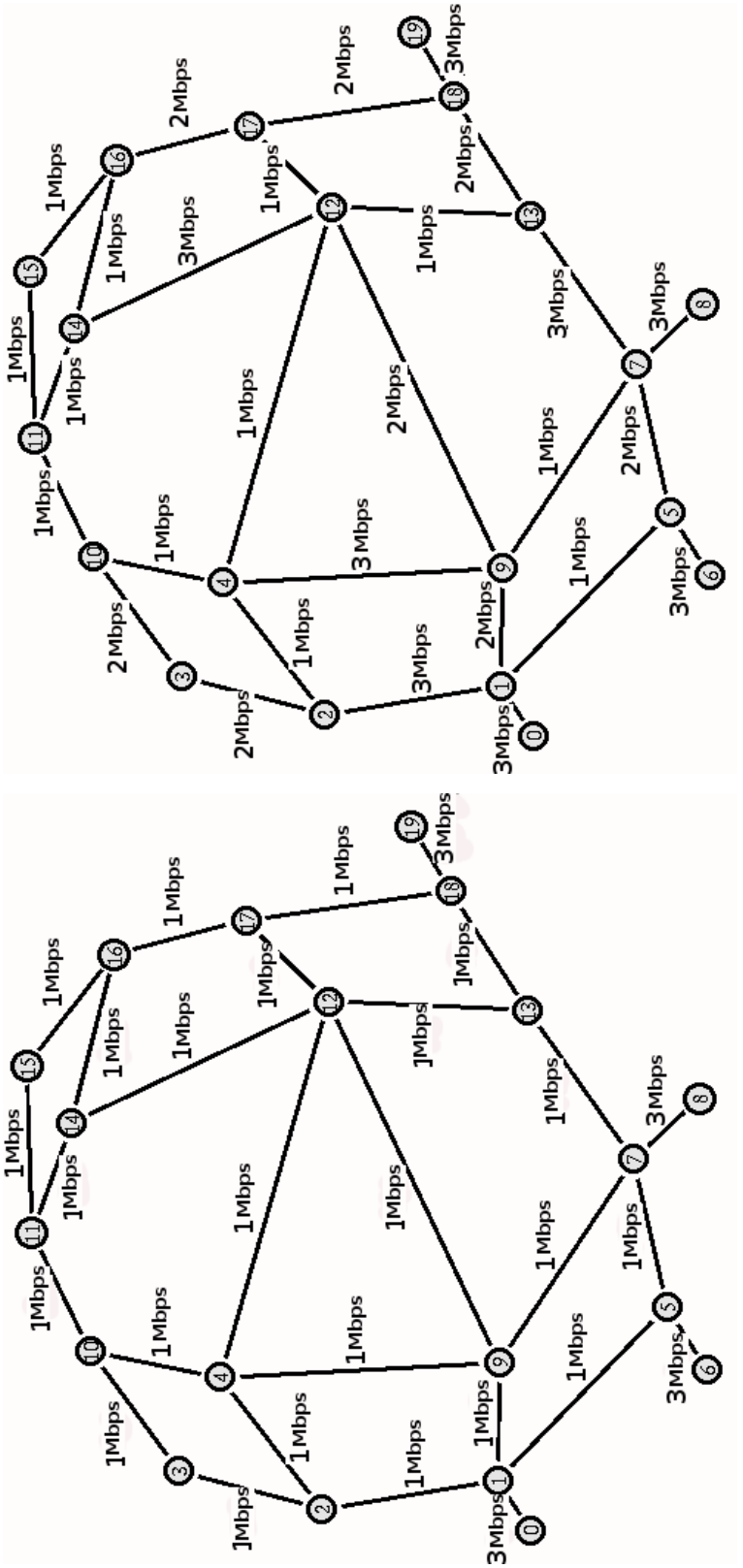
Estes algoritmos serão ainda comparados com o *Link-State* que servirá de análise como método clássico de referência.

Assim de acordo com a análise feita, e com o intuito de reduzir ao máximo as perdas dos pacotes, o parâmetro r assumirá o valor 0.01; o parâmetro α assumirá o valor 0.45, como forma de ter um peso do factor heurístico l_n na expressão (3.8) ligeiramente superior ao de 0.1. Tanto o parâmetro r como o parâmetro α são utilizados em expressões ligadas às formigas da rede enquanto que o Δt é um parâmetro utilizado para gerar os formigas em intervalos regulares. Como já vimos o parâmetro Δt exerce bastante influência na quantidade de tráfego que é gerado na rede. Uma vez que o nosso objectivo não é inundar a rede com tráfego, pois isso daria origem a excessivas perdas de pacotes, vamos escolher um valor de Δt que não seja muito baixo, mas que possibilite a criação de bons caminhos para a redução das perdas. Nota-se que um valor muito alto do Δt, poderá não ser uma boa opção pois estaremos a reduzir o intervalo de criação de novas formigas *Forward* para descobrir ou actualizar as infor-

mações de encaminhamentos. De acordo com a análise feita na Secção 6.2 o valor de $\Delta t = 0.05$ revelou-se razoável por produzir menores perdas. No entanto, nesta secção resolvemos utilizar dois valores para este parâmetro, nomeadamente $\Delta t = 0.03$ e o referido $\Delta t = 0.05$.

Além da *REDE 1* foi ainda considerada uma topologia de rede (*REDE 2*) mais densa que apresenta 20 nós e 28 ligações. Estas são do tipo duplex, com um atraso de $50ms$. À semelhança da *REDE 1*, nesta foram também criada duas situações com diferentes larguras de banda:

- as ligações entre os nós (*routers*) internos possuem a mesma largura de banda - *REDE 2A* (Figura 6.2(a)); e
- algumas ligações entre os nós (*routers*) internos possuem diferentes larguras de banda - *REDE 2B* (Figura 6.2(b)).



(a) *REDE 2A* - As ligações entre os nós internos apresentam a mesma largura de banda.

(b) *REDE 2B* - Algumas ligações entre os nós internos apresetam diferentes larguras de banda.

Figura 6.2: *REDE 2* - Cenários de rede com 20 nós e 28 ligações.

Os tráfegos definidos para cada situação estão identificados como *TRÁFEGO III* (Tabela 6.7) e *TRÁFEGO IV* (Tabela 6.8)

Tabela 6.7: *TRÁFEGO III* - Tipo de tráfego, nós de origem e destino dos pacotes e os tempos de transmissão definidos para *REDE 2A* e *REDE 2B*.

Tráfego TCP e UDP					
Emissor	Receptor	Tipo de pacote	Ritmo (kbps)	Inicio da transmissão(s)	Fim da Transmissão(s)
0	16	TCP	-	0.0	30.0
0	19	UDP	900	0.0	30.0
0	11	UDP	900	0.0	30.0
6	15	UDP	900	0.0	30.0
6	10	TCP	-	0.0	30.0
6	17	UDP	900	0.0	30.0
8	3	TCP	-	0.0	30.0
8	16	TCP	-	0.0	30.0
8	11	UDP	900	0.0	30.0
19	3	UDP	900	0.0	30.0
19	0	UDP	900	0.0	30.0
19	11	TCP	-	0.0	30.0

Tabela 6.8: *TRÁFEGO IV* - Tipo de tráfego, nós de origem e destino dos pacotes e os tempos de transmissão definidos para *REDE 2A* e *REDE 2B*.

Tráfego UDP					
Emissor	Receptor	Tipo de pacote	Ritmo (kbps)	Inicio da transmissão(s)	Fim da Transmissão(s)
0	16	UDP	900	0.0	30.0
0	19	UDP	900	0.0	30.0
0	11	UDP	900	0.0	30.0
6	15	UDP	900	0.0	30.0
6	10	UDP	900	0.0	30.0
6	17	UDP	900	0.0	30.0
8	3	UDP	900	0.0	30.0
8	16	UDP	900	0.0	30.0
8	11	UDP	900	0.0	30.0
19	3	UDP	900	0.0	30.0
19	0	UDP	900	0.0	30.0
19	11	UDP	900	0.0	30.0

O objectivo da criação desta topologia de rede é o de efectuar a comparação entre um cenário em que temos poucos nós e poucas possibilidades de encaminhamento

(REDE 1) e outro com muitos nós e diversas possibilidades de encaminhamento. A principal diferença entre as duas redes está na densidade de nós e ligações.

A comparação entre os algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw*, baseou-se nas médias dos valores relativos dos pacotes perdidos, bem como dos pacotes TCP recebidos nos cenários com o tráfego TCP e UDP. O objectivo desta comparação é o de verificar se houve melhorias nos algoritmos descritos nos Capítulos 4 e 5.

6.3.1 Pacotes úteis perdidos na rede

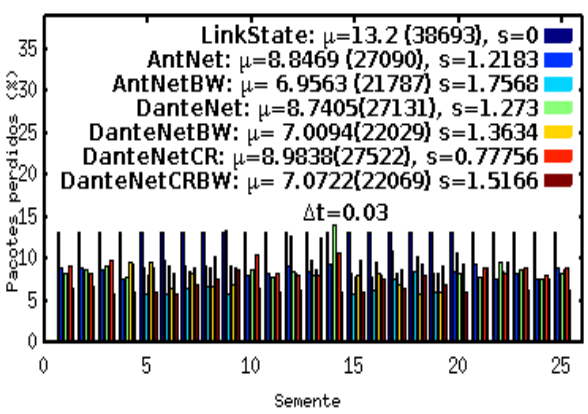
A preocupação constante em reduzir o número de pacotes perdidos na rede, surge com a adopção de critérios de QoS que oferecem serviços diferenciados aos seus clientes, estes critérios obrigam a que os pacotes transmitidos sejam previamente marcados de modo a terem o seu devido tratamento ao nível dos nós da rede.

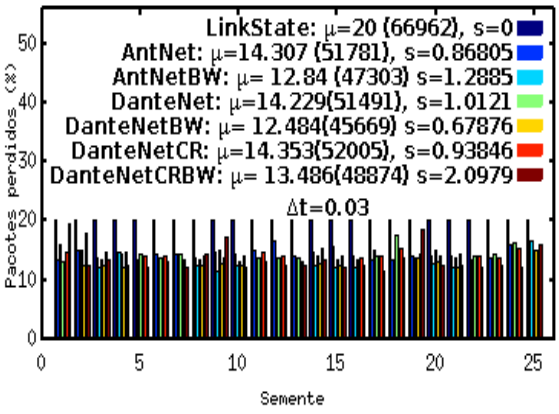
A quantificação ao nível percentual do número de pacotes perdidos ajuda-nos a ter uma ideia do comportamento de um dado algoritmo na rede.

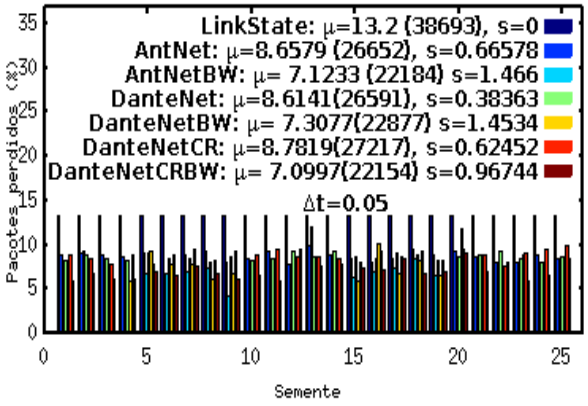
Para efectuar a comparação dos algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw*, ao nível de pacotes úteis perdidos na rede vamos considerar, conforme descrito, a *REDE 1* e *REDE 2* bem como o *TRÁFEGO I, II, III e IV*.

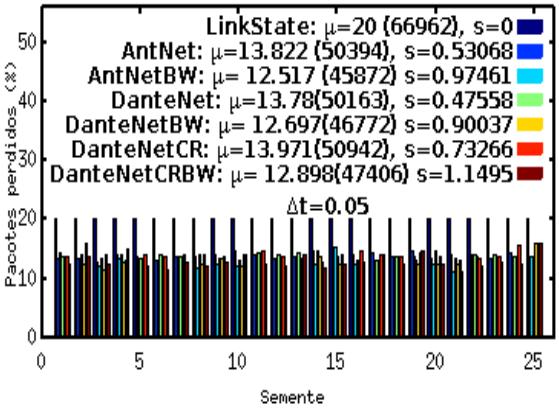
Considerando a *REDE 1A* e *TRÁFEGOS I e II*, o valor relativo percentual das perdas de pacote úteis em cada um dos algoritmos mencionados, pode ser visualizado nos gráficos da Figura 6.3. Os gráficos da parte superior desta figura, referem-se às perdas quando o parâmetro $\Delta t = 0.03$ e os da parte inferior, quando o $\Delta t = 0.05$. O parâmetro μ representa a média dos valores relativos percentuais de perdas de pacotes, sendo que os valores entre parênteses representam o número de pacotes perdidos na

rede, e s representa os desvio padrão.









(a) Médias percentuais das perdas dos pacotes TCP e UDP da *REDE 1A* com o *TRÁFEGO I*.

(b) Médias percentuais das perdas dos pacotes UDP da *REDE 1A* com o *TRÁFEGO II*.

Figura 6.3: Médias percentuais das perdas dos pacotes para os algoritmos *LinkState*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw*, da *REDE 1A* com *TRÁFEGO I* (à esquerda) e *TRÁFEGO II* (à direita).

Analisando os gráficos das Figuras 6.3(a) e 6.3(b), notamos que o algoritmo *AntNet*, que é usado como base de comparação dos algoritmos ACO, apresenta um valor baixo da média de perdas quando o $\Delta t = 0.05$. Em consequência disto a análise seguinte será feita considerando este valor pois queremos efectuar a comparação para o caso em que o *AntNet* apresenta uma situação óptima.

Assim, nos casos em que o $\Delta t = 0.05$, os algoritmos são ordenados em função das perdas de pacotes (de menores para maiores perdas) da seguinte forma:

- **Para o *TRÁFEGO I* (TCP e UDP):**

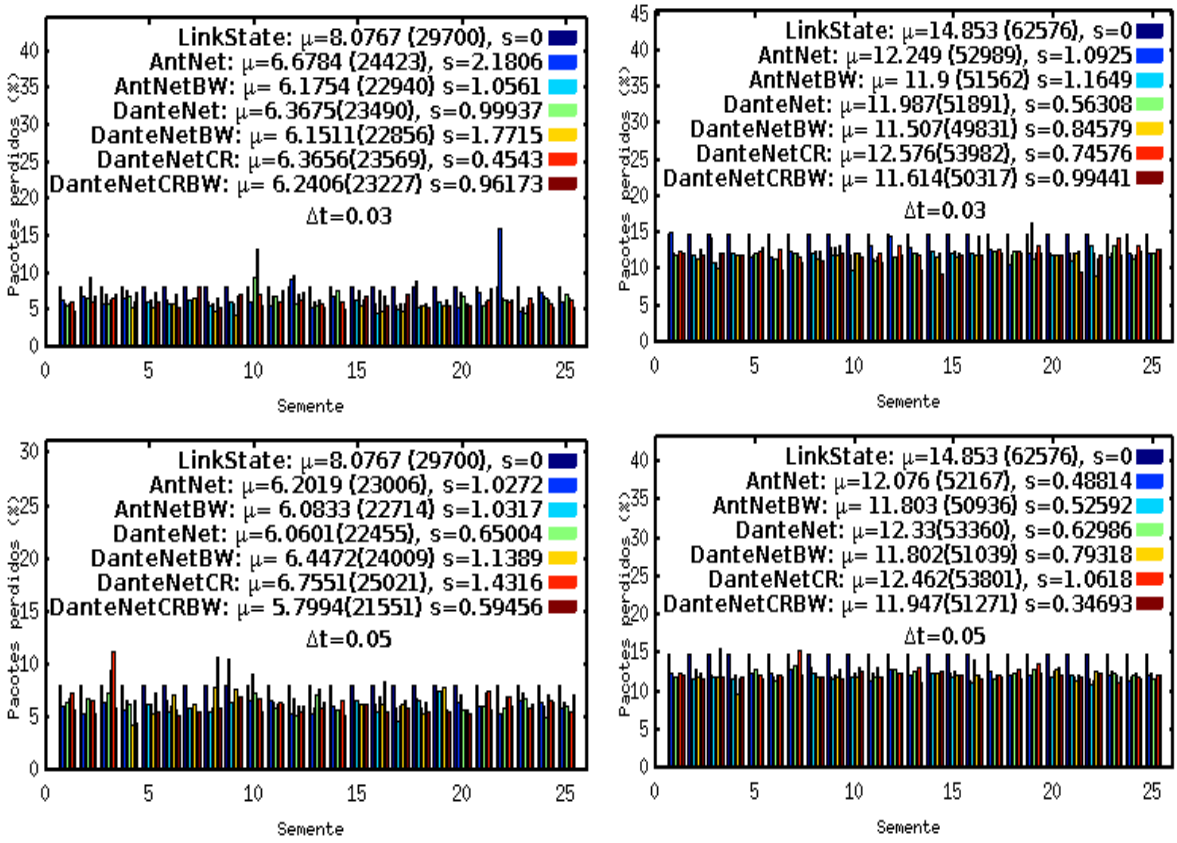
  1º *CR-DANTENetBw;*
  2º *AntNetBw;*
  3º ϵ*-DANTENetBw;*
  4º ϵ*-DANTENet;*
  5º *AntNet;*
  6º *CR-DANTENet;*
  7º *Link-State.*

- **Para o *TRÁFEGO II* (UDP):**

  1º *AntNetBw;*
  2º ϵ*-DANTENetBw;*
  3º *CR-DANTENetBw;*
  4º ϵ*-DANTENet;*
  5º *AntNet;*
  6º *CR-DANTENet;*
  7º *Link-State.*

Nestes casos nota-se claramente que os algoritmos que utilizam a largura de banda, apresentam perdas mais baixas devido ao facto destes explorarem as ligações que são menos utilizadas.

