Universidade Federal do Ceará

Mestrado em Ciência da Computação

Utilizando Redes Neurais Artificiais para Predição de Falhas em *Links* de Redes Ópticas

Carlos Hairon Ribeiro Gonçalves

Fortaleza – Ceará

2003

Mestrado em Ciência da Computação

Universidade Federal do Ceará

Utilizando Redes Neurais Artificiais para Predição de Falhas em *Links* de Redes Ópticas

Carlos Hairon Ribeiro Gonçalves[1]

Setembro de 2003

Banca Examinadora :

- ✓ Prof. Dr. Antônio Mauro Barbosa de Oliveira
  Centro Federal de Educação Tecnológica do Ceará – CEFET-CE (Orientador)
- ✓ Profa. Dra. Rossana Maria de Castro Andrade (Co-orientadora)
  Universidade Federal do Ceará – UFC
- ✓ Prof. Dr. Luiz Fernando Gomes Soares
  Pontifícia Universidade Católica do Rio de Janeiro – PUC-RJ
- ✓ Prof. Dr. Guilherme de Alencar Barreto
  Universidade Federal do Ceará – UFC

[1] Financiado pela CAPES

Dedicatória

Dedico esta dissertação a minha mãe, Edna Ribeiro Gonçalves, eterna inspiração para a vida e a meu pai, João Francisco Ribeiro, eterna inspiração para o trabalho.

Agradecimentos

Gostaria de agradecer:

- ✓ a Deus, por guiar meus passos durante toda a minha vida;
- ✓ a professora Rossana Andrade, pela dedicação para comigo;
- ✓ ao professor Mauro Oliveira, pela ajuda, motivação e otimismo passado;
- ✓ a Marcus Antonio e Miguel Franklin, por terem contribuído com sua experiência e conhecimento na troca de idéias;
- ✓ a todos os colegas, principalmente, a Edson, Fernandes, Janine e Welsinner e Lídia pelo companheirismo e compartilhamento dos bons momentos;
- ✓ a todos os integrantes do LAR, principalmente, a George, Henrique e Tuan pela colaboração neste trabalho;
- ✓ a CAPES e ao departamento de computação da UFC, pela qualidade dos serviços prestados com destaque especial para Manuel Orley e Débora;
- ✓ em especial a Cícera Raquel, minha noiva, pelo apoio incondicional e por todas as flores que ela tem cultivado no meu jardim.

Resumo

Um dos problemas remanescentes no protocolo GMPLS (*Generalized Multiprotocol Label Switching*) é a ausência de mecanismos de re-roteamento rápido destinados ao tratamento de falhas de *links* ópticos. Redes Neurais Artificiais podem predizer situações problemáticas em tais enlaces e, deste modo, a forma de restauração de falhas 1:n (1 *backup* compartilhado entre n *links*) pode ser aproximada da forma de proteção de falhas 1+1 (1 *backup* tipo espelho para cada *link*). Tal aproximação proporciona uma redução de custos para sistemas ópticos que necessitam de alta confiabilidade. Este trabalho apresenta o ambiente de desenvolvimento de Agentes Inteligentes baseados em Redes Neurais Artificiais aplicados à gerência pró-ativa de redes IP, MPLS e GMPLS, denominado RENATA 2. Este ambiente é composto pelas ferramentas GDP e MSDP, desenvolvidas nesta dissertação, sendo integradas aos simuladores *ns* e JNNS já existentes. Uma das aplicações dos Agentes RENATA 2 é atuar na resolução do problema supracitado.

Abstract

A recurring problem in GMPLS (Generalized Multiprotocol Label Switching) is the absence of fast re-routing mechanisms designated to failure treatment of optical links. Artificial Neural Networks can predict problematic situations in such links and, thus, the type of restoring 1:n failures (1 shared backup between n links) can be approximated by the type of failure protection 1+1 (1 backup as a mirror to each link). Such approach provides a decrease in cost for optical systems that need high reliability. This work presents a development environment of Intelligent Agents based on Artificial Neural Networks applied to pro-active management of IP, MPLS, and GMPLS computer networks, which is called RENATA 2. The GDP and MSDP tools, developed in this research, are included in this environment and they are integrated to the existing *ns* and JNNS simulators. An application of RENATA 2 agents is provide the solution of the problem mentioned before.

Índice

Resumo v
Abstract vi
Lista de Figuras xi
Lista de Tabelas xiii
Glossário de Acrônimos xiv
Capítulo 1 Introdução 1
1.1 Internet 2 e as Tecnologias IP e ATM 2
1.2 Internet 3 e Sistemas Inteligentes 4
1.3 Motivação e Objetivos 5
1.4 Conclusões e Estrutura da Dissertação 7
Capítulo 2 Tecnologias de Gerenciamento de Redes e o Ambiente RENATA 9
2.1 Modelos Clássicos de Gerenciamento 10
2.2 Paradigmas Emergentes de Gerenciamento de Redes 14
2.2.1 WBEM - Web-Based Enterprise Management Architecture 14
2.2.2 PBNM - Policy-Based Network Management 16
2.2.2.1 Arquitetura PBNM 17
2.3 O Ambiente RENATA 18
2.3.1 Agentes Inteligentes 19
2.3.2 Arquitetura Funcional do RENATA 23
2.3.2.1 Módulo de Treinamento – Simulador ATM 24
2.3.2.2 Módulo de Treinamento – MSPD 24
2.3.2.3 Módulo de Treinamento – Simulador de Redes Neurais 25
2.3.2.4 Módulo Neural 25
2.3.2.5 Módulo de Gerência 26
2.3.3 Cálculo da Capacidade Requerida de Comutadores ATM 27
2.3.4 Controle de Admissão de Conexões de Redes ATM 28
2.4 Conclusões 29

Capítulo 3 Sistemas de Fibras Ópticas 31
3.1 Lasers Transmissores 32
3.1.1 Fatores que Influenciam no Sinal Emitido por um Laser Semicondutor 32
3.2 Fibras Ópticas 35
3.2.1 Atenuação Causada por Fibras Ópticas 36
3.3 Receptores Ópticos 37
3.4 Conclusões 39
Capítulo 4 Re-roteamento em Redes IP+GMPLS sobre DWDM 40
4.1 Re-roteamento em Redes MPLS/GMPLS 41
4.1.1 Comutação de Pacotes em Redes Ópticas 42
4.1.2 Seleção de Canal 44
4.1.3 Estabelecimento de Conexões Bidirecionais x Conexões Unidirecionais 44
4.1.4 Categorias de Restauração em Redes Ópticas 45
4.2 Redes Neurais para Predição de Falhas em *Links* de Redes Ópticas 47
4.2.1 Equipamentos de Transmissão Ópticos 48
4.2.2 Equipamentos de Transmissão WDM 49
4.2.3 Análise da Qualidade do Sinal a Partir da Camada IP 50
4.2.4 Predição de Séries Temporais 52
4.2.5 Técnicas de Modelagem Global de Séries Temporais 53
4.3 Trabalhos Correlatos 56
4.4 Conclusões 60
Capítulo 5 RENATA 2 – um Ambiente para Gerência Inteligente 61
5.1 Arquitetura do RENATA 2 62
5.1.1 *ns - Network Simulator* 63
5.1.1.1 *Nam - Network Animator* 64
5.1.1.2 MNS - MPLS *Network Simulator* 65
5.1.2 GDP - Gerador de Perturbações 66
5.1.2.1 Transmissor Óptico 67
5.1.2.2 Fibra Óptica 68
5.1.2.3 Receptor Óptico 69
5.1.2.4 Tipos de Simulação 69

5.1.2.5 Saída da Simulação....71
5.1.3 MSPD – Módulo de Seleção e Preparação de Dados....71
5.1.4 JNNS – *Java Neural Network Simulator* 73
5.2 Conclusões 73
Capítulo 6 Utilizando o RENATA 2 para Predição de Falhas de *Links* Ópticos....75
6.1 O Sistema Simulado 76
6.1.1 A Camada Física 76
6.1.2 As Camadas de Enlace e de Rede 77
6.2 A Máquina Inteligente dos Agentes 80
6.3 Resultados 81
6.3.1 Rede Neural Tipo 1 86
6.3.2 Rede Neural Tipo 2 89
6.3.3 Rede Neural Tipo 3 91
6.4 Conclusões 93
Capítulo 7 Cenário de Implementação 95
7.1 Implementação do GDP 95
7.2 Simulações do GDP - Perturbações....100
7.2.1 Processo de Montagem de Simulações do GDP – Matriz de Simulações..103
7.3 Simulações do *ns* 103
7.3.1 Processo de montagem de simulações do *ns* - Simulações Finais....104
7.4 Montagem dos Padrões de Treinamento....105
7.5 Cálculo do MSE nos Pontos de Pico Suaves 107
7.6 Conclusões 110
Capítulo 8 Conclusões e Trabalhos Futuros 111
Referências Bibliográficas 116
Bibliografia....123
Anexo A Redes Neurais Artificiais....125
A.1 Capacidades e Propriedades 126
A.2 Neurônios - Unidades de Processamento....129
A.3 Tipos de Função de Ativação 130
A.4 Arquitetura de Redes Neurais....132

A.4.1 Redes *Feedforward* de Uma Camada 132
A.4.2 Redes *Feedforward* Multicamada 133
A.4.3 Redes Recorrentes 134
A.5 Processo de Aprendizagem 135
A.6 Fases de um Projeto de uma Rede Neural 137
A.6.1 Definição do Problema 138
A.6.2 Seleção e Representação de Dados 139
A.6.3 Seleção do Modelo da Rede Neural 139
A.6.4 Especificação da Arquitetura da Rede Neural 139
A.6.5 Configuração dos Parâmetros de Treinamento 140
A.6.6 Verificação do Aprendizado da Rede 140
A.6.7 Uso da Rede 140
Anexo B Base das Simulações 142

Lista de Figuras

Figura 1.1 – Internet 2 no Continente Americano. 3
Figura 1.2 – Evolução das redes Ópticas. 7
Figura 2.1 – Arquitetura WBEM. 15
Figura 2.2 – Arquitetura de Gerência Paseada em Política.....18
Figura 2.3 – Agente x Ambiente.....20
Figura 2.4 – Agente Inteligente x Ambiente. 22
Figura 2.5 – Arquitetura Funcional do RENATA. 23
Figura 2.6 – Configuração da gerência do dispositivo. 27
Figura 3.1 – Efeitos da variação de temperatura em um transmissor laser.....33
Figura 3.2 – Curvas L-I (Luz-Corrente) de um laser InGaAsP. 34
Figura 3.3 – Tipos de fibras ópticas com a representação física do sinal transmitido. 35
Figura 4.1 – Restauração de uma rede IP sobre MPLS.....42
Figura 4.2 – Uma rede óptica em múltiplas camadas. 43
Figura 4.3 – WDM x Comutação de Pacotes. 44
Figura 4.4 – Rede Neural 5–10–1 (parâmetros de entrada X1,..., X5; e valor de saída X6). 55
Figura 5.1 – Arquitetura RENATA 2.....62
Figura 5.2 – Interface do *Network Animator*.....64
Figura 5.3 – Link Óptico. 67
Figura 5.4 – Janela de entrada de dados do transmissor. 68
Figura 5.5 – Simulação Linear. 70
Figura 5.6 – Simulação Aleatória Linear. 70
Figura 5.7 – Simulação Aleatória. 71
Figura 5.8 – Saída típica do Gerador de Perturbações.....72
Figura 5.9 – Janela principal do Gerador de Perturbações. 72
Figura 6.1 – Topologia da rede simulada.....78
Figura 6.2 – Exemplo de pontos de picos. 84

Figura 6.3 – Exemplo de acerto na predição de falhas em pontos de picos.....85
Figura 6.4 – Exemplo de erro na predição de falhas em pontos de pico. 86
Figura 6.5 – Gráfico MSE de treinamento da rede neural tipo 1. 88
Figura 6.6 – Gráfico MSE de treinamento da rede neural tipo 2. 90
Figura 6.7 –Saídas dos arquivos Testa.res x Testa.pat sob a rede neural tipo 2. 91
Figura 6.8 – Gráfico MSE de treinamento da rede neural tipo 3. 92
Figura 7.1 – Diagrama de Classes do GDP.de Classes do GDP.....97
Figura 7.2 – Diagrama de Seqüência do GDP. 98
Figura 7.3 – Exemplo de andamento de uma simulação do GDP. 101
Figura 7.4 – Diagrama de seqüência mais genérico do ambiente RENATA 2. 109

Figura A.1 – Modelo básico para um neurônio. 129
Figura A.2 – (a) Função de limiar. (b) Função linear por partes. (c) Função sigmóide. 132
Figura A.3 – Rede neural *feedforward* unicamada. 133
Figura A.4 – Rede neural 5-3-2, *feedforward m*ulticamada totalmente ligada. 134
Figura A.6 – Exemplo de rede neural recorrente.....135
Figura A.7 – Fases do desenvolvimento de uma rede neural. 138