Vamos agora analisar a situação em que na mesma rede, algumas ligações dos nós centrais possuem larguras de banda diferentes. Para tal consideramos a *REDE 1B* e *TRÁFEGOS III e IV*. O valor relativo percentual das perdas de pacote úteis em cada um dos algoritmos estudados, pode ser visualizados nos gráficos da Figura 6.4.



(a) Médias percentuais das perdas dos pacotes TCP e UDP da *REDE 1B* com o *TRÁFEGO I*.

(b) Médias percentuais das perdas dos pacotes UDP da *REDE 1B* com o *TRÁFEGO II*.

Figura 6.4: Médias percentuais das perdas dos pacotes para os algoritmos *Link-State*, *AntNet*, *ϵ-DANTENet*, *CR-DANTENet*, *AntNetBw*, *ϵ-DANTENetBw*, *CR-DANTENetBw*, da *REDE 1B* com *TRÁFEGO I* (à esquerda) e *TRÁFEGO II* (à direita).

Para o caso dos gráficos das Figuras 6.4(a) e 6.4(b), trabalharemos com o valor de $\Delta t = 0.05$ pelos mesmos motivos mencionados anteriormente. Assim, os algoritmos ficam ordenados da seguinte forma:

- **Para o *TRÁFEGO I* (TCP e UDP):**

  1º *CR-DANTENetBw;*

  2º *ϵ-DANTENet;*

3º *AntNetBw;*
4º *ϵ-DANTENetBw;*
5º *AntNet;*
6º *CR-DANTENet;*
7º *Link-State.*

- **Para o *TRÁFEGO II* (UDP):**

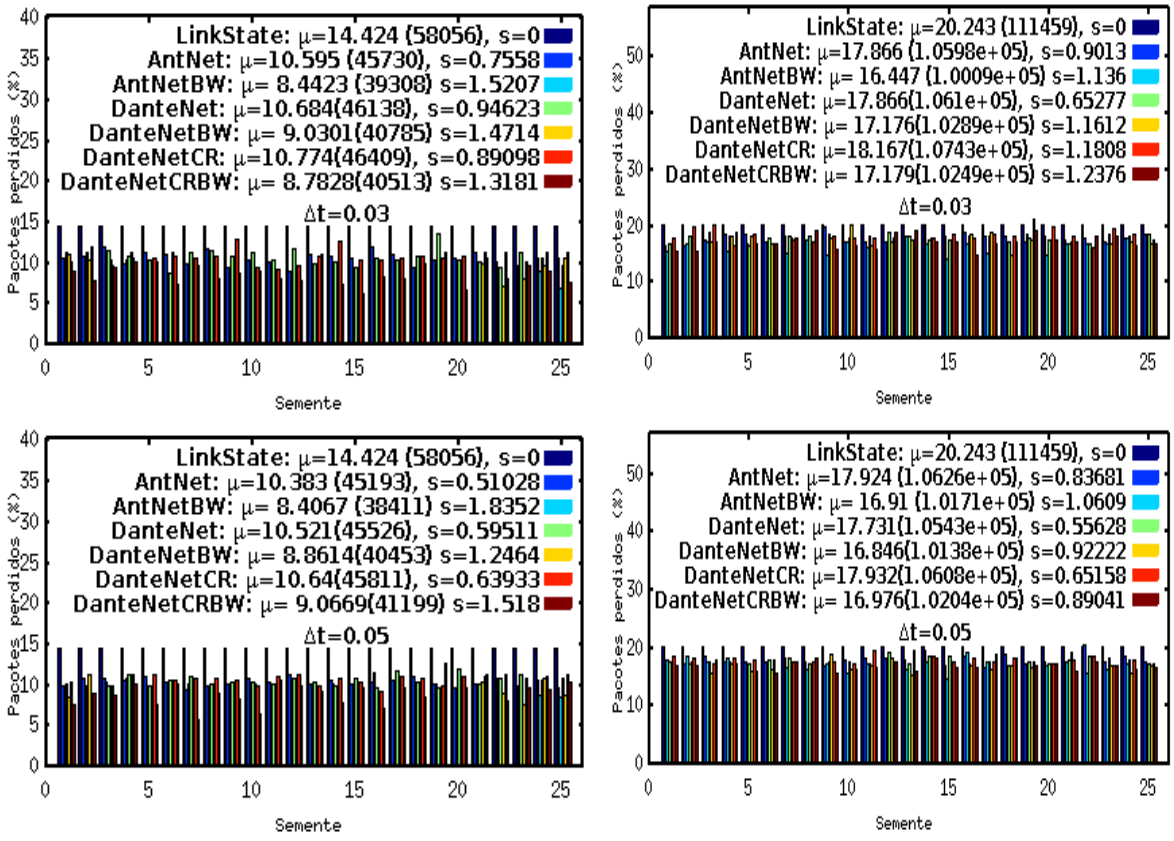
1º *ϵ-DANTENetBw;*
2º *AntNetBw;*
3º *CR-DANTENetBw;*
4º *AntNet;*
5º *ϵ-DANTENet;*
6º *CR-DANTENet;*
7º *Link-State.*

Estes casos ajudam-nos a perceber como funciona a dinâmica destes algoritmos. De acordo com a descrição do Capítulo 5, os algoritmos que utilizam a largura de banda visam evitar o congestionamente e com isso reduzem a probabilidade de existência de muitas perdas na rede. Este facto é notório nos exemplos acima referidos pois existe uma grande tendência dos algoritmos *CR-DANTENetBw*, e *ϵ-DANTENetBw* apresentarem melhores resultados em relação ao *AntNet*.

O estudo estatístico realizado para a *REDE 1A*, aplicando o teste t (Tabelas C.1 e C.2 do Apêndice C) mostra que os algoritmos *AntNetBw*, *ϵ-DANTENetBw* e *CR-DANTENetBw* são efectivamente melhores que o *AntNet*.

Para a *REDE 1B*, esse estudo (Tabelas C.3 e C.4 do Apêndice C) mostra que os algoritmos *AntNetBw*, *ϵ-DANTENetBw* e *CR-DANTENetBw* também são efectivamente melhores que o *AntNet*. Os testes estatisticos não nos permitem concluir qual destes três é o melhor.

No caso da REDE 2A com os *TRÁFEGOS III e IV*, o valor relativo percentual das perdas de pacote úteis para cada algoritmo é mostrado nos gráficos da Figura 6.5.



(a) Médias percentuais das perdas dos pacotes TCP e UDP da *REDE 2A* com o *TRÁFEGO III*.

(b) Médias percentuais das perdas dos pacotes UDP da *REDE 2A* com o *TRÁFEGO IV*.

Figura 6.5: Médias percentuais das perdas dos pacotes para os algoritmos *Link-State*, *AntNet*, *ϵ-DANTENet*, *CR-DANTENet*, *AntNetBw*, *ϵ-DANTENetBw*, *CR-DANTENetBw*, da *REDE 2A* com *TRÁFEGO III* (à esquerda) e *TRÁFEGO IV* (à direita).

Para efeitos de comparação, utilizaremos os gráficos com o $\Delta t = 0.05$ e $\Delta t = 0.03$ nas Figuras 6.5(a) e 6.5(b) correspondentes ao *TRÁFEGO III e IV*. Deste modo, ordenamos os algoritmos em função das suas perdas (das menores para as maiores) da seguinte forma:

- **Para o *TRÁFEGO III* (TCP e UDP):**

  1º *AntNetBw;*

2º *ϵ-DANTENetBw;*

3º *CR-DANTENetBw;*

4º *AntNet;*

5º *ϵ-DANTENet;*

6º *CR-DANTENet;*

7º *Link-State.*

- **Para o *TRÁFEGO IV* (UDP) :**

1º *AntNetBw;*

2º *ϵ-DANTENetBw;*

3º *CR-DANTENetBw;*

4º *ϵ-DANTENet;*

5º *AntNet;*

6º *CR-DANTENet;*

7º *Link-State.*

Uma vez mais os algoritmos que utilizam a largura de banda apresentam perdas mais baixas. Neste caso vamos verificar se, numa rede densa, o facto de termos diferentes larguras de banda, fará com que o encaminhamento se faça através dos nós que se apresentem com maiores larguras de banda. Para tal consideramos a *REDE 2B* e *TRÁFEGOS III e IV*. O valor relativo percentual das perdas de pacote úteis em cada um dos algoritmos estudados, pode ser visualizado nos gráficos da Figura 6.6.

Para o caso dos gráficos das Figuras 6.6(a) e 6.6(b), trabalharemos com o valor de $\Delta t = 0.05$. Assim, a ordenação dos algoritmos a partir dos que apresentam uma média percentual mais baixa para a mais alta de pacotes perdidos, é a seguinte:

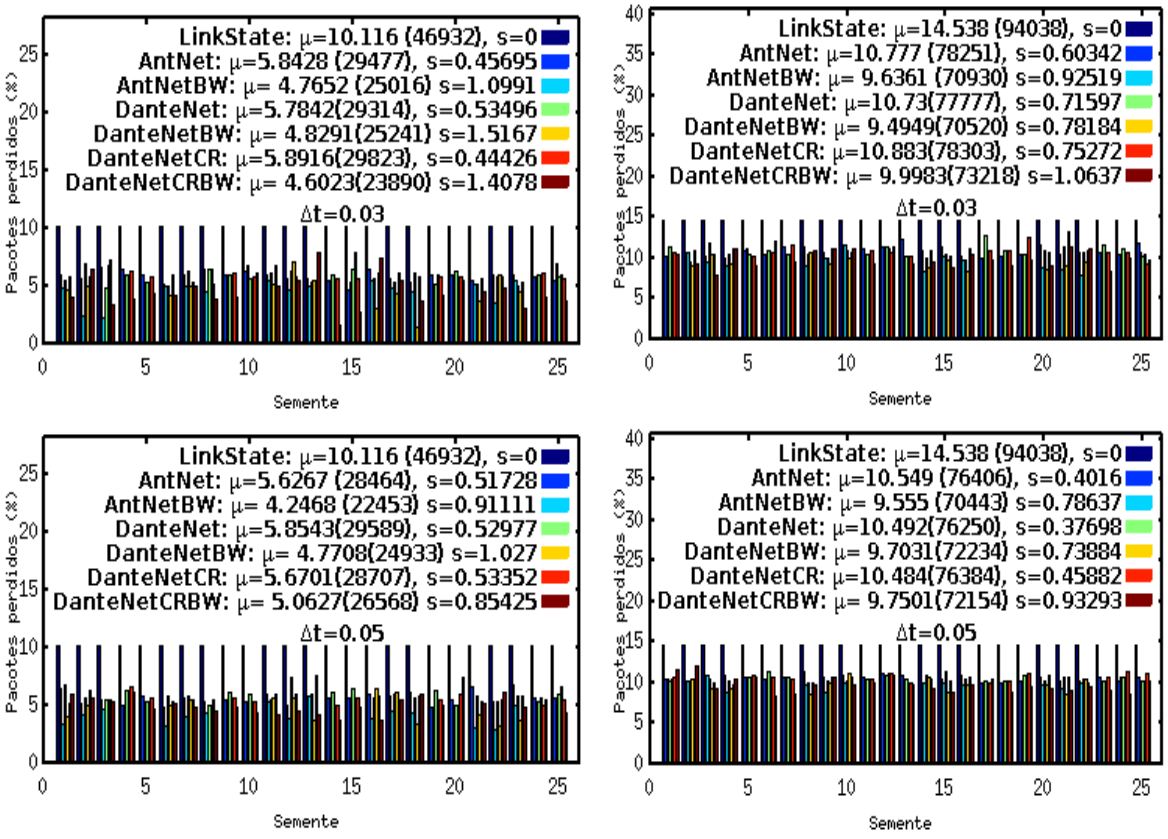
- **Para o *TRÁFEGO III* (TCP e UDP):**

1º *AntNetBw;*

2º *ϵ-DANTENetBw;*

3º *CR-DANTENetBw;*

4º *AntNet;*



(a) Médias percentuais das perdas dos pacotes TCP e UDP da *REDE 2B* com o *TRÁFEGO III*.

(b) Médias percentuais das perdas dos pacotes UDP da *REDE 2B* com o *TRÁFEGO IV*.

Figura 6.6: Médias percentuais das perdas dos pacotes para os algoritmos *Link-State*, *AntNet*, *ϵ-DANTENet*, *CR-DANTENet*, *AntNetBw*, *ϵ-DANTENetBw*, *CR-DANTENetBw*, da *REDE 2B* com *TRÁFEGO III* (à esquerda) e *TRÁFEGO IV* (à direita).

5º *CR-DANTENet;*

6º *ϵ-DANTENet;*

7º *Link-State.*

- **Para o *TRÁFEGO IV* (UDP):**

1º *AntNetBw;*

2º *ϵ-DANTENetBw;*

3º *CR-DANTENetBw;*

4º *CR-DANTENet;*
5º *ϵ-DANTENet;*
6º *AntNet;*
7º *Link-State.*

Na *REDE 1* nota-se que os algoritmos que utilizam a largura de banda, possuem um valor percentual de perdas de pacotes baixo. O estudo estatístico realizado para a *REDE 2*, aplicando o teste t (Tabelas C.5, C.6, C.7 e C.8 do Apêndice C) mostra que os algoritmos *AntNetBw*, *ϵ-DANTENetBw* e *CR-DANTENetBw* apresentam-se como sendo melhores do que o *AntNet*.

Analisando a *REDE 1 e 2*, nota-se uma forte tendência destes algortimos apresentarem melhores resultados quanto mais densas e complexas forem as redes, tanto para pacotes TCP e UDP como apenas para pacotes UDP. Um facto curioso é que os algoritmos *AntNetBw*, *ϵ-DANTENetBw* e *CR-DANTENetBw* não apresentam diferenças significativas, entre si, ao nível de percentagem de pacotes perdidos na rede, segundo dados estatisticos, o que uma vez mais nos leva a concluir que não se sabe qual destes três é o melhor..

Por outro lado os algoritmos *ϵ-DANTENet* e *CR-DANTENet* não apresentam diferenças significativas com o *AntNet*, daí que podemos afirmar com base nos resultados estatísticos que esses algoritmos não melhoram o *AntNet*, tanto em redes simples como para redes densas e complexas.