Lista de Tabelas

Tabela 2.1 - Passos para o desenvolvimento de um Agente RENATA. 27
Tabela 4.1 - Métodos para prover restauração em redes ópticas.....46
Tabela 6.1 - Especificações do sistema Mux/Demux DWDM Altamar. 77
Tabela 6.2 - Parâmetros de treinamento das redes MPL escolhidas.....82
Tabela 6.3 - Resultados da Rede Tipo 1. 88
Tabela 6.4 - Resultados da Rede Tipo 2. 90
Tabela 6.5 - Resultados da Rede Tipo 3. 92
Tabela 7.1 - Descrição das Classes do *GDP*. 99
Tabela 7.2 - Detalhamento das simulações de perturbações feitas no GDP. 102
Tabela 7.3 - Descrição das classes do MSPD.....107

Tabela A.1 - Algoritmos de aprendizagem e suas características. 137

Glossário de Acrônimos

ANFIS	*Adaptive Network Based Fuzzy Inference System*
ATM	*Asynchronous Transfer Mode*
BBNs	*Bayesian Belief Networks*
BER	*Bit Error Rate*
CBR	*Constant Bit Rate*
CIM	*Common Information Mode*
CMIP	*Common Management Information Protocol*
COPS	*Common Open Policy Service*
Diff-Sev	*Differentiated Services*
DWDM	*Dense Wavelength Division Multiplexing*
Edge LSRs	*Edge Label Switch Routers*
FEC	*Forwarding Equivalence Classes*
GMPLS	*Generalized Multiprotocol Label Switching*
HMMP	*Hypermedia Management Protocol*
HMMS	*Hypermedia Management Scheme*
IETF	*Internet Engineering Task Force*
ILMI	*Integrated Local Management Interface*
IP	*Internet Protocol*
IPv6	*Internet Protocol Version 6*
ISO	*International Standardization Organization*
LAN	*Local Area Network*
LDPA	*Light Directory Access Protocol*
LED	Diodo Emissor de Luz
LER	*Label Edge Routers*
LIB	*Label Information Base*

LMP	*Link Management Protocol*
LOL	*Loss of Light*
LSPs	*Label Switch Paths*
LSRs.	*Label Switch Routers*
MAN	*Metropolitan Area Network*
MC	*Management Console*
MIB	*Management Information Base*
MOF	*Managed Object Format*
MLP	*Multi-Layer Prceptron*
MPLS	*Multiprotocol Label Switching*
MSE	*Mean Square Error*
MSPD	*Módulo de Seleção e Preparação de Dados*
NAR	*Nonlinear Autoregressive*
NIST	*National Institute of Standards and Technology*
ns	*Network Simulator*
OSI	*Open Systems Interconnection*
OSRN	*Optical Signal-to-Noise Ratio*
OTS	*Optical Transport System*,
OXC	*Optical Cross-Connects*
PA	Progressão Aritmética
PBNM	*Policy-Based Network Management*
PDL	*Policy Definition Language*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Points*
PPB	Ponto Pico Brusco
PPS	Ponto Pico Suave
PR	*Policies Repository*
QoS	*Quality of Service*
RENATA	REdes Neurais Aplicadas ao Tráfego ATM
RFCs	*Request for Comments*
RNA	Rede Neural Artificial

RP	*Policies Repository*
SDH	*Synchronous Digital Hierarchy*
SNMP	*Simple Network Management Protocol*
SNNS	*Stuttgart Neural Network Simulator*
SNR	*Signal-to-Noise Ratio*
SONET	*Synchronous Optical NETwork*
TCP	*Transmission Control Protocol*
UML	*Unified Modeling Language*
VBR	*Variable Bit Rate*
VINT	*Virtual InterNetwork Testbed*
VLSI	*Very Large Scale Integration*
VPN	*Virtual Private Network*
WBEM	*Web-Based Enterprise Management Architecture*
WDM	*Wavelength Division Multiplexing*
XML	*Extended Markup Language*

Capítulo 1

Introdução

O compartilhamento de recursos em ambientes corporativos é um fator fundamental para a competitividade das empresas na economia atual. As redes de computadores têm como principal função permitir o compartilhamento destes recursos, sejam eles físicos ou lógicos, melhorando a interatividade entre os componentes de um grupo de trabalho. Se o tráfego destas redes era apenas caracterizado pelo transporte de dados, hoje é bastante heterogêneo. As aplicações atuais de voz sobre IP (*Internet Protocol*), videoconferência, telemetria, ensino a distância, entre outras, exigem serviços mais eficientes.

Por exemplo, durante a Copa do Mundo de Futebol de 2002 montou-se uma rede de computadores que suportava mais de 40.000 usuários. Dentre os requisitos necessários estavam confiabilidade, precisão, disponibilidade, grande largura de banda passante, segurança das informações, suporte a aplicações distribuídas e multimídia, principalmente voz sobre IP. Exemplos como este mostram que, cada vez mais, têm crescido as exigências, as diversidades e os avanços tanto na área de telecomunicações como na área de informática, implicando assim em sistemas de gerenciamento cada vez mais sofisticados, em que soluções inovadoras, robustas e confiáveis são vistas como características comuns.

Um estudo realizado pela Universidade de Austin, EUA, revela que uma falha em uma rede de computadores causa um prejuízo de 2% a 30% da receita anual de uma empresa do primeiro ao trigésimo dia de paralisação respectivamente [23]. Em

certas ocasiões é inadmissível uma falha na rede de computadores. Em um evento curto como a Copa do Mundo, por exemplo, uma falha da rede de computadores pode ocasionar a frustração de bilhões de espectadores e prejuízos exorbitantes. Em aplicações críticas, por exemplo, como o monitoramento de uma usina de energia nuclear, uma falha pode causar prejuízos incalculáveis se envolver vítimas humanas.

Assim, percebe-se que a necessidade de gerenciamento das redes de computadores é inerente ao crescimento e competitividade das instituições atuais. Em certas ocasiões, como as já citadas, o simples paradigma de Melhor Esforço (*Best Effort*) do *Internet Protocol* não é mais suficiente. O gerenciamento pode otimizar a infra-estrutura das redes de computadores tornando-as mais robustas. Neste cenário, surge a necessidade de parâmetros de QoS (*Quality of Service*) capazes de quantificar melhores valores de vazão, atraso, *jitter* e taxa de perdas limitadas. Desta forma, a Internet está migrando do melhor esforço para uma solução que forneça mecanismos de provisão de QoS.