Um outro teste estatístico utilizado para verificar se as médias percentuais das perdas de pacotes obedecem a uma distribuição normal, foi o teste de *Shapiro-Wilk* conforme a descrição do Apêndice B. Este teste foi escolhido por termos amostras com apenas 25 simulações. Foi aplicado para os algoritmos *AntNet*, *ϵ-DANTENet*, *CR-DANTENet*, *AntNetBw*, *ϵ-DANTENetBw* e *CR-DANTENetBw* e o resultado pode ser visto nas tabelas do Apêndice D. O nível de significância γ escolhido foi de 5%.

Os resultados deste teste sugerem uma forte tendência da distrubuição de perdas de pacotes aproximar-se a uma distribuição normal, quanto mais densa for a rede, para os algoritmos *AntNetBw*, ϵ-*DANTENetBw* e *CR-DANTENetBw*.

6.4 Conclusões

Os algoritmos apresentados nesta dissertação possuem caracteristicas individuais que os diferenciam. Por exemplo o algoritmo ϵ-*DANTENet*, introduz um comportamento exploratório para as redes com fios. Esse comportamento exploratório dá origem à inundação dos pacotes na rede devido aos agentes (formigas artificiais). Na maior parte dos casos ilustrados nos gráficos das Figuras 6.3, 6.4, 6.5 e 6.6, este algoritmo apresenta a média percentual de perdas ligeiramente baixo em comparação com o *AntNet*, sendo que para todos os casos da *REDE 1* com o intervalo de lançamento dos agentes $\Delta t = 0.03$, este algoritmo apresenta melhores resultados em relação ao *AntNet*; podemos assim afirmar que quanto menor for o intervalo de lançamento dos agentes a partir dos nós destino, e quanto menos densa for a rede, tanto maior é a probabilidade deste algoritmo ser melhor em relação ao *AntNet*.

O algoritmo *CR-DANTENet* surge para resolver o problema de inundação provocada pelo algoritmo ϵ-*DANTENet*. Este algoritmo tem a particularidade de criar as *DANTE's* quando as filas de espera para um dado destino se encontram cheias. Podemos assim dizer que se trata de um algoritmo que combina os comportamentos reactivo e exploratórios. Exploratório porque após reagir ao enchimento das filas de espera, procura novas soluções diferentes da presente, de modo a aliviar o congestionamento. Nos gráficos mencionados, nota-se que poucas são as situações em que existem diferenças consideráveis entre os dois algoritmos mencionados.

Em cada um dos gráficos das Figuras 6.3, 6.4, 6.5 e 6.6 os algoritmos criados com

o parâmetro de utilização da largura de banda, apresentam melhorias relactivamente aos algoritmos *AntNet*, ϵ-*DANTENet*, *CR-DANTENet* (à excepção da *REDE 1B* com *TRÁFEGO I* e $\Delta t = 0.05$).

O estudo estatístico feito mostrou que de um modo geral, os algoritmos *AntNetBw*, ϵ-*DANTENetBw* e *CR-DANTENetBw* apresentam médias percentuais de perdas diferentes dos outros algoritmos em análise, sendo no entanto mais baixas.

CAPÍTULO

7

Conclusões e trabalho futuro

Os algoritmos de encaminhamentos propostos neste trabalho tiveram como objectivo melhorar as soluções encontradas pelo *AntNet*. Essas propostas foram inicialmente introduzidas no Capítulo 4, onde é apresentado o ϵ-*DANTENet* como a primeira adaptação do algoritmo ϵ-*DANTE*, para problemas de redes de dados com fios. O ϵ-*DANTENet* é um algoritmo híbrido que combina os algoritmos ACO para o encaminhamento em redes com fios, com a pesquisa em profundidade. Apesar deste algoritmo provocar uma inundação de pacotes na rede, devido às formigas que são geradas (conforme pode ser vista no gráfico da Figura 4.5) este apresenta-se como uma alternativa do *AntNet* para os casos estudados.

Apesar disso, a diferença entre o ϵ-*DANTENet* e o *AntNet* é bastante reduzida. A principal desvantagem deste algoritmo inundar a rede com formigas, produzindo resultados ligeiramente superiores ao *AntNet*. Este facto conduziu-nos à criação do algoritmo *CR-DANTENet* com o objectivo de reduzir o número de formigas geradas

por este algoritmo na rede.

No entanto, a vantagem do algoritmo *CR-DANTENet* não se fez sentir em nenhuma das redes testadas, facto que levou-nos a pesquisar outras soluções que fossem integradas nos algoritmos ACO e que resultassem em perdas reduzidas de pacotes na rede.

Essa pesquisa conduziu-nos à introdução do parâmetro "utilização largura de banda", que foi apresentado no Capítulo 5. A sua introdução foi feita nos algoritmos *AntNet*, ϵ-*DANTENet* e *CR-DANTENet* ao nível da expressão (3.8), tendo-se chegado à expressão (5.6).

Os algoritmos *AntNetBw*, ϵ-*DANTENetBw* e *CR-DANTENetBw* produziram bons resultados para os casos em estudo tal como se pôde verificar no Capítulo 6. O estudo estatístico realizado (Apêndice C), mostra que na maioria dos casos em estudo, estes algoritmos não apresentam diferenças significativas entre si, apresentando diferenças significativas quando comparados com os mesmos métodos sem a utilização da largura de banda. Este estudo foi feito para o nível de significância de 5%.

No entanto, a grande vantagem destes algoritmos está no facto de apresentarem uma baixa percentagem de perda de pacotes em relação ao *AntNet*; facto que os coloca numa melhor posição relativamente a este e nos permite afirmar que são os melhores algoritmos em relação aos restantes estudados. De acordo com os resultados das experiências realizadas no Capítulo 6, estas vantagens são tanto mais visíveis quanto maior e mais densa for a rede e o tráfego. Para uma situação em que temos os *routers* centrais com a mesma largura de banda, estes algoritmos apresentam um bom desempenho, facto que pode ser comprovado nos gráficos das Figuras 6.3 e 6.6.

De um modo geral podemos dizer que foram atingidos todos os objectivos do trabalho, não tendo apenas sido satisfeita apenas a primeira hipótese, conforme o

argumento acima.

Como proposta para trabalho futuro, gostariamos de explorar a possibilidade de encaminhar os pacotes na rede através de ligações que disponibilizassem largura de banda suficiente para responder aos seus ritmos de transmissão, sendo que o encaminhamento através de tais ligações não apresentasse um custo muito superior em relação aos restantes. Além dos custos esse encaminhamento tem de ser feito de forma a minimizar as perdas.

Além disso, pretendemos estudar o método quando são considerados tráfegos com distintas prioridades (QoS). A análise de fórmulas alternativas à expressão (5.5) poderá também ser alvo de trabalho futuro. Uma hipótese será a de considerar o problema como multi-objectivo, o que de alguma forma poderá ir ao encontro do problema de QoS. Ao nível experimental, testes com novas redes com novas topologias e maiores dimensões estão também nos nossos objectivos de estudo.

Referências bibliográficas

Baran, B., e Sosa, R. 2000. *A* new approach for AntNet routing. IEEE Press.

Barán, Benjamín, e Sosa, Rubén. 2001. AntNet: Routing algorithm for data networks based on mobile agents. *R*evista iberoamericana de inteligência artificial, 75–84.

Bellman, Richard Ernest. 1958. *O*n a routing problem. the quartarly of applied mathematics. Vol. 16. Rand Corp.

Bertsekas, Dimitri P. 1995. *D*ynamic programming and optimal control. Vol. I–II. Athena Scientific.

Bertsekas, Dimitri P., e Gallager, Robert G. 1992. *D*ata networks. Prentice Hall.

Blum, Christian, e Merkle, Daniel. 2008. *S*warm intelligence: introduction and applications. Springer.

Bonabeau, E., Henaux, F., Guérin, S., Snyers, D., Kuntz, P., e Theraulaz, G. 1998. Routing in telecommunication networks with "smart" ant-like agents. *I*n: *I*n proceedings of IATA'98, second int. workshop on intelligent agents for telecommunication applications. Springer-Verlag.

Bonabeau, Eric, Dorigo, Marco, e Theraulaz, Guy. 1999. *S*warm intelligence: From natural to artificial systems. Oxford University Press.

Cardoso, Pedro, Jesus, Mário, e Márquez, Alberto. 2010. DANTE: an ant colony oriented depth search procedure. *S*oft Computing. Springer-verlag.

Caro, G. Di, e Vasilakos, T. 2000. Ant-sela: Ant-agents and stochastic automata learn adaptive routing tables for qos routing in atm networks. Ants'2000 -

from ant colonies to artificial Ants: Second international workshop on ant colony optimization. 8-9 de Setembro. Bruxelas, Bélgica.

Cherkassky, B., Goldberg, A. V., e Radzik, T. 1994. *S*hortest paths algorithms: Theory and experimental evaluation. Technical Report. Dep. of Computer Science, Stanford University.

Chung, Jae, e Claypool, Mark. 2009. *N*s by example. http://nile.wpi.edu/NS/, acedido em 13/01/2010.

Cisco Systems. 2002. *I*nternetworking technology handbook. Cisco Press.

Coutinho, Mauro Margalho. 2009. *S*imulação - network simulator. http://www.cci.unama.br/margalho/simulacao/simulacao.htm, acedido em 14/01/2010.

Da Silva, Rodrigo Cunha. 2008 (Junho). *O*timização multiobjectivo baseada em colónia de formigas para o roteamento e consumo de energia em redes de sensores sem fios. Msc. thesis. Universidade Tecnológica Federal do Paraná. Brazil.

De Araújo, Filipe João Boavida de Mendonça Machado. 1999 (Outubro). *A*plicação das redes de hopfield no encaminhamento em redes de dados. Msc. thesis. Departamento de Engenharia Informática. Faculdade de Ciências e Tecnologia. Universidade de Coimbra.

Di Caro, G., e Dorigo, M. 1997a. Adaptive learning of routing tables in communication networks. In proceedings of the italian workshop on machine learning. 9 e 10 de Dezembro. Italia.

Di Caro, G., e Dorigo, M. 1998a. Extending AntNet for best-effort quality-of-service routing. Ants'98 - from ant colonies to artificial Ants: First international workshop on ant colony optimization. 15-16 de Outubro. Bruxelas.

Di Caro, Gianni. 2004 (September). *A*nt colony optimization and its application to adaptive routing in telecommunication networks. Ph.D. thesis, Universidade Livre de Bruxelas.

Di Caro, Gianni, e Dorigo, Marco. 1997b. AntNet: A mobile agents approach to adaptive routing. *I*ridia. Bélgica. Bruxelas.

Di Caro, Gianni, e Dorigo, Marco. 1998b. AntNet: Distributed stigmergetic control for communications networks. *I*ridia. Bélgica. Bruxelas.

Di Caro, Gianni, Ducatelle, Frederick, e Gambardella, Luca Maria. 2004. Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *E*uropean transactions on telecommunications. Istituto Dalle Molle sull'Intelligenza Artificiale (IDSIA). Manno-Lugano. Switzerland.

Doi, S., e Yamamura, M. 2000. Bntnetl: Evaluation of its performance under congestion. *J*ournal of IEICE (in japanese), 1702–1711.

Doi, S., e Yamamura, M. 2002. *B*ntnetl and its evaluation on a situation of congestion. Electronics and communications in japan (part I). John Wiley e Sons. USA.

Dorigo., Marco. 1992. *O*ptimization, learning and natural algorithm (em italiano). Ph.D. thesis, Dipartimento di Electtronica e Informazione, Politecnico di Milano.

Dorigo, Marco, e Stutzle, Thomas. 2004. *A*nt colony optimization. MIT Press. Cambridge.

Dorigo, Marco, Maniezzo, Vittorio, e Colorni, Alberto. 1991. *P*ositive feedback as a search strategy. Tech. rept. Dipartimento di Electtronica e Informazione, Politecnico di Milano.

Dorigo, Marco, Maniezzo, Vittorio, e Colorni, Alberto. 1996. The ant system: Optimization by a colony of cooperating agents. *I*EEE, **26**(1), 1–13.

Dorigo, Marco, Caro, Gianni Di, e Gambardella, Luca Maria. 1999. Ant algorithms for discrete optimization. *I*ridia, université libre de bruxelles.

Doyle, Jeff, e Carroll, Jennifer DeHaven. 2005. *R*outing tcp/ip. Cisco Press.

Du Plessis, Johan. 2005. *A*codv: Ant colony optimisation distance vector routing in ad hoc networks. Ph.D. thesis. University of Pretoria. South Africa.

Ducatelle, Frederick. 2007 (Maio). *A*daptive routing in ad hoc wireless multi-hop networks. Ph.D. thesis, Faculdade de Informática da Universidade della Svizzera Italiana. Italia.

Fall, Kevin, e Varadhan, Kannam. 2009. *T*he ns manual. The VINT Project.

Farooq, Muddassar. 2006. *F*rom the wisdom of the hive to intelligent routing in telecommunication networks: A step towards intelligent network management through natural engineering. Ph.D. thesis. University of Dortmund. Alemanha.

Fenet, S., e Hassas, S. 1998. An Ant based system for dynamic multiple criteria balancing. Proceedings of the first international workshop on ant colony optimization (Ants'98). Bruxelas. Bélgica.

Fenet, S., e Hassas, S. 2000. A.n.t.: a distributed network control framework based on mobile agents. In proceedings of the international icsc congress on intelligent systems and applications.

Ferreira, Fábio Santos, Monteiro, Glauber Duarte, e Teixeira, Otávio Noura. 2008. Colónia evolucionária de formigas: Uma proposta inicial aplicada ao problema do caixeiro viajante. *R*evista hífen, uruguaiana, vol.32, pp 286–292. Brazil.

Ford, L. R., e Fulkerson, D. R. 1962. *F*low in networks. Princeton University Press, Princeton.

Gai, Silvano. 1998. *I*nternetworking ipv6 with cisco routers. McGraw-Hill.

Gallego-Schmid, M. 1999. Modified AntNet: Software application in the evaluation and management of a telecommunication network. Genetic and evolutionary computation conference (GECCO-99). Orlando. Florida.

García, Salvador, Molina, Daniel, Lozano, Manuel, e Herrera, Francisco. 2008. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *S*pringer science and business media.

Garson, G. David. 2010. *f*rom statnotes: Topics in multivariate analysis. Acedido a 30/01/2010 em http://faculty.chass.ncsu.edu/garson/pa765/statnote.htm.

Glover, Fred. 1986. Future paths for integer programming and links to artificial intelligence. **13**, 533–549. Elsevier Science Ltd. Oxford. UK.

Greis, Marc. 2009. *T*utorial for the network simulator-ns. http://www.isi.edu/nsnam/ns/tutorial/, acedido em 14/01/2010.

Hayzelden, Alex L. G., e Bigham, John. 1999. *S*oftware agents for future communication systems. capitulo 13. Springer.

Heusse, M., Snyers, D., Guérin, S., e Kuntz, P. 1998. *A*daptive agent-driven routing and load balancing in communication networks. Advances in complex systems.

Heusse, M., Snyers, D., e Kermarrec, Y. 1999. *A*daptive agent driven routing in communication networks: comparison with a classical approach. Advances in complex systems.

Jain, P. 2002 (Junho). *V*alidation of AntNet as a superior single path, single constrained routing protocol. Msc. thesis, Department of Computer Science and Engineering, University of Minnesota. USA.

Kassabalidis, I., El-Sharkawi, M. A., II, R. J. Marks, Arabshahi, P., e Gray., A. A. 2002. Adaptive-sdr: Adaptive swarm-based distributed routing. in proceedings of ijcnn. pages 351–354. In proceedings of ijcnn. IEEE Press. Honolulu, HI . USA.

Kaufmann, Morgan. 2007. *N*etwork routing: Algorithms, protocols and architectures. Elsevier Inc.

Kebria, R. Vallaie, Aman, S. Saffari, Shamshirband, S. S., Shirgahi, H., Gholami, M., e Kia, B. 2009. The effect of AntNet parameters on its performance. Março. Vol.4 (3). pp 159–166. Scientific Research and Essay.

Kurose, James F., e Ross, Keith W. 2003. *C*omputer networking: A top-down approach featuring the Internet. Addison Wesley.

Lawler, E.L., Lenstra, J.K., Kan, A.H.G. Rinnooy, e Shmoys, D.B. 1985. *T*he traveling salesman problem. John Wiley e Sons.

Lawrence, Steve, e Giles, C Lee. 1998. *A*ntNet: An ACO algorithm for data network routing.

Lima, Richardson. 2009 (Novembro). *A*CO routing algorithm in practice. http://AntNet.wordpress.com; acedido em 05/11/09.

Malhotra, Ravi. 2002. *I*P routing. O'Reilly and Associates, Inc.

Malkin, G. 1998. *R*ip version 2. http://www.ietf.org/rfc/rfc2453.txt, acedido em 23/04/2010.

Mapisse, João. 2010. *A*lgoritmo ACO para encaminhamento em redes com fios. http://w3.ualg.pt/∼pcardoso/AntNetBW/.

McQuillan, John M., Richer, Isaac, e Rosen, Eric C. 1978. Arpanet routing algorithm improvements. **3803**(Abril).

Michalareas, T., e Sacks, L. 2001a. Link-state and ant-like algorithm behaviour for single-constrained routing. IEEE workshop on high performance switching and routing. Maio.

Michalareas, T., e Sacks, L. 2001b. *S*tigmergic techniques for solving multi-constraint routing for packet networks. Vol. 2094 of Lecture Notes in Computer Science. Springer-Verlag.

Moore, Edward F. 1959. *S*hortest path through a maze. Bell Telephone System.

Oida, K., e Kataoka, A. 1999. Lock-free AntNet and its evaluation adaptiveness. *J*ournal of IEICE b (in japanese).

Oida, K., e Sekido, M. 1999. An agent-based routing system for qos guarantees. In proceedings of the IEEE international conference on systems, man, and cybernetics. 833–838.

Oida, K., e Sekido, M. 2000. Ars: an efficient agent-based routing system for qos guarantees. computer communications. 1437–1447.

Ott, R. Lyman, e Longnecker, Micheal. 2008. *A*n introduction to statistical methods and data analysis.

Park, Kun I. 2005. *Q*os in packet networks. Springer.

Peterson, Larry L., e Davie, Bruce S. 2003. *C*omputer networks: A systems approach. Elsevier Science.

Pontes, Antonio Carlos Fonseca, e Corrente, José Eduardo. 2001. ComparaÇÕes múltiplas não-paramétricas para o delineamento com um fator de classificaÇão simples. *R*ev. mat. estat., são paulo.

Rajagopalan, Sundaram, e Shen, Chien-Chung. 2006. Ansi: A swarm intelligence-based unicast routing protocol for hybrid ad hoc networks. *E*lsevier.

Rekhter, Y. 1995. *A* border gateway protocol 4 (bgp-4). http://www.ietf.org/rfc/rfc1771.txt, acedido em 23/04/2010.

Rodriguez, Adolfo, Gatrell, John, Karas, John, e Peschke, Roland. 2001. *T*cp/ip tutorial and technical overview. IBM Corporation, International Technical Support Organization.

Sandalidis, H., Mavromoustakis, K., e Stavroulakis, P. 2001. Performance measures of an ant based decentralized routing scheme for circuit switching communication networks. soft computing. 313–317.

Sandalidis, H., Mavromoustakis, K., e Stavroulakis, P. 2004. Ant-based probabilistic routing with pheromone and antipheromone mechanisms. international journal of communication systems (ijcs). Janeiro, 55–62.

Schonderwoerd, R., Holland, O., Bruten, J., e Rothkrantz, L. 1996. *A*nt-based load balancing in telecommunications networks. adaptive behavior. Springer.

Sheskin, David J. 2003. *H*andbook of parametric and nonparametric statistical procedures. 3ª edição edn. CHAPMAN e HAL/CRC.

Spiegel, Murray R., Scheller, John, e Srinivasan, R. Alu. 2000. *P*robabilidade e estatística. 3ª edição edn. Colecção Schaum. McGraw-Hill.

Stevens, W., e Wright, G. 1995. *T*cp/ip illustrated: The implementation. Vol. 2. Addison Wesley.

Subing, Z., e Zemin, L. 2001. A qos routing algorithm based on ant algorithm. In proceedings of the IEEE international conference on communications (icc'01). 1587–1591.

Subramanian, D., Druschel, P., e Chen, J. 1997. *A*nts and reinforcement learning: A case study in routing in dynamic networks. In proceedings of ijcai. Morgan Kaufmann.

Tadrus, S., e Bai, L. 2003.. A qos network routing algorithm using multiple pheromone tables. *I*n web intelligence, 132–138.

Tadrus, S., e Bai, L. 2005. Qcolony: a multi-pheromone best-fit qos routing algorithm as an alternative to shortest-path routing algorithms. *I*nternational journal of computational intelligence and applications, 141–167.

Tanenbaum, Andrew S. 2003. *C*omputer networks. Prentice Hall PTR.

Thomas, Ivan John. 2006 (Janeiro). *D*esign, analysis and simulation of a distributed multiple criteria network routing method. Msc. thesis. Department of Electrical Engineering and Computer Science. Case Western Reserve University. Cleveland. USA.

van der Put, R. 1998. Routing in the faxfactory using mobile agents. *K*pn research.

White, T., Pagurek, B., e Oppacher, F. 1998a. Asga: improving the ant system by integration with genetic algorithms. In proceedings of the third genetic programming conference. 610–617.

White, T., Pagurek, B., e Oppacher, F. 1998b. Connection management using adaptive mobile agents. *P*age 802–809 of: Arabnia, In H.R. (ed), *P*roceedings of the pdpta. CSREA Press.

Zhong, W., e Evans, D. 2002. *W*hen Ants attack: Security issues for stigmergic systems. Tech. rept. Department of Computer Science, University of Virginina.

Z.Wang, e J.Crowcroft. 1996. Quality-of-service routing for supporting multimedia applications. *I*EEE.

APÊNDICE

A

O Network Simulator – 2 (ns-2)

O *ns-2* é um simulador de eventos discretos, de código fonte aberto, usado para simulações de LAN's, MAN's e WAN's sobre IP. Suporta os protocolos TCP e UDP na camada de transporte e FTP, *Telnet*[1] , Web, CBR e VBR na camada de aplicação; possui suporte para mecanismos de gestão de filas de espera (*DropTail (FIFO), RED, CBQ entre outros*); suporta os algoritmos de encaminhamentos *unicast e multicast*, alguns protocolos da camada MAC (para LAN's) e algoritmos que implementam o QoS, para redes com e sem fios [Chung & Claypool, 2009].

A estrutura de funcionamento do *ns-2* compreende o script *OTcl*, o interpretador *Tcl* juntamente com as bibliotecas do *ns-2* e os resultados da simulação (Figura A.1). Os códigos das simulações são escritos na linguagem *OTcl* (*OTcl script*), através de um programa de processamento de texto. Todos os ficheiros escritos nesta linguagem são gravados com a extensão *.tcl* e usam as bibliotecas do *ns-2* em conjunto com o interpretador *Tcl* para produzir os resultados da simulação.

[1] O *Telnet* é um protocolo cliente-servidor usado para permitir a comunicação entre computadores através do acesso remoto.

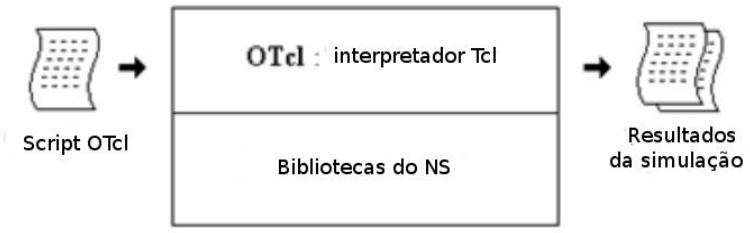


Figura A.1: Estrutura de funcionamento do *ns-2*. Nesta estrutura o script *OTcl* contém a codificação da rede feita com o recurso as funções existentes nas bibliotecas do *ns-2*. O interpretador permite produzir os resultados da simulação.

A.1 Princípios de simulação no *ns-2*

De uma forma genérica, a simulação no *ns-2.34* é feita de acordo com os seguintes passos [Coutinho, 2009]:

planeamento da simulação: O planeamento da simulação consiste em fazer um esboço do cenário de rede que se pretende. Isto dá uma visão macro do tipo de simulação que se pretende. A Figura A.2 apresenta um modelo de planeamento de uma rede com dois nós contendo toda a estrutura da simulação, ou seja, possui as camadas de aplicação e de transporte bem como a topologia de rede. A noção de tempo no *ns-2* é obtida através de unidades de simulação que podem ser associadas a segundos.

definição dos nós da rede: A definição dos nós da rede no *ns-2* é feita recorrendo à comandos *OTcl*. Os componentes do tipo nó usados no *ns-2* possuem a estrutura básica ilustrada na Figura A.3. No *ns-2* os nós representam classes em *OTcl* e os componentes dos nós são objectos *Tcl*. O nó unicast, situado à esquerda da Figura A.3, possui dois objectos *Tcl*, nomeadamente o porto (*Port Classifier*) e o endereço (*Addr Classifier*).

O Endereço permite encaminhar os pacotes que entram num nó, para a rede

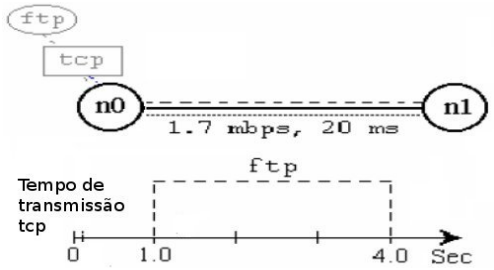


Figura A.2: Planeamento de uma rede com dois nós

de destino (*link*) ou para o agente de destino *(Null ou Sink Agent)* caso este nó seja um terminal. O nó *multicast*, situado a direita da Figura A.3, possui o classificador *multicast* e os replicadores além dos dois objectos *Tcl* do nó unicast. Os objectos de encaminhamento *unicast e multicast* são separados por meio de um *switch*. Numa simulação *multicast* os pacotes possuem um bit de classificação para pacotes que são enviados em *multicast* ou *unicast*. Se esse bit tomar o valor 0 então o pacote é *unicast* e se tomar o valor 1 então o pacote é *multicast*. A função do *switch* é de encaminhar os pacotes de acordo com o valor desse bit.

Para definir um nó *unicast* no *OTcl* usamos o comando (precedido do comando: *set ns [new Simulator]*):

```
set <nome do nó> [$ns node]
```

Para definir um nó multicast no *OTcl* usamos o comando (precedido do comando: *set ns [new Simulator -multicast on]*):

```
set <nome do nó> [$ns node]
```

definição dos tipos de ligação: A ligação é uma estrutura que permite que sejam efectuadas as comunicações entre dois ou mais pontos distintos. Existem três

tipos de ligações, nomeadamente as ligações *simplex* (Figura A.4) e *duplex* (*half-duplex* e *full-duplex*).

No *ns-2* a estrutura da ligação com fios está associada às políticas de gestão de filas de espera. Assim, o atraso no encaminhamento dos pacotes é composto pelo atraso proveniente da fila de espera (que depende do congestionamento da rede) e pelo atraso da própria ligação (definido pelo programador da rede a simular), dado em milissegundos.

Nas ligações com fios, a definição do tipo de ligação entre dois nós é feita no ficheiro *OTcl* recorrendo ao seguinte comando:

```
$ns <tipo de ligação> $<nó de origem> $<nó de destino>
  <largura de banda da ligação>  <atraso da ligação>
  <politica de gestão de filas de espera>
```

onde

<tipo de ligação>: deve ser *simplex-link*, *duplex-link* conforme o objectivo do problema;

<nó de origem> e <nó de destino>: devem conter os nomes dos nós de origem e de destino respectivamente;

<largura de banda da ligação> e <atraso da ligação>: especificam a largura de banda da ligação, definida em unidades de bytes e o atraso da ligação definido em milissegundos;

<politica de gestão de filas de espera>: parâmetro que especifica tipo de filas de espera que pretendemos na ligação. No *ns-2.34* os mais comuns são:

- DropTail – nesta política o primeiro pacote que entra na fila de espera de um nó, é o primeiro que a sair da mesma;

- *Fair Queuing* (FQ) – esta política escalona os fluxos de tráfego de pequena largura de banda em primeiro plano, partilhando a restante largura de banda pelos fluxos que requeiram elevada largura de banda;
- *Statistical Fair Queuing* (SFQ) – efectua uma distribuição da largura de banda estatisticamente;
- *Deficit Round Robin* (DRR) – efectua um prévio agendamento de prioridade para os diferentes fluxos.

O planeamento do tipo de ligação ajuda na avaliação dos pacotes que são descartados na rede.

definição do tráfego na rede: No *ns-2* o tráfego na rede é definido através do tipo de aplicação, associada a camada de aplicação, bem como o protocolo da camada de transporte que transportara essa aplicação. Alguns dos protocolos que o *ns-2* suporta na camada de transporte são:

- o UDP; e
- o TCP.

Os principais tipos de tráfego gerado pelo *ns-2* são:

- o CBR;
- o FTP; e
- o HTTP.

Os comandos de *OTcl* usados para definir o protocolo UDP na camada de transporte de um nó emissor são:

```
set udp [new Agent/UDP]
$ns attach-agent $<nome do nó> $udp
```

Podem ainda ser definidos outros parâmetros para os pacotes UDP através dos comandos abaixo:

```
$udp set packetSize_ <tamanho do pacote>
$udp set dst_addr_ <endereço>
$udp set dst_port_ <porto>
$udp set class_ <cor dos pacotes>
$udp set ttl_ <TTL>
```

Associado ao emissor devemos definir o nó receptor com as seguintes definições:

```
set null [new Agent/Null]
$ns attach-agent $<nome do nó> $null
```

O correcto encaminhamento de pacotes só é possível estabelecendo uma ligação entre o emissor e o receptor através do comando:

```
$ns connect $udp $null
```

Para o caso do protocolo TCP, a definição de um nó emissor é feita recorrendo aos comandos:

```
set tcp [new Agent/TCP]
$ns attach-agent $<nome do nó> $tcp
```

Ao contrário do UDP, no TCP podem ser definidos outros *agentes* para os nós emissores; tais como:

- Agent/TCP;
- Agent/TCP/Reno;
- Agent/TCP/Newreno;
- Agent/TCP/Sack1;

- Agent/TCP/Vegas; e
- Agent/TCP/Fack

Os outros parâmetros (opcionais) associados aos pacotes tcp são:

```
$tcp set window_ <tamanho máximo da janela>
$tcp set packetSize_ <tamanho do pacote a enviar[bytes]>
```

Associado ao emissor teremos o nó receptor:

```
set sink [new Agent/TCPSink]
$ns attach-agent $<nome do nó> $sink
```

A semelhança do emissor TCP existem outros *agentes* receptores TCP, tais como:

- Agent/TCPSink;
- Agent/TCPSink/DelAck;
- Agent/TCPSink/Sack1; e
- Agent/TCPSink/Sack1/DelAck

A Ligação entre o nó emissor e o receptor é feita da seguinte forma:

```
$ns connect $tcp $sink
```

No *ns-2* existe um agente TCP que é tanto emissor como receptor, definido da seguinte forma:

- como emissor:

  ```
  set no1 [new Agent/TCP/FullTcp];
  ```

- como receptor:

  ```
  set sink [new Agent/TCP/FullTcp];
  $ns attach-agent $<nó emissor> $no1;
  $ns attach-agent $<nó receptor> $sink;
  ```

Para definir o tráfego FTP da camada de aplicação de um dado nó, fazemo-lo recorrendo aos seguintes comandos:

```
set ftp [new Application/FTP]
$ftp attach-agent $<nome do nó>
$ftp set packetSize_ <tamanho do pacote>
ou
set ftp [$<nome do nó> attach-app FTP]
$ftp set packetSize_ <tamanho do pacote>
```

Para o tráfego CBR temos:

```
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ <tamanho do pacote em bytes>
$cbr set rate_ 64Kb
```

análise de resultados: A análise dos resultados de uma simulação é feita analisando o ficheiro de *trace*. Este ficheiro contém informações que permitem analisar o comportamento de uma dada rede, e varia consoante se trata de uma rede com ou sem fios (Secção A.3.2).