1.1 Internet 2 e as Tecnologias IP e ATM

Nos anos 90, o desenvolvimento da Internet 2 tinha como objetivo o desenvolvimento de tecnologias e aplicações inter-redes para a comunidade acadêmica e de pesquisa, dentre as quais destacavam-se: telemedicina, bibliotecas digitais, laboratórios virtuais. Tais aplicações necessitam de características relacionadas com qualidade de serviço, tais como, garantia de entrega de pacotes e garantia de retardo, as quais não são implementadas na Internet padrão. O objetivo final da iniciativa não é somente o desenvolvimento de pesquisas exclusivamente voltadas para a área acadêmica, mas a transferência, ao setor comercial, das tecnologias desenvolvidas e testadas ao longo da execução dos projetos [43]. A Figura 1.1 mostra os países do continente americano que estão ligados atualmente na Internet 2. No Brasil, a versão 2 da Internet foi implantada pelas RMAVs (Redes Metropolitanas de Alta Velocidade) que são interligadas pela RNP 2 (Rede Nacional de Pesquisa).

Figura 1.1 – Internet 2 no Continente Americano[2].

Na década de 90, o IP se mostrava como uma solução barata (*Ethernet* + IP) e descomplicada para as redes de computadores, além de seu amplo uso em virtude da Internet. O ATM era uma solução mais robusta que provia eficiência a aplicações que necessitassem de garantias de qualidade de serviço. ATM mostrava-se, também, como uma tecnologia unificadora de LANs e WANs [24]. Para as RMAVs brasileiras, o ATM foi escolhido como tecnologia de rede metropolitana, pois se adequava as exigências da Internet 2 apesar de seu alto custo.

Por volta de 1995, o IETF (*Internet Engineering Task Force*) através das RFCs (*Request for Comments*) 1483, 1577 e 1755 propôs a criação de redes virtuais IP sobre uma rede ATM, denominada de IP sobre ATM. Tal iniciativa visava propiciar um serviço não orientado a conexão em redes ATM, através da sobreposição do protocolo IP às redes ATM (abordagem indireta), alcançando com isso, a interconexão de LANs e MANs.

---

[2] Figura disponível em [43].

Enquanto a popularidade do IP o transformou em padrão *de facto* para as redes de computadores, os recursos oferecidos pelo protocolo ATM como, baixa latência e a possibilidade de se estabelecer prioridade no envio de células, possibilitaram a implantação de serviços isócronos[3] com elevada eficiência. Uma consideração importante a respeito do ATM é seu suporte ao IP. Como já mencionado, o IP é um protocolo bastante difundido e permite a conectividade entre redes com diferentes protocolos na camada de enlace. Desta forma, o IP sobre ATM possibilita a comutação de pacotes em uma infra-estrutura ATM, além de proporcionar uma conectividade mais abrangente.

1.2 Internet 3 e Sistemas Inteligentes

A Internet 3 (I3) deverá tomar decisões acerca do próprio tráfego e oferecer garantias de *QoS*. Tal fato deverá prover um verdadeiro "porto seguro" para aplicações multimídia, como também, fornecerá protocolos mais estáveis e falhas serão, praticamente, imperceptíveis ao usuário. A concretização dessa proposta requer uma rede inteligente e ativa.

Um sistema inteligente deve ser capaz de adaptar a rede a situações novas ou inesperadas, bem como controlar ou atenuar falhas. Tais sistemas podem ser vistos com máquinas que não só aprendem, mas também fazem inferências sobre determinado assunto. Agentes Inteligentes têm se destacado nesta área. Um agente é uma entidade autônoma (geralmente um *software*) capaz de se comunicar com outros agentes e monitorar o ambiente ao seu redor.

Nesse contexto, destacam-se as redes ativas e os agentes móveis. As redes ativas agem no nível de roteamento e adicionam código aos nós da própria rede e nas estações de trabalho dos usuários para adaptar ambos ao tipo de informação recebida. Por exemplo, os cabeçalhos dos pacotes podem conter código que forneçam procedimentos a serem seguidos pelos roteadores da rede. Agentes móveis podem atuar em diferentes níveis e/ou pontos da rede, por exemplo, roteadores, estações de

[3] Serviços sensíveis a intervalos de tempos constantes, que necessitam de tráfego tipo CBR, por exemplo, certas transmissões de vídeo e áudio.

trabalho e servidores. Tais agentes coletam informações, repassando-as para o gerente da rede, ou se forem ativos ou autônomos, podem resolver problemas de gerenciamento diretamente.

Um problema a ser resolvido para implementação de uma rede inteligente é a escolha de uma arquitetura que integre entidades inteligentes. Uma proposta de uma arquitetura para este fim é o gerenciamento baseado em políticas PBNM (*Policy-Based Network Management*). O conceito de políticas consiste na idéia de que cada recurso ou processo da rede deve agir conforme uma regra preestabelecida. Um exemplo de política seria na ausência de congestionamento, disparar um mecanismo redundante para um *link* que esteja apresentando um número excessivo de pacotes perdidos.

1.3 Motivação e Objetivos

Em 1999, foi criado no LAR[4] em conjunto como o MCC[5], o RENATA (REdes Neurais Aplicadas ao Tráfego ATM), um ambiente destinado a gerência pró-ativa de redes ATM [16]. Para suportar as especificidades da gerência ATM, o RENATA permite o desenvolvimento de Agentes Inteligentes baseados em Redes Neurais Artificiais (RNAs). Dentre os possíveis problemas solucionados pelo RENATA estão: a estimativa da capacidade requerida em comutadores ATM [8] e a realização do CAC (Controle de Admissão de Conexões) VBR (*Variable Bit Rate*) em uma rede ATM [16]. Este fato ocasionou o surgimento de pesquisas relacionadas a RNAs na gerência de redes ATM pelo laboratório supracitado.

Quando o RENATA foi desenvolvido, pensava-se que a tecnologia ATM se tornaria em poucos anos um padrão para redes globais, metropolitanas e até mesmo para redes locais. Entretanto, o modelo de quatro camadas, em que o IP é a camada de suporte a aplicações e serviços, ATM se responsabiliza pela engenharia de tráfego,

---

[4] Laboratório Multiinstitucional de Redes de Computadores e Sistemas Distribuídos do CEFET-CE (Centro Federal de Educação Tecnológica do Ceará).

[5] Mestrado em Ciência da Computação da UFC (Universidade Federal do Ceará).