A.2 O *Network Animator* (NAM)

A ferramenta NAM permite-nos ver a simulação da rede num ambiente gráfico animado. Com o NAM pode-se observar o decurso da simulação, a transmissão dos fluxos, a formação de filas, as perdas de pacotes, entre outros (Figura A.5). Na parte superior do NAM encontramos os menus *Files*, *Views* e *Analysis*; botões de controlo da animação, o tempo da animação e a velocidade da animação dada em unidades de décimas de milissegundos. No lado esquerdo temos a barra de ferramentas. No centro temos o nosso cenário de rede onde é possível observar os nós, os fluxos, os congestionamentos bem como as filas de espera e na parte inferior encontra-se uma barra que assinala o decurso da simulação.

A.3 Simulação de redes com fios no *ns-2.34*

No *ns-2* as configurações para a simulação de redes com fios apresenta uma estrutura diferente da para sem fios, devido aos meios de transmissão e as definições que cada nó deve comportar. Assim sendo a estrutura do *OTcl* e do *trace* (que serão estudadas nas secções seguites) apresentam diferenças nos dois tipos de redes, Tendo em conta o teor do nosso trabalho, apenas estudaremos as estruturas correspondentes as redes com fios.

A.3.1 O Ficheiro *OTcl*

No *ns-2*, toda a configuração de rede é feita em linguagem *OTcl*, num programa de processamento de texto e gravada com a extensão *.tcl*, ficheiro este que é interpretado pelo interpretador de *Tcl* produzindo resultados que são usados para visualizar a animação da rede no *Network Animator - NAM* e ou analisar o comportamento da rede (Figura A.6).

Para melhor explicar o processo suponhamos que se pretende simular uma rede com fios com 5 nós e 4 ligações do tipo duplex com larguras de banda de $5Mbps$ e $2.5Mbps$ e atrasos de $10ms$ (Figura A.7). O ficheiro script *OTcl* terá por exemplo, a estrutura seguinte:

- começamos por criar o objecto de simulação:

  ```
  set ns [new Simulator]
  ```

- definimos as cores que serão à frente usadas para distinguir os diferentes fluxos:

  ```
  $ns color 1 Blue
  $ns color 2 Red
  $ns color 3 Green
  ```

- definimos dois ficheiros, um para o *nam* e o outro para o *trace* da simulação:

```
set nf [open out.nam w]
$ns nam\textit{trace}-all $nf
set tf [open out.tr w]
$ns \textit{trace}-all $tf
```

- criamos o procedimento `finalizar` que irá fecha a escrita nos ficheiros criados:

```
proc finalizar {} {
     global ns nf
     $ns flush-\textit{trace}
     close $nf
     close $tf
     exit 0
}
```

- criamos os 4 nós da rede:

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
```

- criamos as ligações entre os nós, especificando a largura de banda de cada ligação, o atraso e o tipo de politica de gestão de filas de espera:

```
$ns duplex-link $n0 $n3 5Mb 10ms DropTail
$ns duplex-link $n1 $n3 5Mb 10ms DropTail
$ns duplex-link $n2 $n3 5Mb 10ms DropTail
$ns duplex-link $n3 $n4 2.5Mb 10ms DropTail
```

- definimos o tempo de espera de cada pacote na fila de espera da ligação entre os nós 3 e 4:

```
$ns duplex-link-op $n3 $n4 queuePos 0.3
```

- definimos as posições relativas dos nós na animação, especificando as orientações das ligações:

```
$ns duplex-link-op $n0 $n3 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n2 $n3 orient right-up
$ns duplex-link-op $n3 $n4 orient right
```

- definimos o protocolo da camada de transporte nos nós 0, 1 e 2:

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
$udp0 set class_ 1

set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
$udp1 set class_ 2

set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n2 $tcp
$tcp set class_ 3
```

- definimos o protocolo da camada de aplicação (tráfego da rede). Para o tráfego CBR definimos o tamanho do pacote (*packetSize_*) bem como o intervalo de envio (*interval_*). Estes dois podem ser substituídos pelo ritmo de transmissão (*rate_*), em unidades de bytes por segundo:

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005

set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set rate_ 2Mb

set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

- definimos o nó 4 como o receptor de todos os tráfegos e criamos nele *agentes* para receber os pacotes que lhe chegam:

```
set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
```

```
set null1 [new Agent/Null]
$ns attach-agent $n4 $null1

set sink [new Agent/TCPSink]\newline
$ns attach-agent $n4 $sink
```

- conectamos os emissores ao receptor de moda a que haja a troca de informações:

```
$ns connect $udp0 $null1
$ns connect $udp1 $null1
$ns connect $tcp $sink
```

- definimos os tempos de envio dos diferentes fluxos (tráfegos):

```
$ns at 0.2 "$ftp start"
$ns at 1.5 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 10.0 "$cbr0 stop"
$ns at 13.0 "$cbr1 stop"
$ns at 14.0 "$cbr0 start"
$ns at 20.0 "$cbr1 start"
$ns at 40.0 "$cbr1 stop"
$ns at 40.0 "$cbr0 stop"
$ns at 48.0 "$ftp stop"
```

- chamamos o procedimento "finalizar":

```
$ns at 50.0 "finalizar"
```

- definimos o ponto em que o interpretador efectua a simulação:

```
$ns run
```

Os comandos acima são gravados num o ficheiro com a extensão *.tcl*. O simulador *ns-2* é então executado recebendo este ficheiro como argumento. Como resultado da execução tipicamente são criados dois ficheiros definidos no início do scripy de configuração, um que é usado para a animação (com a extensão *.nam*) e outro usado para a análise dos resultados da simulação da rede (com a extensão *.tr*).

A.3.2 O trace NAM

O ficheiro *out.nam* criado através da descrição da secção anterior, é usado para visualizar a animação. O comando que permite a execução do NAM é: ***exec nam out.nam*** e é inserido no procedimento `finalizar` da seguinte forma:

```
proc finalizar {} {
     global ns nf
     $ns flush-\textit{trace}
     close $nf
     close $tf
     exec nam out.nam &
     exit 0
}
```

O ficheiro de *trace* out.tr em geral é usado para analisar os resultados da simulação. Nele encontramos toda a informação dos pacotes durante a simulação. O ficheiro de *trace* de uma rede com fios apresenta o seguinte formato:

Tabela A.1: Formato do *trace* das redes com fios

Evento	Tempo	Nó de proveniência	Nó destino	Tipo de pacote	Tamanho do pacote	Flags	Id fluxo	Endereço de origem	Endereço de destino	Num seq	Id pacote

Evento: pode possuir os seguintes valores: + (pacote entra na fila de espera), - (pacote sai da fila de espera), r (pacote recebido), d (pacote descartado);

tempo: indica o instante de ocorrência do evento;

nó de proveniência: representa o nó onde ocorreu o evento;

nó destino: representa o nó para onde seguiu o pacote correspondente ao evento;

tipo de pacote: corresponde ao pacote que é transportado na rede (exemplo.:cbr, tcp, ...);

tamanho do pacote: corresponde ao tamanho em bytes do pacote que é transportado na rede;

flags: estão relacionadas com notificações antecipadas de congestionamento;

id fluxo: corresponde a um número que identifica o fluxo transportado na rede (por exemplo: udp0 pode possuir fid_1 e o tcp pode possuir fid_3);

endereço de origem: representa o endereço do nó emissor;

endereço destino: representa o endereço do nó destinatário;

num_seq: é o número de sequência de um dado pacote. Este número surge quando um determinado fluxo é subdivididos em pacotes mais pequenos para ser transportado na rede devido à pouca largura de banda disponibilizada. O número de sequência identifica o fragmento do pacote. Quando o receptor recebe cada um dos fragmentos de um dado pacote subdividido, este volta a agrupa-los de acordo com a sequência. No caso dos pacotes TCP, se o receptor não receber uma dada sequência de um dado pacote subdividido, este envia uma mensagem ao emissor pedindo que o envie novamente;

id pacote: identifica o pacote. Se considerarmos que um dado pacote foi subdividido em dois, cada um dos fragmentos possuirá o mesmo id pacote e diferentes números de sequência (por exemplo o primeiro fragmento terá num_seq=1 e o segundo num_seq=2).

Um exemplo do ficheiro de *trace* gerado nas simulações, em redes com fios é apresentado a seguir, onde encontramos pacotes TCP, UDP (CBR) e ACK para os

diversos eventos:

```
r  11.984288  3  4  cbr  500  -------  2  1.0  4.0  1982  7153
r  11.984976  2  3  tcp  1040 -------  3  2.0  4.1  1749  7191
+  11.984976  3  4  tcp  1040 -------  3  2.0  4.1  1749  7191
+  11.985     1  3  cbr  500  -------  2  1.0  4.0  1997  7198
-  11.985     1  3  cbr  500  -------  2  1.0  4.0  1997  7198
r  11.9858    1  3  cbr  500  -------  2  1.0  4.0  1995  7192
+  11.9858    3  4  cbr  500  -------  2  1.0  4.0  1995  7192
-  11.985872  3  4  cbr  500  -------  2  1.0  4.0  1984  7159
-  11.987472  3  4  tcp  1040 -------  3  2.0  4.1  1739  7161
r  11.987616  3  4  tcp  1040 -------  3  2.0  4.1  1736  7152
+  11.987616  4  3  ack  40   -------  3  4.1  2.0  1736  7199
-  11.987616  4  3  ack  40   -------  3  4.1  2.0  1736  7199
r  11.987888  4  3  ack  40   -------  3  4.1  2.0  1734  7193
+  11.987888  3  2  ack  40   -------  3  4.1  2.0  1734  7193
-  11.987888  3  2  ack  40   -------  3  4.1  2.0  1734  7193
r  11.988096  3  2  ack  40   -------  3  4.1  2.0  1732  7187
+  11.988096  2  3  tcp  1040 -------  3  2.0  4.1  1752  7200
-  11.988096  2  3  tcp  1040 -------  3  2.0  4.1  1752  7200
r  11.989216  3  4  cbr  500  -------  2  1.0  4.0  1983  7156
r  11.989904  2  3  tcp  1040 -------  3  2.0  4.1  1750  7194
+  11.989904  3  4  tcp  1040 -------  3  2.0  4.1  1750  7194
+  11.99      1  3  cbr  500  -------  2  1.0  4.0  1998  7201
-  11.99      1  3  cbr  500  -------  2  1.0  4.0  1998  7201
-  11.9908    3  4  cbr  500  -------  2  1.0  4.0  1985  7162
r  11.9908    1  3  cbr  500  -------  2  1.0  4.0  1996  7195
+  11.9908    3  4  cbr  500  -------  2  1.0  4.0  1996  7195
```

A.3.3 Comandos usados para a configuração do *AntNet* no *ns-2.34*

O *ns-2* não possui a implementação do protocolo *AntNet*. Este protocolo é a base do nosso trabalho e foi necessário efectuar tal implementação, de acordo com as instruções constantes no manual de implementação do autor Lima [2009].

Os comandos que se seguem são utilizados para configurar uma rede que utiliza o *AntNet* no *ns-2*. A configuração do *AntNet* é feita no script *OTcl* de acordo com os seguites comandos:

- criação dos *agentes*[2] *AntNet*:

```
for {set i 0} {$i < 12 } {incr i} {
set   formiga($i)  [ new Agent/AntNet $i]
}
```

- ligação dos *agentes AntNet* a cada nó criado:

```
for {set i 0} {$i < 12} {incr i} {
$ns attach-agent  $no($i)  $formiga($i)
}
```

- criação das ligações entre os nós *AntNet* (em conformidade com a topologia de rede):

```
$ns connect $ formiga (0) $ formiga (5)
$ns connect $ formiga (5) $ formiga (0)
$ns connect $ formiga (1) $ formiga (2)
$ns connect $ formiga (2) $ formiga (1)
$ns connect $ formiga (2) $ formiga (3)
$ns connect $ formiga (3) $ formiga (2)
$ns connect $ formiga (2) $ formiga (5)
$ns connect $ formiga (5) $ formiga (2)
$ns connect $ formiga (2) $ formiga (7)
$ns connect $ formiga (7) $ formiga (2)
$ns connect $ formiga (3) $ formiga (6)
$ns connect $ formiga (6) $ formiga (3)
$ns connect $ formiga (3) $ formiga (7)
$ns connect $ formiga (7) $ formiga (3)
$ns connect $ formiga (4) $ formiga (5)
$ns connect $ formiga (5) $ formiga (4)
$ns connect $ formiga (5) $ formiga (6)
$ns connect $ formiga (6) $ formiga (5)
$ns connect $ formiga (5) $ formiga (8)
$ns connect $ formiga (8) $ formiga (5)
$ns connect $ formiga (5) $ formiga (9)
$ns connect $ formiga (9) $ formiga (5)
$ns connect $ formiga (6) $ formiga (7)
$ns connect $ formiga (7) $ formiga (6)
$ns connect $ formiga (6) $ formiga (9)
$ns connect $ formiga (9) $ formiga (6)
```

[2] *Agentes* ou formigas são pacotes de dados com as características do algoritmo *AntNet*.

```
$ns connect $ formiga (6) $ formiga (10)
$ns connect $ formiga (10) $ formiga (6)
$ns connect $ formiga (7) $ formiga (11)
$ns connect $ formiga (11) $ formiga (7)
$ns connect $ formiga (9) $ formiga (10)
$ns connect $ formiga (10) $ formiga (9)
$ns connect $ formiga (10) $ formiga (11)
$ns connect $ formiga (11) $ formiga (10)
```

- adicionamos os nós vizinhos (em conformidade com a topologia de rede):

```
$ns at now "$ formiga (0) add-neighbor $no(0) $no(5)"
$ns at now "$ formiga (0) add-neighbor $no(1) $no(2)"
$ns at now "$ formiga (0) add-neighbor $no(2) $no(3)"
$ns at now "$ formiga (0) add-neighbor $no(2) $no(5)"
$ns at now "$ formiga (0) add-neighbor $no(2) $no(7)"
$ns at now "$ formiga (0) add-neighbor $no(3) $no(6)"
$ns at now "$ formiga (0) add-neighbor $no(3) $no(7)"
$ns at now "$ formiga (0) add-neighbor $no(4) $no(5)"
$ns at now "$ formiga (0) add-neighbor $no(5) $no(6)"
$ns at now "$ formiga (0) add-neighbor $no(5) $no(8)"
$ns at now "$ formiga (0) add-neighbor $no(5) $no(9)"
$ns at now "$ formiga (0) add-neighbor $no(6) $no(7)"
$ns at now "$ formiga (0) add-neighbor $no(6) $no(9)"
$ns at now "$ formiga (0) add-neighbor $no(6) $no(10)"
$ns at now "$ formiga (0) add-neighbor $no(7) $no(11)"
$ns at now "$ formiga (0) add-neighbor $no(9) $no(10)"
$ns at now "$ formiga (0) add-neighbor $no(10) $no(11)"
```

- definimos os parâmetros dos *agentes AntNet*:

```
for {set i 0} {$i < 12} {incr i} {
$ formiga ($i) set num_nodes_ 12
$ formiga ($i) set timer_ant_ 0.03
$ formiga ($i) set r_factor_ 0.05
$ns  at  1.0  "$ formiga ($i) start"
}
```

Nota:

***num_nodes_*:** corresponde ao número de nós *AntNet* da rede;

***timer_ant_*:** corresponde ao intervalo de tempo (Δt) em que são lançados os *agentes*;

***r_factor_*:** corresponde ao factor r referente a actualização da tabela de feromonas (expressões 3.12 e 3.13);

$ns at 1.0 "$formiga ($i) start": corresponde ao tempo inicial de lançamento do primeiro agente.