SONET[6]/SDH[7] suportam o transporte de dados e DWDM[8] atua como infra-estrutura física está em fase desuso devido aos seguintes fatores:

- ✓ custo elevado;
- ✓ complexidade do gerenciamento dos quatro planos;
- ✓ subutilização dos recursos providos pelo modelo de sobreposição do IP sobre ATM e
- ✓ o aparecimento de tecnologias mais eficientes.

O MPLS (*Multiprotocol Label Switching*) foi projetado para aproveitar a infra-estrutura já existentes de redes de computadores, sendo uma forma de possibilitar comutação rápida de rótulos em redes IP. Com o uso da forma generalizada do MPLS (GMPLS) é possível colocar o IP diretamente sobre redes ópticas para dar suporte a aplicações isócronas. Segundo [3], há uma forte tendência para o uso de IP+GMPLS sobre redes DWDM em substituição ao modelo de quatro camadas no qual a "pedra angular" era o ATM/SONET. A Figura 1.2 mostra a evolução das redes de computadores em direção ao modelo IP+GMPLS sobre uma infra-estrutura óptica. Atualmente, o ATM ainda resiste como uma solução viável para *backbones* de redes metropolitanas, mas com o surgimento do IPv6 (*Internet Protocol Version 6*) e das tecnologias MPLS (*Multiprotocol Label Switching*) e Diff-Sev (*Differentiated Services*) a tendência é, cada vez mais, o fortalecimento destas tecnologias, as quais prometem inserir formas de garantias de qualidade de serviço no protocolo IP.

Este trabalho apresenta um ambiente de gerenciamento pró-ativo de rede IP, MPLS, GMPLS de acordo com a arquitetura RENATA. A nova versão do RENATA se chama RENATA 2, sendo uma proposta de melhoria e simplificação da interface com o usuário, como também, de adequação à nova realidade da área de redes de computadores.

Um dos problemas atuais do MPLS, quando este é usado sobre redes ópticas (MPλS ou GMPLS), é permitir o re-roteamento rápido após uma falha ocorrida em um *link* de transmissão. Agentes inteligentes baseados em redes neurais podem prever

---

[6] SONET (*Synchronous Optical NETwork*).

[7] SDH (*Synchronous Digital Hierarchy).*

[8] DWDM (*Dense Wavelength Division Multiplexing*).

falhas em *links* IP+GMPLS sobre DWDM e desta forma garantir agilidade no re-roteamento destas redes. A Máquina Inteligente[9] de tais agentes é apresentada no Capítulo 7 (Cenário de Experimentação), sendo tais agentes uma proposta de solução para o problema de re-roteamento citado.

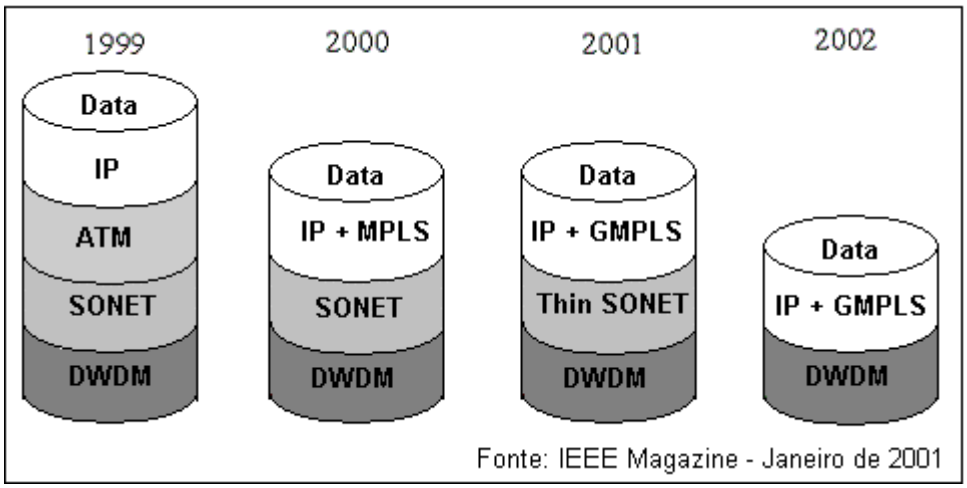

Figura 1.2 – Evolução das redes Ópticas[10].

1.4 Conclusões e Estrutura da Dissertação

O IP sobre ATM clássico é um modelo de sobreposição de protocolos que não aproveita por completo as potencialidades do ATM. Em oposição, o MPLS foi proposto como uma tecnologia unificadora e preparada para adicionar novas funcionalidades aos diversos protocolos. Por isso, o esse protocolo vem ganhando espaço no mercado de redes de computadores. O MPLS foi inicialmente pensado como uma extensão do protocolo IP [7] [11]. Ele agrega formas de garantias qualidade ao IP, como é o caso de reserva de banda, ou estabelecimento de uma hierarquia de prioridades na entrega de pacotes.

---

[9] Máquina Inteligente: estrutura computacional que proporciona autonomia ao agente para tomada de decisão.

[10] Figura traduzida e retirada da página 145 de [3].

O ambiente RENATA 2 tem, neste trabalho, a preocupação de fornecer mecanismos para a construção de agentes inteligentes aplicados ao gerenciamento destas novas tecnologias de redes, visto que a inteligência tende a estar presente nas futuras redes de computadores para prover, cada vez mais, um ambiente de comunicação confiável. Portanto, e**sta dissertação diz respeito ao processo de construção da máquina inteligente (redes neurais artificiais do tipo perceptron de múltiplas camadas) de softwares agentes que se aplicam ao gerenciamento de falhas em *links* ópticos, usado como cenário de experimentação redes IP+GMPLS sobre DWDM.**

O Capítulo 2 trata das tecnologias de gerenciamento de redes de computadores, com ênfase no ambiente RENATA, que utiliza mecanismos inteligentes para a tomada de decisão. No Capítulo 3, os sistemas de fibras ópticas são abordados, com enfoque nos seus componentes básicos. O Capítulo 4 explica o problema do re-roteamento rápido em redes GMPLS e também como redes neurais podem ajudar na solução desse problema. Já Capítulo 5, o RENATA 2 é apresentado. O cenário de experimentação com seus resultados é assunto do Capítulo 6. Por fim, o Capítulo 7 traz detalhes da implementação das ferramentas desenvolvidas para o RENATA 2 e o Capítulo 8, as conclusões e as propostas de trabalhos futuros. Após as referências bibliográficas e a bibliografia são expostos três anexos: o Anexo A é um tutorial sobre RNAs, o Anexo B comenta o *script* do *ns* responsável pela base das simulações desenvolvidas nesta dissertação e o Anexo C discorre sobre o MPLS.