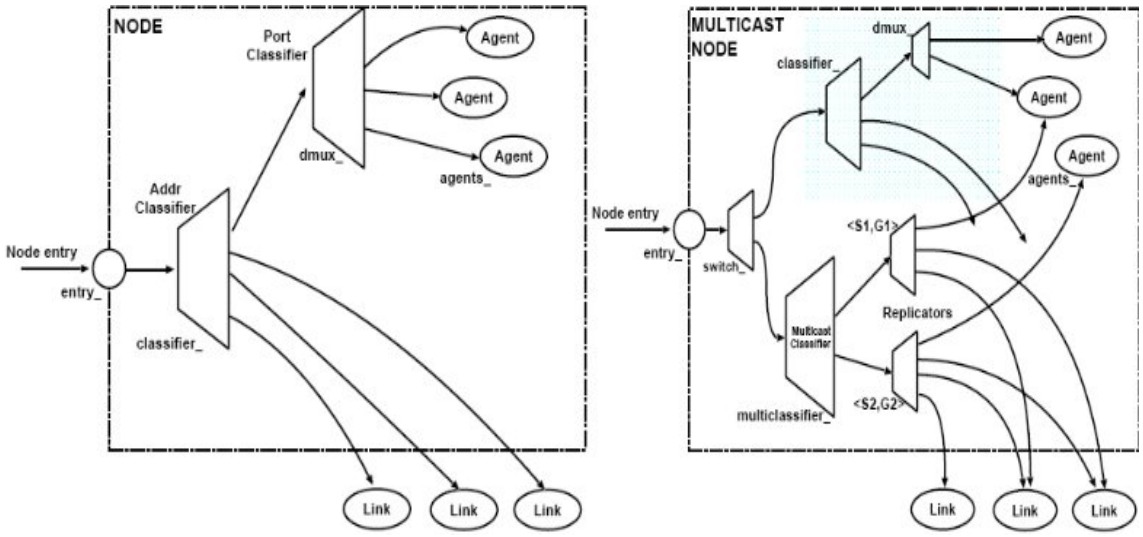


Figura A.3: Estrutura básica dos componentes do tipo nó unicast (à esquerda) e do tipo nó multicast (à direita) no *ns-2*

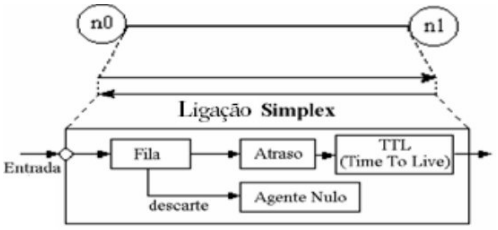


Figura A.4: Estrutura de uma ligação *simplex* entre dois nós no *ns-2*

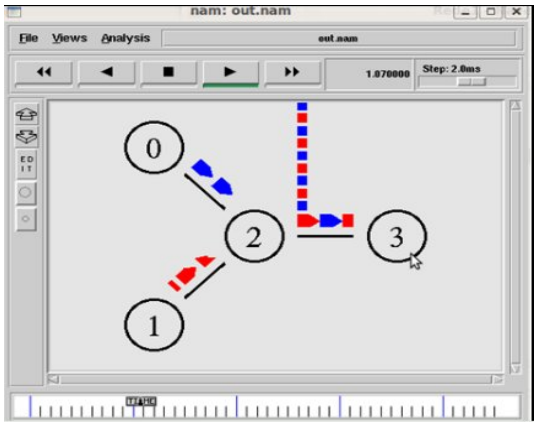


Figura A.5: Diferentes fluxos (côr azul que parte do nó 0 e côr vermelha que parte do nó 1), decurso da simulação (parte inferior), menus, botões de controlo da simulação, tempo e velocidade (em segundos) da simulação (situados na parte superior da Figura) e barra de ferramentas do *nam* (lado esquerdo), da rede com 4 nós

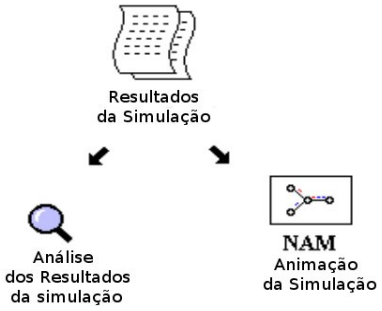


Figura A.6: Resultados de uma simulação.

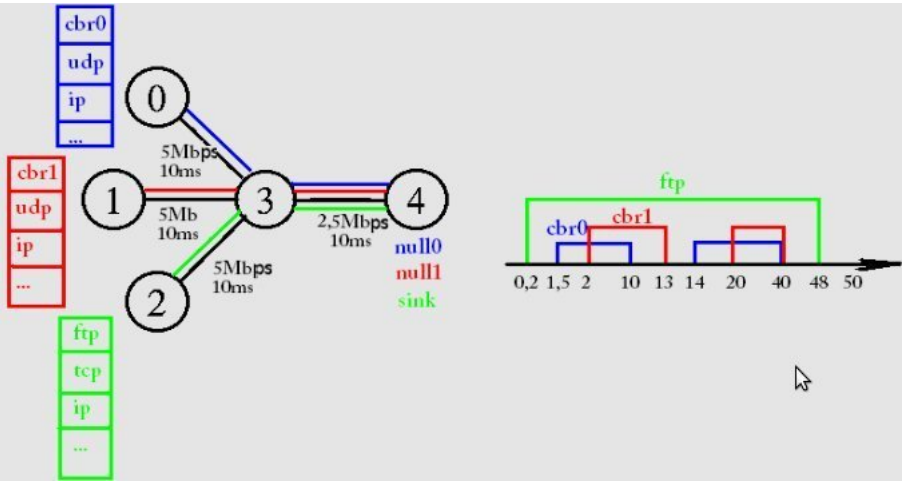


Figura A.7: Topologia de uma rede com fios com cinco nós e quatro ligações *duplex* com uma largura de banda de $5Mbps$ cada com um atraso de $50ms$.

APÊNDICE

B
Testes de Hipóteses

Neste capítulo vamos apresentar os testes de hipóteses paramétricos utilizados para a comparação dos algoritmos de encaminhamentos analisados nesta dissertação. A escolha do teste paramétrico escolhido (que veremos a seguir), derivou do facto de termos trabalhado com amostras de dimensão inferior a 30, de termos utilizado como parâmetro de comparação as médias estatísticas bem como pelo facto de não haver dependência entre os resultados das simulações dos algoritmos estudados.

B.1 Introdução

De uma forma geral, uma hipótese é uma suposição sobre um fenómenos ou conjuntos de factos. Na inferência estatística a hipótese é uma suposição sobre um ou mais parâmetros de uma amostra populacional (média, variância e proporção populacional), ou seja, testar hipóteses, requer determinar a importância da diferença entre um valor observado de uma amostra populacional e o da suposição.

A realização de um teste de hipóteses é feito de forma a garantir um compromisso entre o erro (que deve ser mínimo) e a decisão (que deve ser correcta). A hipótese que se pretende testar denomina-se hipótese nula - H_0, sendo H_1 a hipótese alternativa.

Tratando-se de um teste paramétrico, a hipótese nula baseia-se no pressuposto que a amostra segue uma determinada lei de probabilidades, que explica a distribuição das frequências na população de onde essa amostra foi retirada [Sheskin, 2003]. Existem dois tipos de hipóteses, nomeadamente a bilateral e a unilateral (à esquerda ou à direita). Abaixo estão ilustrados alguns dos testes de hipóteses que podemos encontrar em alguns estudos estatísticos:

a) H_0: μ =parâmetro **versus** H_1: $\mu \neq$ parâmetro $\longmapsto$ teste bilateral

b) H_0: μ =parâmetro **versus** H_1: $\mu <$ parâmetro $\longmapsto$ teste unilateral à

esquerda

c) H_0: μ =parâmetro **versus** H_1: $\mu >$ parâmetro $\longmapsto$ teste unilateral à direita

d) H_0: $\mu \leq$parâmetro **versus** H_1: $\mu >$ parâmetro $\longmapsto$ teste unilateral à direita

e) H_0: $\mu >$parâmetro **versus** H_1: $\mu \leq$ parâmetro $\longmapsto$ teste unilateral à esquerda

f) H_0: $X \sim$ Normal **versus** H_1: $X \nsim$ Normal $\longmapsto$ teste de ajustamento

Nos testes de hipóteses, existe a probabilidade de ser cometido erro sobre a decisão tomada. Essa probabilidade é denominada de **nível de significância -** γ, e pode também ser definida como a probabilidade de se rejeitar a hipótese nula, quando ela é verdadeira. Existem dois tipos de erro possíveis de serem cometidos num teste de hipótese, nomeadamente o erro do tipo I e o do tipo II.

Erro do tipo I relaciona-se com o facto de se rejeitar a hipótese nula, H_0, quando ela é verdadeira, e o erro do tipo II relaciona-se com o facto de aceitar a hipótese nula, H_0, quando ela é falsa. Nota-se que a probabilidade do erro do tipo I é igual ao nível de significância. A Tabela B.1 mostra as possibilidades existentes nos testes de hipóteses.

Definimos o *p-value* como a probabilidade de obter um valor da estatística de teste que serve de comparação com o nível de significância γ, ajudando na tomada de decisões. Assim se *p-value*$> \gamma$ então aceita-se H_0 como verdadeira e se *p-value*$< \gamma$ então não se aceita H_0 como verdadeira.

Tabela B.1: Possibilidades existentes nos testes de hipóteses.

Realidade \ **Decisão**	**Aceitar H_0**	**Rejeitar H_0**
H_0 é verdadeira	**Decisão correta** 1 - α = P(Aceitar H_0 / H_0 é V) = P(H_0 / H_0)	**Erro do Tipo I** **α** = P(Erro do tipo I) = P(Rejeitar H_0 / H_0 é V) = Nível de significância do teste = P(H_1 / H_0)
H_0 é falsa	**Erro do Tipo II** **β** = P(Erro do tipo II) = = P(Aceitar H_0 / H_0 é falsa) = P(Aceitar H_0 /H_1 é V) = P(H_0 /H_1)	**Decisão correta** **1 - β** = P(Rejeitar H_0 / H_0 é falsa) = P(H_1 / H_1) = Poder do teste.

B.2 Testes estatísticos paramétricos

Um teste paramétrico é um tipo de teste de hipóteses que é aplicado mediante a verificação das seguintes propriedades:

independência: em caso de testes de duas ou mais parâmetros estatísticos é necessário que estes sejam independentes. Por independentes entende-se que a ocorrência de um dos parâmetros não modifica a probabilidade do outro ocorrer;

normalidade: a normalidade dos parâmetros estatísticos esta ligada ao facto dos mesmo seguirem uma distribuição Normal ou Gaussiana com um determinado valor da média - μ e da variância σ. Alguns dos testes de normalidade existentes e que serão estudados adiante são [García *e*t al. , 2008]:

- o Kolmogorov-Smirnov: compara a distribuição cumulativa dos dados observados com a distribuição cumulativa Gaussiana, obtendo o *p-value* a partir da discrepância das duas distribuições;
- *Shapiro-Wilk*: analisa os dados observados para determinar o nível de simetria e kurtosis (*Shape of the curve*) de modo a determinar a diferença entre estes e a distribuição Gaussiana, obtendo o *p-value* através da soma dos quadrados das discrepâncias;

homogeneidade: indica a existência de uma violação da hipótese de igualdade da variância. A verificação da homogeneidade é feita aplicado o teste de Levene (o exemplo deste teste pode ser encontrado na secção B.2.2).

O estudo dos testes paramétricos pode ser realizado sobre:

- uma amostra;
- duas amostras independentes;
- duas amostras emparelhadas (dependentes);
- várias amostras (análise de variância).

Alguns dos testes paramétricos mais utilizados são: o Teste Binomial; o Teste t e o Teste da Curva Normal [Garson, 2010]. Neste trabalho incidiremos o estudo apenas no teste paramétrico t.

Qualquer teste de hipóteses paramétrico segue os seguintes passos:

1º Passo: formulação das hipóteses - Nesta fase são definidas as hipóteses nula e alternativa. Exemplo: considerando uma amostra populacional da(s) variável(eis) $\mathbf{X}$. Definimos a hipótese $H_0 : \theta = \theta_0$ (sendo θ o parâmetro populacional). Tendo definido a hipótese nula é conveniente determinar qual será a alternativa caso a hipótese nula seja rejeitada. podemos definir a hipótese alternativas mediante uma das seguintes opções: $H_1 : \theta = \theta_2$; $\theta > \theta_0$; $\theta < \theta_0$ ou $\theta \neq \theta_0$ (ver a Secção B.1);

2º Passo: definição do estimador - definir a variável e a distribuição teórica a ela associada;

3º Passo: definição o nível de significância - definir a probabilidade de se cometer o erro do tipo I. Com esta definição é possível determinar o valor crítico, que é lido na tabela da distribuição teórica estatística da variável estatística. Este valor vai separar a região crítica (de rejeição) da região de aceitação;

4º Passo: definição da estatística para a hipótese nula - a estatística do teste, permite-nos calcular o valor da estimativa, que servirá para concluir acerca das hipóteses;

5º Passo: decisão - rejeita-se a hipótese nula se o valor da estimativa estiver na região crítica e aceita-se caso contrário;

6º Passo: conclui-se

B.2.1 *Teste t de Student* no SPSS para uma amostra da média populacional

Considerando uma amostra de uma população com os seguintes elementos:

```
75 60 55 80 52 90 60 91 72 58 77 80 66 40 62
```

Pretendemos testar a hipótese da média ser igual é 72, com um nível de significância (γ) de 5%. Portanto formulamos a nossa hipótese:
$H_0 : \mu = 72$
$H_1 : \mu \neq 72$ (Teste bilateral)
Para aplicar este tipo de teste no SPSS (versão 17.0) escolhemos a opção Teste t para uma amostra (*One-Sample T Test*), como ilustrado na Figura B.1, e definindo o parâmetro de teste com o valor de 72 (*Test Value*) (Figura B.2).

Os resultados do teste podem ser visualizados nas Tabelas B.2 e B.3.

Tabela B.2: Estudo estatístico para uma amostra.

	N	Média	Desvio Padrão	Erro Médio Padrão
Amostra	15	67,87	14,451	3,731

Interpretação, Decisão e Conclusão Dado tratar-se de um teste bilateral, fazemos uma comparação directa entre o valor de *p-value*=0.287 com o valor de γ=0.05 (nível de significância). Como *p-value*=0.287 $>$ 0.05 então não se rejeita H_0, ou seja, as evidências estatísticas encontradas não nos permitem rejeitar a hipótese de que a média é de 72 com um nível de significancia, $\gamma = 0.05$.

Num teste unilateral teríamos que comparar o valor $\frac{\textit{p-value}}{2}$ com o valor de γ.

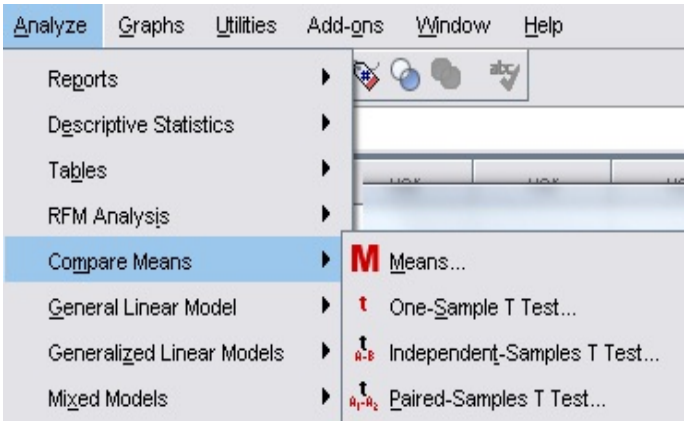


Figura B.1: Acesso ao teste *t de Student* no SPSS (versão 17.0).

Figura B.2: Definição do parâmetro de teste no SPSS (versão 17.0).

Tabela B.3: Teste *t* de *student* para uma amostra.

	Média de teste = 72					
					95% Confidence Interval of the Difference	
	t	df	p-value	Diferença Média	Lower	Upper
Amostra	-1,108	14	,287	-4,133	-12,14	3,87

B.2.2 *Teste t de Student* no SPSS para amostras independentes, com diferentes médias populacionais

Considerando dois grupos de amostras populacionais com os seguintes elementos:

```
Grupo A (Variável X): 2 5 4 6 8  9  7 5  6 5
Grupo B (Variável Y): 9 3 8 7 10 11 9 11 7 8
```

Pretendemos testar se existem diferenças significativas entre as médias dos dois grupos a um nível de significância (γ) de 5%, sabendo que a variável em estudo segue uma distribuição normal.
Portanto formulamos a nossa hipótese:
$H_0 : \mu_1 = \mu_2$ ou $\mu_1 - \mu_2 = 0$
$H_1 : \mu_1 \neq \mu_2$ ou $\mu_1 - \mu_2 \neq 0$ (Teste bilateral)

Para aplicar este tipo de teste no SPSS escolhemos a opção teste t para amostras independentes (*Independent-Samples T Test*), (Figura B.1), e definindo os parâmetros de entrada do teste (Figura B.3).