Capítulo 2

Tecnologias de Gerenciamento de Redes e o Ambiente RENATA

O contínuo crescimento da economia mundial exige, cada vez mais, a busca pela excelência de qualidade de produtos e serviços. Assim, as redes de computadores devem apresentar novas soluções de aplicações que permitam interconectar empresas geograficamente distribuídas Nesse contexto, as aplicações, tais como, vídeo conferência, comunicação pessoal, telemedicina e educação a distância, estão em ascensão. Estas aplicações requerem novas soluções de gerenciamento para garantir o bom funcionamento das redes de computadores, o que é essencial para a manutenção da qualidade dos seus serviços prestados. Desta forma, o surgimento de novos usuários e aplicações faz as redes de computadores estarem em um constante processo de aumento de tamanho e complexidade.

De acordo com [28], é de responsabilidade do gerenciamento de redes a coordenação (controle de atividades e monitoração de uso) de recursos materiais (modems, roteadores, pontes, entre outros) e lógicos (protocolos), fisicamente distribuídos na rede, assegurando, na medida do possível, confiabilidade, tempos de resposta aceitáveis e segurança das informações. Muitas vezes, o desenvolvimento e implantação de aplicações de gerenciamento de redes são atividades de difícil resolução. Isto ocorre devido à natureza heterogênea, tanto das redes de computadores, como das soluções proprietárias de gerência fornecidas por diversos fabricantes.

Este capítulo apresenta uma descrição dos principais modelos de gerenciamento de redes de computadores. Na Seção 2.1 é feita uma explanação sobre os modelos clássicos de gerência de redes (Modelos OSI e Internet), enquanto na Seção 2.2 são apresentados dois paradigmas mais recentes de gerenciamento de redes: WBEM (*Web-Based Enterprise Management Architecture*) e PBNM (*Policy-Based Network Management*). A Seção 2.3 apresentada o ambiente RENATA, o qual se destina ao apoio da gerencia ATM. Pro fim, na Seção 2.4 as conclusões deste capítulo são introduzidas.

2.1 Modelos Clássicos de Gerenciamento

Para garantir a interoperabilidade dos equipamentos de uma rede, várias organizações como a ISO *(International Standardization Organization*) através de sua subdivisão OSI (*Open Systems Interconnection*) e a IETF (*Internet Engineering Task Force*) têm proposto soluções de padrões na área de gerência de redes. Na gerência OSI o protocolo adotado é o CMIP (*Common Management Information Protocol*) e o IETF contempla o modelo Internet que utiliza o protocolo SNMP (*Simple Network Management Protocol*). O SNMP se enquadra na camada de aplicação do TCP/IP. A popularidade das redes TCP/IP contribuiu para que o modelo Internet se tornasse o padrão de *facto* da gerência de redes de computadores. Desta forma, o SNMP é incluído comumente na maioria dos sistemas operacionais de redes e softwares de gerência. Assim, este protocolo (SNMP) atingiu grande número de objetos e agentes instalados nas redes de computadores atuais.

Independentemente do protocolo adotado, os seguintes termos chaves são usados em gerência de redes.

1. **Objeto Gerenciado -** Um objeto gerenciado é a representação de um recurso da rede, o qual se queira monitorar e/ou controlar. Tal recurso pode ser lógico ou físico, como por exemplo, uma entidade de camada, uma conexão ou um dispositivo de comunicação. Para se tornar visível a um sistema de gerenciamento convencional, um recurso deve ser tratado necessariamente como um objeto gerenciado. A definição de um objeto apresenta dois aspectos

principais: sua localização dentro do sistema e sua natureza. Desta forma, uma boa modelagem de um objeto deve levar em consideração as seguintes entidades:

- ✓ Atributos;
- ✓ Operações as quais os objetos podem ser submetidos;
- ✓ Notificações que um objeto pode emitir sobre eventos de gerência; e
- ✓ Relações possíveis entre os objetos gerenciados.

2. **MIB (*Management Information Base*) -** A MIB é um repositório de dados local que armazena informações sobre o conjunto de objetos gerenciados relativos a um sistema de gerência. A MIB procura abranger todas as informações necessárias para a gerência da rede, visando automatizar grande parte das tarefas de gerência.
3. **Paradigma Gerente x Agente -** Os elementos de uma rede de computadores são entidades naturalmente distribuídas pela área geográfica de abrangência da rede. Assim uma aplicação de gerência que se proponha a monitorar e/ou controlar tais elementos é inerentemente distribuída. Os processos utilizados por tais aplicações são:
   - ✓ Processo Gerente: é a parte da aplicação distribuída associada ao usuário. Sua principal responsabilidade é realizar operações de gerência sobre os objetos gerenciados, bem como receber notificações sobre os mesmos. Para isso se relaciona de maneira impositiva sobre os Processos Agentes.
   - ✓ Processo Agente: é a parte da aplicação distribuída que irá executar sobre os objetos gerenciados os comandos enviados pelo gerente. Assim este processo tem a obrigação de capturar e transmitir ao gerente o estado e/ou comportamento do objeto gerenciado.
4. **SMI (*Structure of Management Information*) –** A SMI funciona como uma interface bem definida entre o protocolo de gerência e a MIB. E pode ser definida como um conjunto de regras a serem obedecidas na identificação e na definição de objetos na MIB.

Além disso, os requisitos definidos pela ISO também devem ser satisfeitos pelas atividades de gerenciamento de sistemas, os quais são classificados em cinco áreas funcionais:

- ✓ **Gerenciamento de Falhas –** Os meios para detecção de falhas, bem como o isolamento e correção de operações anormais em um ambiente gerenciado, são de responsabilidade desta área do gerenciamento;
- ✓ **Gerenciamento de Configuração –** Meios para controlar e identificar objetos gerenciados, bem como coletar e prover dados que facilitem o fornecimento contínuo dos serviços do objeto em questão, realizando, caso necessário, a reconfiguração do mesmo;
- ✓ **Gerenciamento de Desempenho** – Avalia o comportamento dos objetos gerenciados e a eficiência de suas atividades;
- ✓ **Gerenciamento de Contabilização** – Inclui funções para informar aos usuários os custos ou recursos consumidos de uma entidade da rede;
- ✓ **Gerenciamento de Segurança** – Atualmente é uma das áreas de maior demanda por aplicações devido à fragilidade de segurança dos sistemas operacionais de rede de computadores em uso. Inclui funções para criar e controlar mecanismos de segurança, além de distribuir informações e registrar eventos relativos à segurança da rede.

O gerenciamento OSI utiliza o CMIP e o gerenciamento Internet utiliza o protocolo SNMP. O CMIP baseia-se em um modelo orientado à conexão enquanto o SNMP trabalha sem conexão. Um objeto gerenciado OSI é definido segundo o paradigma de orientação a objetos. Assim, é definido em termos de seus atributos, operações, notificações que pode emitir e relações com outros objetos. Os objetos SNMP não são considerados como tais, segundo a orientação a objetos. Eles são apenas variáveis simples com algumas características básicas, como seu tipo e seu modo de acesso. Da mesma forma como os agentes CMIP são mais sofisticados que os agentes SNMP, os primeiros são mais avançados que os segundos, embora mais complexos. A simplicidade do SNMP, protocolo da camada de aplicação TCP/IP,

associado à popularização da Internet, fez com que a maioria das soluções de gerência se baseasse no SNMP. Algumas limitações do SNMP são o serviço não confiável de mensagens, funcionalidade restrita, filosofia de aquisição de informações falhas, entre outras. Mesmo assim, este protocolo ganhou grande popularidade.

As diferenças entre os dois modelos propostos, OSI e Internet, podem ser resumidas da seguinte maneira [23]:

- ✓ **Filosofia de aquisição de informações** – O SNMP indaga periodicamente cada recurso sobre seu estado, com possibilidade do agente sinalizar ao gerente a necessidade de indagação através de um *trap*. O CMIP utiliza tanto a técnica de indagação periódica por parte do gerente, como o mecanismo de notificação direta por parte do agente, eliminado assim indagações *a posteriori.*
- ✓ **Funcionalidade –** O CMIP se mostra um protocolo mais funcional que o SNMP, uma vez que é baseado em classes, podendo criar e eliminar objetos dinamicamente.
- ✓ **Tamanho e Desempenho:** Uma implementação SNMP tende a ser mais rápida e menor que uma implementação CMIP, cujo uso requer mais recursos de memória e de processamento que a implementação do SNMP.
- ✓ **Protocolos de Transporte –** O SNMP requer apenas um simples datagrama como mecanismo de transporte de dados, já o CMIP exige um serviço confiável, tal qual o TCP.
- ✓ **Padrões de Teste –** Como o CMIP é um padrão internacional *de jure*, possui um conjunto de padrões de teste de conformidade e de interoperabilidade. Em contraste, o SNMP não é um padrão *de jure,* embora seja *de facto.* Assim os fabricantes não dispõem de padrões de conformidade deste protocolo e suas implementações são verificadas apenas através de testes de interoperabilidade.

2.2 Paradigmas Emergentes de Gerenciamento de Redes

O WBEM e PBNM são dois paradigmas recentes de gerenciamento de redes de computadores se comparados com os modelos clássicos. Ambos procuram ser suficientemente escaláveis para acompanhar, adequadamente, o gerenciamento de uma rede de computadores em diferentes estágios de complexidade.

Antes que estes dois modos de gerenciamento assumissem um papel de destaque, o ambiente RENATA (precursor do presente trabalho) já se preocupava em dar suporte às especificidades impostas pelo ATM. Para isso, utiliza Agentes Inteligentes Baseados em Redes Neurais Artificiais. A tarefa dos agentes é distribuir inteligência em diversos pontos da rede para que o gerente não seja sobrecarregado, além de permitir soluções para problemas não convencionais, tais como, o controle de admissão de conexões e o cálculo da capacidade requerida em comutadores ATM.

2.2.1 WBEM - Web-Based Enterprise Management Architecture

WBEM é uma tecnologia desenvolvida por um consórcio de empresas formadas pela Microsoft, Intel, IBM, BMC *Softwares*, Compaq e Cisco, e recebe suporte de mais outras 75 companhias [26]. O objetivo do WBEM é simplificar o gerenciamento de ambientes de tecnologia da informação distribuídos e complexos. Uma proposta semelhante à desse consórcio é a JMAPI (*Java Management Application Program Interface*) definida na década de 90 pelas empresas: JavaSoft, BMC, Cisco, Bay Networks, dentre outras. Atualmente, a ferramenta JMX (*Java Management Extensions*) substitui a JMAPI [29].

O modelo proposto pelo WBEM é similar ao gerenciamento OSI. Ambos usam o paradigma gerente/agente. Os agentes em conjunto com o protocolo de acesso mantêm a MIB e um serviço de gerenciamento. Esses dois modelos de gerenciamento estão em conformidade com o paradigma de orientação a objetos. A Figura 2.1 mostra a arquitetura WBEM, com seus dois módulos principais: o gerenciador de objetos e o provedor de objetos. Para o WBEM foi definido um protocolo chamado HMMP (*Hypermedia Management Protocol*), que se responsabiliza pela comunicação entre

gerentes e agentes. Também foi criado um modelo de informação para os objetos gerenciados, o qual consiste de HMMS (*Hypermedia Management Scheme*); MOF (*Managed Object Format*) e CIM (*Common Information Mode*). O modelo funcional e organizacional fazem parte do CIM e do HMMP, ficando estes responsáveis por propósitos administrativos e de segurança respectivamente.