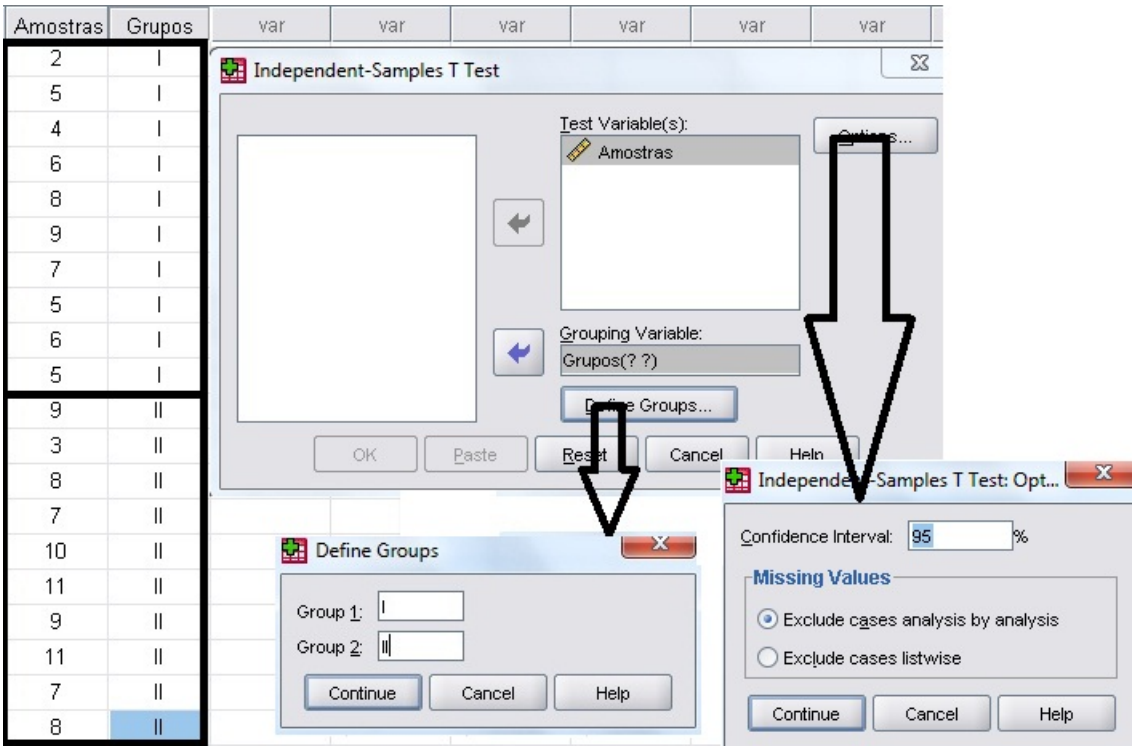


Figura B.3: Definição dos parâmetros de entrada do teste t para amostras independentes com diferentes médias populacionais no SPSS (versão 17.0).

Os resultados deste teste podem ser visualizados nas Tabelas B.4 e B.5.

Interpretação, Decisão e Conclusão A Tabela B.4 apresenta as medidas descritivas dos dados (μ, σ e erro-padrão amostral). Tabela B.5 apresenta o teste de homogeneidade (igualdade) de Levene (com $H_0 : \sigma_1^2 = \sigma_2^2$ e $H_1 : \sigma_1^2 \neq \sigma_2^2$) para as variâncias e teste t para a comparação de duas médias no caso de duas amostras independentes. Assim, podemos tirar as seguintes conclusões:

Tabela B.4: Estudo Estatístico dos Grupos.

	Grupos	N	Mean	Std. Deviation	Std. Error Mean
Amostras	I	10	5,70	2,003	,633
	II	10	8,30	2,359	,746

Tabela B.5: Teste t de Student para os Grupos.

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	p-value	Mean Difference	Std. Error Difference	Lower	Upper
Amostras	Equal variances assumed	,103	,751	-2,657	18	,016	-2,600	,979	-4,656	-,544
	Equal variances not assumed			-2,657	17,537	,016	-2,600	,979	-4,660	-,540

- para o teste de Levene temos o *p-value*=0.751, que significa que não se rejeita a hipótese de que as variâncias sejam iguais (apesar de desconhecidas);
- para o teste-t de Student (para a igualdade das médias) temos o *p-value*=0.016. Uma vez que se trata de um teste bilateral compara-se directamente *p-value*=0.016 com γ=0.05. Como *p-value*=0.016 $\leq$ 0.05 rejeita-se H_0, e podemos afirmar com 95% de confiança que existem diferenças significativas entre as duas médias.

B.2.3 Testes não-paramétricos de ajustamento específicos para distribuição normal no SPSS

Os teste de ajustamento são testes não-paramétricos utilizados para averiguar se uma dada amostra pode ser considerada como sendo proveniente de uma certa distribuição.

Testes de Kolmogorov-Smirnov

A aplicação do teste de Kolmogorov-Smirnov requer que a amostra possua uma distribuição contínua, e os parâmetros da distribuição são pré-especificados, pois não devem ser estimados a partir da amostra. No SPSS, este teste aplica-se recorrendo ao menú ***Analyze / Nonparametric Tests / 1 Sample KS*** e permite testar apenas 4 distribuições, entre as quais se inclui a Normal e a de Poisson (neste caso viola-se o pressuposto de continuidade da distribuição em teste).

Testes de *Shapiro-Wilk*

Shapiro e Wilk propuseram um teste de ajustamento específico para a distribuição Normal, que tem uma melhor performance que o teste anterior em amostras reduzidas ($n < 30$).

Estes testes estão disponíveis no SPSS através do menú ***Analyze / Descriptive Statistics /Explore***; onde seleccionamos o botão *Charts* e escolhemos a opção ***Normality Tests with Plots***.

APÊNDICE

C

Análise estatística aplicando o teste t para a média de pacotes perdidos nas redes

Tabela C.1: Resultados do teste estatístico t para as médias percentuais das perdas de pacotes do *TRÁFEGO I* da *REDE 1A*, realizados para os Algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw* com um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$. A probabilidade - p_{value} corresponde ao valor da estatística de teste que serve de comparar com o γ.

$\gamma = 0.05$			
Algoritmos	Testes de Hipóteses	p_{value}	Observações
AntNet ($\mu = 8.6579$) vs *Link-State* ($\mu = 13.2000$)	$H_0 : \mu_{AntNet} = \mu_{Link\text{-}State}$ $H_1 : \mu_{AntNet} \neq \mu_{Link\text{-}State}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 8.6579$) vs *AntNetBw* ($\mu = 7.1233$)	$H_0 : \mu_{AntNet} = \mu_{AntNetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{AntNetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 8.6579$) vs ϵ-*DANTENet* ($\mu = 8.6141$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENet}$	0.778	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 8.6579$) vs ϵ-*DANTENetBw* ($\mu = 7.3077$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 8.6579$) vs *CR-DANTENet* ($\mu = 8.7819$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENet}$	0.500	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 8.6579$) vs *CR-DANTENetBw* ($\mu = 7.0997$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 7.1233$) vs ϵ-*DANTENet* ($\mu = 8.6141$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 7.1233$) vs ϵ-*DANTENetBw* ($\mu = 7.3077$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENetBw}$	0.657	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 7.1233$) vs *CR-DANTENet* ($\mu = 8.7819$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 7.1233$) vs *CR-DANTENetBw* ($\mu = 7.0997$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENetBw}$	0.947	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENet* ($\mu = 8.6141$) vs ϵ-*DANTENetBw* ($\mu = 7.3077$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENet* ($\mu = 8.6141$) vs *CR-DANTENet* ($\mu = 8.7819$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENet}$	0.258	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENet* ($\mu = 8.6141$) vs *CR-DANTENetBw* ($\mu = 7.0997$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENetBw* ($\mu = 7.3077$) vs *CR-DANTENet* ($\mu = 8.7819$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENetBw* ($\mu = 7.3077$) vs *CR-DANTENetBw* ($\mu = 7.0997$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENetBw}$	0.554	$\gamma < p_{value}$ Insignificante
CR-DANTENet ($\mu = 8.7819$) vs *CR-DANTENetBw* ($\mu = 7.0997$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante

Tabela C.2: Resultados do teste estatístico t para as médias percentuais das perdas de pacotes do *TRÁFEGO II* da *REDE 1A*, realizados para os Algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw* com um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$. A probabilidade - p_{value} corresponde ao valor da estatística de teste que serve de comparar com o γ.

$\gamma = 0.05$			
Algoritmos	Testes de Hipóteses	p_{value}	Observações
AntNet ($\mu = 13.8220$) vs *Link-State* ($\mu = 19.9996$)	$H_0 : \mu_{AntNet} = \mu_{Link\text{-}State}$ $H_1 : \mu_{AntNet} \neq \mu_{Link\text{-}State}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 13.8220$) vs *AntNetBw* ($\mu = 12.5171$)	$H_0 : \mu_{AntNet} = \mu_{AntNetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{AntNetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 13.8220$) vs ϵ-*DANTENet* ($\mu = 13.7803$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENet}$	0.771	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 13.8220$) vs ϵ-*DANTENetBw* ($\mu = 12.6968$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 13.8220$) vs *CR-DANTENet* ($\mu = 13.9714$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENet}$	0.413	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 13.8220$) vs *CR-DANTENetBw* ($\mu = 12.8982$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENetBw}$	0.001	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 12.5171$) vs ϵ *DANTENet* ($\mu = 12.3296$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 12.5171$) vs ϵ *DANTENetBw* ($\mu = 11.8015$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENetBw}$	0.502	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 12.5171$) vs *CR-DANTENet* ($\mu = 13.9714$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 12.5171$) vs *CR-DANTENetBw* ($\mu = 12.8982$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENetBw}$	0.212	$\gamma < p_{value}$ Insignificante
ϵ *DANTENet* ($\mu = 12.3296$) vs ϵ *DANTENetBw* ($\mu = 11.8015$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
ϵ *DANTENet* ($\mu = 12.3296$) vs *CR-DANTENet* ($\mu = 13.9714$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENet}$	0.279	$\gamma < p_{value}$ Insignificante
ϵ *DANTENet* ($\mu = 12.3296$) vs *CR-DANTENetBw* ($\mu = 12.8982$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.001	$\gamma > p_{value}$ Significante
ϵ *DANTENetBw* ($\mu = 11.8015$) vs *CR-DANTENet* ($\mu = 13.9714$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
ϵ *DANTENetBw* ($\mu = 11.8015$) vs *CR-DANTENetBw* ($\mu = 12.8982$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENetBw}$	0.494	$\gamma < p_{value}$ Insignificante
CR-DANTENet ($\mu = 13.9714$) vs *CR-DANTENetBw* ($\mu = 12.8982$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante

Tabela C.3: Resultados do teste estatístico t para as médias percentuais das perdas de pacotes do *TRÁFEGO I* da *REDE 1B*, realizados para os Algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw* com um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$. A probabilidade - p_{value} corresponde ao valor da estatística de teste que serve de comparar com o γ.

$\gamma = 0.05$			
Algoritmos	Testes de Hipóteses	p_{value}	Observações
AntNet ($\mu = 6.2019$) vs *Link-State* ($\mu = 8.0767$)	$H_0 : \mu_{AntNet} = \mu_{Link\text{-}State}$ $H_1 : \mu_{AntNet} \neq \mu_{Link\text{-}State}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 6.2019$) vs *AntNetBw* ($\mu = 6.0833$)	$H_0 : \mu_{AntNet} = \mu_{AntNetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{AntNetBw}$	0.685	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 6.2019$) vs ϵ-*DANTENet* ($\mu = 6.0601$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENet}$	0.563	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 6.2019$) vs ϵ-*DANTENetBw* ($\mu = 6.4472$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENetBw}$	0.428	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 6.2019$) vs *CR-DANTENet* ($\mu = 6.7551$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENet}$	0.124	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 6.2019$) vs *CR-DANTENetBw* ($\mu = 5.7994$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENetBw}$	0.096	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 6.0833$) vs ϵ-*DANTENet* ($\mu = 6.0601$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENet}$	0.925	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 6.0833$) vs ϵ-*DANTENetBw* ($\mu = 6.4472$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENetBw}$	0.242	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 6.0833$) vs *CR-DANTENet* ($\mu = 6.7551$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENet}$	0.063	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 6.0833$) vs *CR-DANTENetBw* ($\mu = 5.7994$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENetBw}$	0.239	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENet* ($\mu = 6.0601$) vs ϵ-*DANTENetBw* ($\mu = 6.4472$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{\epsilon DANTENetBw}$	0.147	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENet* ($\mu = 6.0601$) vs *CR-DANTENet* ($\mu = 6.7551$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENet}$	0.032	$\gamma > p_{value}$ Significante
ϵ-*DANTENet* ($\mu = 6.0601$) vs *CR-DANTENetBw* ($\mu = 5.7994$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.198	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENetBw* ($\mu = 6.4472$) vs *CR-DANTENet* ($\mu = 6.7551$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENet}$	0.404	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENetBw* ($\mu = 6.4472$) vs *CR-DANTENetBw* ($\mu = 5.7994$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENetBw}$	0.015	$\gamma > p_{value}$ Significante
CR-DANTENet ($\mu = 6.7551$) vs *CR-DANTENetBw* ($\mu = 5.7994$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.003	$\gamma > p_{value}$ Significante

Tabela C.4: Resultados do teste estatístico t para as médias percentuais das perdas de pacotes do *TRÁFEGO II* da *REDE 1B*, realizados para os Algoritmos *Link-State*, *AntNet*, *ϵ-DANTENet*, *CR-DANTENet*, *AntNetBw*, *ϵ-DANTENetBw*, *CR-DANTENetBw* com um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$. A probabilidade - p_{value} corresponde ao valor da estatística de teste que serve de comparar com o γ.

$\gamma = 0.05$			
Algoritmos	Testes de Hipóteses	p_{value}	Observações
AntNet ($\mu = 12.0756$) vs *Link-State* ($\mu = 14.8528$)	$H_0 : \mu_{AntNet} = \mu_{Link\text{-}State}$ $H_1 : \mu_{AntNet} \neq \mu_{Link\text{-}State}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 12.0756$) vs *AntNetBw* ($\mu = 11.8028$)	$H_0 : \mu_{AntNet} = \mu_{AntNetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{AntNetBw}$	0.063	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 12.0756$) vs *ϵ-DANTENet* ($\mu = 12.3296$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENet}$	0.118	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 12.0756$) vs *ϵ-DANTENetBw* ($\mu = 11.8015$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENetBw}$	0.149	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 12.0756$) vs *CR-DANTENet* ($\mu = 12.4617$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENet}$	0.108	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 12.0756$) vs *CR-DANTENetBw* ($\mu = 11.9468$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENetBw}$	0.288	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 11.8028$) vs *ϵ DANTENet* ($\mu = 12.3296$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENet}$	0.002	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 11.8028$) vs *ϵ DANTENetBw* ($\mu = 11.8015$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENetBw}$	0.995	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 11.8028$) vs *CR-DANTENet* ($\mu = 12.4617$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENet}$	0.008	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 11.8028$) vs *CR-DANTENetBw* ($\mu = 11.9468$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENetBw}$	0.259	$\gamma < p_{value}$ Insignificante
ϵ DANTENet ($\mu = 12.3296$) vs *ϵ DANTENetBw* ($\mu = 11.8015$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{\epsilon DANTENetBw}$	0.012	$\gamma > p_{value}$ Significante
ϵ DANTENet ($\mu = 12.3296$) vs *CR-DANTENet* ($\mu = 12.4617$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENet}$	0.595	$\gamma < p_{value}$ Insignificante
ϵ DANTENet ($\mu = 12.3296$) vs *CR-DANTENetBw* ($\mu = 11.9468$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.011	$\gamma > p_{value}$ Significante
ϵ DANTENetBw ($\mu = 11.8015$) vs *CR-DANTENet* ($\mu = 12.4617$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENet}$	0.016	$\gamma > p_{value}$ Significante
ϵ DANTENetBw ($\mu = 11.8015$) vs *CR-DANTENetBw* ($\mu = 11.9468$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENetBw}$	0.406	$\gamma < p_{value}$ Insignificante
CR-DANTENet ($\mu = 12.4617$) vs *CR-DANTENetBw* ($\mu = 11.9468$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.026	$\gamma > p_{value}$ Significante

Tabela C.5: Resultados do teste estatístico t para as médias percentuais das perdas de pacotes do *TRÁFEGO I* da *REDE 2A*, realizados para os Algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw* com um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$. A probabilidade - p_{value} corresponde ao valor da estatística de teste que serve de comparar com o γ.

$\gamma = 0.05$			
Algoritmos	Testes de Hipóteses	p_{value}	Observações
AntNet ($\mu = 10.3835$) vs *Link-State* ($\mu = 14.4241$)	$H_0 : \mu_{AntNet} = \mu_{Link\text{-}State}$ $H_1 : \mu_{AntNet} \neq \mu_{Link\text{-}State}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 10.3835$) vs *AntNetBw* ($\mu = 8.4067$)	$H_0 : \mu_{AntNet} = \mu_{AntNetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{AntNetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 10.3835$) vs ϵ-*DANTENet* ($\mu = 10.5205$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENet}$	0.386	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 10.3835$) vs ϵ-*DANTENetBw* ($\mu = 8.8614$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 10.3835$) vs *CR-DANTENet* ($\mu = 10.6400$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENet}$	0.124	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 10.3835$) vs *CR-DANTENetBw* ($\mu = 9.0669$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 8.4067$) vs ϵ-*DANTENet* ($\mu = 10.5205$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 8.4067$) vs ϵ-*DANTENetBw* ($\mu = 8.8614$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENetBw}$	0.311	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 8.4067$) vs *CR-DANTENet* ($\mu = 10.6400$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 8.4067$) vs *CR-DANTENetBw* ($\mu = 9.0669$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENetBw}$	0.172	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENet* ($\mu = 10.5205$) vs ϵ-*DANTENetBw* ($\mu = 8.8614$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENet* ($\mu = 10.5205$) vs *CR-DANTENet* ($\mu = 10.6400$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENet}$	0.497	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENet* ($\mu = 10.5205$) vs *CR-DANTENetBw* ($\mu = 9.0669$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENetBw* ($\mu = 8.8614$) vs *CR-DANTENet* ($\mu = 10.6400$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENetBw* ($\mu = 8.8614$) vs *CR-DANTENetBw* ($\mu = 9.0669$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENetBw}$	0.603	$\gamma < p_{value}$ Insignificante
CR-DANTENet ($\mu = 10.6400$) vs *CR-DANTENetBw* ($\mu = 9.0669$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante

Tabela C.6: Resultados do teste estatístico t para as médias percentuais das perdas de pacotes do *TRÁFEGO II* da *REDE 2A*, realizados para os Algoritmos *Link-State*, *AntNet*, *ϵ-DANTENet*, *CR-DANTENet*, *AntNetBw*, *ϵ-DANTENetBw*, *CR-DANTENetBw* com um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$. A probabilidade - p_{value} corresponde ao valor da estatística de teste que serve de comparar com o γ.

$\gamma = 0.05$			
Algoritmos	Testes de Hipóteses	p_{value}	Observações
AntNet ($\mu = 17.8664$) vs *Link-State* ($\mu = 20.2434$)	$H_0 : \mu_{AntNet} = \mu_{Link\text{-}State}$ $H_1 : \mu_{AntNet} \neq \mu_{Link\text{-}State}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 17.8664$) vs *AntNetBw* ($\mu = 16.4475$)	$H_0 : \mu_{AntNet} = \mu_{AntNetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{AntNetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 17.8664$) vs *ϵ-DANTENet* ($\mu = 17.8655$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENet}$	0.997	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 17.8664$) vs *ϵ-DANTENetBw* ($\mu = 17.1758$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENetBw}$	0.023	$\gamma > p_{value}$ Significante
AntNet ($\mu = 17.8664$) vs *CR-DANTENet* ($\mu = 18.1668$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENet}$	0.317	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 17.8664$) vs *CR-DANTENetBw* ($\mu = 17.1788$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENetBw}$	0.029	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 16.4475$) vs *ϵ-DANTENet* ($\mu = 17.8655$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 16.4475$) vs *ϵ-DANTENetBw* ($\mu = 17.1758$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENetBw}$	0.030	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 16.4475$) vs *CR-DANTENet* ($\mu = 18.1668$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 16.4475$) vs *CR-DANTENetBw* ($\mu = 17.1788$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENetBw}$	0.034	$\gamma > p_{value}$ Significante
ϵ-DANTENet ($\mu = 17.8655$) vs *ϵ-DANTENetBw* ($\mu = 17.1758$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{\epsilon DANTENetBw}$	0.013	$\gamma > p_{value}$ Significante
ϵ-DANTENet ($\mu = 17.8655$) vs *CR-DANTENet* ($\mu = 18.1668$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENet}$	0.270	$\gamma < p_{value}$ Insignificante
ϵ-DANTENet ($\mu = 17.8655$) vs *CR-DANTENetBw* ($\mu = 17.1788$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.018	$\gamma > p_{value}$ Significante
ϵ-DANTENetBw ($\mu = 17.1758$) vs *CR-DANTENet* ($\mu = 18.1668$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENet}$	0.004	$\gamma > p_{value}$ Significante
ϵ-DANTENetBw ($\mu = 17.1758$) vs *CR-DANTENetBw* ($\mu = 17.1788$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENetBw}$	0.993	$\gamma < p_{value}$ Insignificante
CR-DANTENet ($\mu = 18.1668$) vs *CR-DANTENetBw* ($\mu = 17.1788$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.006	$\gamma > p_{value}$ Significante

Tabela C.7: Resultados do teste estatístico t para as médias percentuais das perdas de pacotes do *TRÁFEGO I* da *REDE 2B*, realizados para os Algoritmos *Link-State*, *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw* com um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma$. A probabilidade - p_{value} corresponde ao valor da estatística de teste que serve de comparar com o γ.

$\gamma = 0.05$			
Algoritmos	Testes de Hipóteses	p_{value}	Observações
AntNet ($\mu = 5.6267$) vs *Link-State* ($\mu = 10.1156$)	$H_0 : \mu_{AntNet} = \mu_{Link\text{-}State}$ $H_1 : \mu_{AntNet} \neq \mu_{Link\text{-}State}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 5.6267$) vs *AntNetBw* ($\mu = 4.2468$)	$H_0 : \mu_{AntNet} = \mu_{AntNetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{AntNetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 5.6267$) vs ϵ-*DANTENet* ($\mu = 5.8543$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENet}$	0.131	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 5.6267$) vs ϵ-*DANTENetBw* ($\mu = 4.7708$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENetBw}$	0.001	$\gamma > p_{value}$ Significante
AntNet ($\mu = 5.6267$) vs *CR-DANTENet* ($\mu = 5.6701$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENet}$	0.771	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 5.6267$) vs *CR-DANTENetBw* ($\mu = 5.0627$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENetBw}$	0.007	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 4.2468$) vs ϵ-*DANTENet* ($\mu = 5.8543$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 4.2468$) vs ϵ-*DANTENetBw* ($\mu = 4.7708$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENetBw}$	0.062	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 4.2468$) vs *CR-DANTENet* ($\mu = 5.6701$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 4.2468$) vs *CR-DANTENetBw* ($\mu = 5.0627$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENetBw}$	0.002	$\gamma < p_{value}$ Significante
ϵ-*DANTENet* ($\mu = 5.8543$) vs ϵ-*DANTENetBw* ($\mu = 4.7708$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENet* ($\mu = 5.8543$) vs *CR-DANTENet* ($\mu = 5.6701$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENet}$	0.226	$\gamma < p_{value}$ Insignificante
ϵ-*DANTENet* ($\mu = 5.8543$) vs *CR-DANTENetBw* ($\mu = 5.0627$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENetBw* ($\mu = 4.7708$) vs *CR-DANTENet* ($\mu = 5.6701$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
ϵ-*DANTENetBw* ($\mu = 4.7708$) vs *CR-DANTENetBw* ($\mu = 5.0627$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENetBw}$	0.280	$\gamma < p_{value}$ Insignificante
CR-DANTENet ($\mu = 5.6701$) vs *CR-DANTENetBw* ($\mu = 5.0627$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.004	$\gamma > p_{value}$ Significante

Tabela C.8: Resultados do teste estatístico t para as médias percentuais das perdas de pacotes do *TRÁFEGO II* da *REDE 2B*, realizados para os Algoritmos *Link-State*, *AntNet*, *ϵ-DANTENet*, *CR-DANTENet*, *AntNetBw*, *ϵ-DANTENetBw*, *CR-DANTENetBw* com um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$. A probabilidade - p_{value} corresponde ao valor da estatística de teste que serve de comparar com o γ.

$\gamma = 0.05$			
Algoritmos	Testes de Hipóteses	p_{value}	Observações
AntNet ($\mu = 10.5494$) vs *Link-State* ($\mu = 14.53584$)	$H_0 : \mu_{AntNet} = \mu_{Link\text{-}State}$ $H_1 : \mu_{AntNet} \neq \mu_{Link\text{-}State}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 10.5494$) vs *AntNetBw* ($\mu = 9.5550$)	$H_0 : \mu_{AntNet} = \mu_{AntNetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{AntNetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 10.5494$) vs *ϵ-DANTENet* ($\mu = 10.4916$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENet}$	0.602	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 10.5494$) vs *ϵ-DANTENetBw* ($\mu = 9.7031$)	$H_0 : \mu_{AntNet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNet ($\mu = 10.5494$) vs *CR-DANTENet* ($\mu = 10.4842$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENet}$	0.595	$\gamma < p_{value}$ Insignificante
AntNet ($\mu = 10.5494$) vs *CR-DANTENetBw* ($\mu = 9.7501$)	$H_0 : \mu_{AntNet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNet} \neq \mu_{CRDANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 9.5550$) vs *ϵ DANTENet* ($\mu = 12.3296$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 9.5550$) vs *ϵ DANTENetBw* ($\mu = 11.8015$)	$H_0 : \mu_{AntNetBw} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{\epsilon DANTENetBw}$	0.496	$\gamma < p_{value}$ Insignificante
AntNetBw ($\mu = 9.5550$) vs *CR-DANTENet* ($\mu = 10.4842$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
AntNetBw ($\mu = 9.5550$) vs *CR-DANTENetBw* ($\mu = 9.7501$)	$H_0 : \mu_{AntNetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{AntNetBw} \neq \mu_{CRDANTENetBw}$	0.428	$\gamma < p_{value}$ Insignificante
ϵ DANTENet ($\mu = 12.3296$) vs *ϵ DANTENetBw* ($\mu = 11.8015$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{\epsilon DANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{\epsilon DANTENetBw}$	0.000	$\gamma > p_{value}$ Significante
ϵ DANTENet ($\mu = 12.3296$) vs *CR-DANTENet* ($\mu = 10.4842$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENet}$	0.950	$\gamma < p_{value}$ Insignificante
ϵ DANTENet ($\mu = 12.3296$) vs *CR-DANTENetBw* ($\mu = 9.7501$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.001	$\gamma > p_{value}$ Significante
ϵ DANTENetBw ($\mu = 11.8015$) vs *CR-DANTENet* ($\mu = 10.4842$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENet}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENet}$	0.000	$\gamma > p_{value}$ Significante
ϵ DANTENetBw ($\mu = 11.8015$) vs *CR-DANTENetBw* ($\mu = 9.7501$)	$H_0 : \mu_{\epsilon DANTENetBw} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENetBw} \neq \mu_{CRDANTENetBw}$	0.844	$\gamma < p_{value}$ Insignificante
CR-DANTENet ($\mu = 10.4842$) vs *CR-DANTENetBw* ($\mu = 9.7501$)	$H_0 : \mu_{\epsilon DANTENet} = \mu_{CRDANTENetBw}$ $H_1 : \mu_{\epsilon DANTENet} \neq \mu_{CRDANTENetBw}$	0.001	$\gamma > p_{value}$ Significante

APÊNDICE

D

Teste estatístico de *Shapiro-Wilk* para verificar o comportamento da média de pacotes perdidos nas redes em relação a distribuição normal

Tabela D.1: Teste estatístico de *Shapiro-Wilk* para verificar o comportamento da média de pacotes perdidos nas redes em relação a distribuição normal, originado pelos algoritmos *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw*, na *REDE 1A* com *TRÁFEGOS I e II*, para um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$ e a probabilidade p_{value} de obter um valor da estatística de teste que serve de comparar com o γ.

	TRÁFEGO I Shapiro-Wilk			TRÁFEGO II Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AntNet	,924	25	,062	,981	25	,907
AntNetBw	,896	25	,015	,920	25	,051
DanteNet	,969	25	,622	,973	25	,723
DanteNetBw	,851	25	,002	,833	25	,001
DanteNetCR	,978	25	,833	,900	25	,018
DanteNetCRBw	,935	25	,111	,882	25	,007

Tabela D.2: Teste estatístico de *Shapiro-Wilk* para verificar o comportamento da média de pacotes perdidos nas redes em relação a distribuição normal, originado pelos algoritmos *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw*, na *REDE 1B* com *TRÁFEGOS I e II*, para um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$ e a probabilidade p_{value} de obter um valor da estatística de teste que serve de comparar com o γ.

	TRÁFEGO I Shapiro-Wilk			TRÁFEGO II Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AntNet	,669	25	,000	,962	25	,450
AntNetBw	,950	25	,250	,874	25	,005
DanteNet	,958	25	,369	,938	25	,136
DanteNetBw	,967	25	,562	,873	25	,005
DanteNetCR	,749	25	,000	,814	25	,000
DanteNetCRBw	,959	25	,396	,964	25	,509

Tabela D.3: Teste estatístico de *Shapiro-Wilk* para verificar o comportamento da média de pacotes perdidos nas redes em relação a distribuição normal, originado pelos algoritmos *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw*, na *REDE 2A* com *TRÁFEGOS I e II*, para um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$ e a probabilidade p_{value} de obter um valor da estatística de teste que serve de comparar com o γ.

	TRÁFEGO I Shapiro-Wilk			TRÁFEGO II Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AntNet	,931	25	,092	,931	25	,092
AntNetBw	,950	25	,252	,950	25	,252
DanteNet	,938	25	,130	,938	25	,130
DanteNetBw	,946	25	,207	,946	25	,207
DanteNetCR	,941	25	,153	,941	25	,153
DanteNetCRBw	,958	25	,384	,958	25	,384

Tabela D.4: Teste estatístico de *Shapiro-Wilk* para verificar o comportamento da média de pacotes perdidos nas redes em relação a distribuição normal, originado pelos algoritmos *AntNet*, ϵ-*DANTENet*, *CR-DANTENet*, *AntNetBw*, ϵ-*DANTENetBw*, *CR-DANTENetBw*, na *REDE 2B* com *TRÁFEGOS I e II*, para um nível de significância de $\gamma = 5\%$, correspondente ao grau de confiança de 95% $(1 - \gamma)$ e a probabilidade p_{value} de obter um valor da estatística de teste que serve de comparar com o γ.

	TRÁFEGO I Shapiro-Wilk			TRÁFEGO II Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AntNet	,947	25	,209	,975	25	,768
AntNetBw	,957	25	,365	,961	25	,437
DanteNet	,957	25	,351	,904	25	,022
DanteNetBw	,957	25	,349	,980	25	,880
DanteNetCR	,824	25	,001	,975	25	,776
DanteNetCRBw	,965	25	,523	,954	25	,309