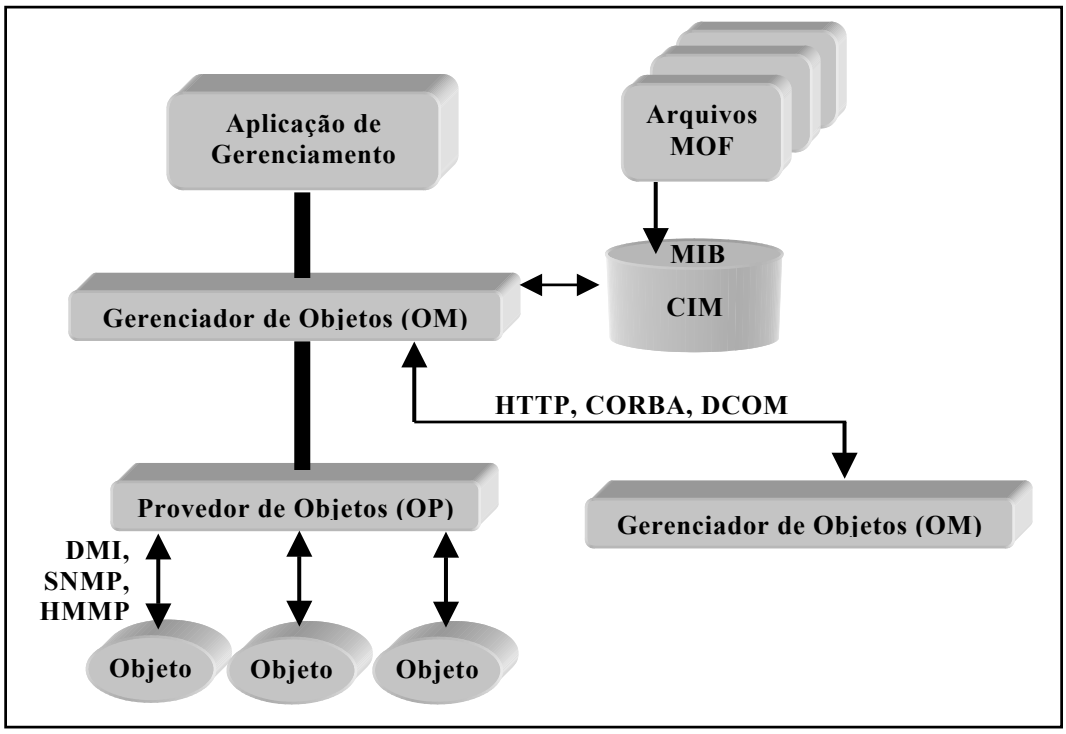

Figura 2.1 – Arquitetura WBEM.

O módulo gerenciador de objetos é responsável pela coleta e processamento de informações de gerenciamento, que são capturadas da MIB no formato CIM. O módulo gerenciador de objetos pode se comunicar com outros módulos deste tipo através de diversos protolocos, por exemplo, HTTP, DCOM e CORBA. Este módulo não trabalha apenas com processamento de informações, mas também fornece alguns serviços, tais como, correlação de eventos, ações corretivas automáticas e monitoramento preemptivo. As informações são disponibilizadas ao gerente da rede (ser humano) via navegador *web* ou através de ferramentas de gerência convencionais como a HP *OpenView*.

O módulo provedor de objetos atua como elo de ligação entre o módulo gerenciador e os recursos gerenciados, sendo este o responsável pela execução das ações no mundo real. A diversidade de protocolos usados pelo provedor de objetos possibilita uma visão uniforme de um gerenciamento em domínios heterogêneos.

O WBEM abrange todo o escopo do gerenciamento de redes de computadores, desde uma simples estação de trabalho até gerenciamento corporativo em escala completa. A sua arquitetura é escalável e distribuída, sendo o modelo de informação capaz descrever qualquer ambiente de gerenciamento existente. Por fim, WBEM é compatível com a maioria dos protocolos de gerência disponíveis, tais como, o SNMP e CMIP. O WBEM também fornece uma maneira de integrar ferramentas de gerenciamento usado tecnologias emergentes, entre elas, CIM e XML (*Extended Markup Language*) [27].

2.2.2 PBNM - Policy-Based Network Management

PBNM ou Gerenciamento Baseado em Políticas é uma nova proposta para diminuir a complexidade atual da gerência de redes de computadores. Complexidade esta, causada pelo contínuo crescimento das redes e surgimento de novas aplicações, que contribuem para o aumento do caráter heterogêneo das redes. O gerenciamento de recursos heterogêneos exige uma forma de gerência alternativa, em contraste com o modelo de controle e monitoramento que a gerência reativa oferece.

O conceito de políticas é baseado na idéia de que cada recurso ou processo da rede tem um papel e regras específicas de procedimento. A agregação de um conjunto de ações em um nível de abstração maior é denominada **Política**. As políticas, associadas à PBNM, definem um método eficaz de expressar o comportamento desejado de recursos e suportar esta complexa tarefa de gerenciamento, especificando meios que possibilitam forçar o comportamento desejado [25]. As ações definidas por uma política podem ser aplicadas diretamente por um operador ou administrador de rede, ou terem suas execuções automatizadas através de um sistema de gerência. Exemplos de políticas seriam, a saber, destinar ao setor de vendas alta prioridade de acesso ao servidor de banco de dados nos últimos dias do mês; se um aplicativo de fluxo de áudio exceder 60% do uso da largura de banda, então o coloque em baixa prioridade; caso se verifique uma perda crescente de

pacotes em um *link*, alocar um canal redundante para suprir eventuais falhas. Nos exemplos citados, e nos demais casos de uso de PBNM, em vez de configurar os dispositivos da rede individualmente, é feita uma análise de como os usuários e aplicações devem ser trados e posteriormente são estabelecidas políticas de uso e funcionamento da rede.

2.2.2.1 Arquitetura PBNM

Um sistema padrão de gerência baseado em políticas deve conter os seguintes elementos:

1. Console de Gerenciamento (MC – *Management Console);*
2. Repositório de Políticas (PR – *Policies Repository*);
3. Ponto de Decisão de Políticas (PDP – *Policy Decision Point*);
4. Pontos de Aplicação de Política (PEP – *Policy Enforcement Points*);

A Figura 2.2 mostra a arquitetura PBNM. O MC permite a criação de políticas que são armazenadas em um repositório de políticas (PR), ou seja, em um diretório para armazená-las junto com as informações dos recursos e/ou dos usuários de rede. O PDP é responsável pela interpretação, recuperação e detecção de conflitos relativos às políticas. Ainda são responsabilidades do PDP, o atendimento de eventos provenientes dos PEPs e a decisão de quais políticas serão aplicadas. Os PEPs são os objetos gerenciados. A comunicação entre os diversos módulos desta arquitetura se dá através dos seguintes protocolos:

- ✓ Protocolo de Acesso ao Repositório de Políticas (LDAP – *Lightweight Directory Access Protocol*) – É usado para armazenar e obter informações do repositório de políticas.
- ✓ Protocolo de comunicação entre o PDP e os PEPs – É utilizado para transmitir as requisições dos PEPs para o PDP e retornar a decisão do PDP. O SNMP e o COPS (*Common Open Policy Service*) são exemplos dos possíveis protocolos de comunicação utilizados.

É necessária a utilização de uma linguagem de definição de políticas (PDL – *Policy Definition Language*), a qual deve definir regras de sintaxe e semântica, de forma que possam ser interpretadas pelo PDP. As políticas são classificadas de diversas maneiras

dependendo de seu objetivo. Há políticas dinâmicas que descrevem como agir quando alguma restrição surge e políticas estáticas, que descrevem restrições a serem aplicadas em um momento específico.

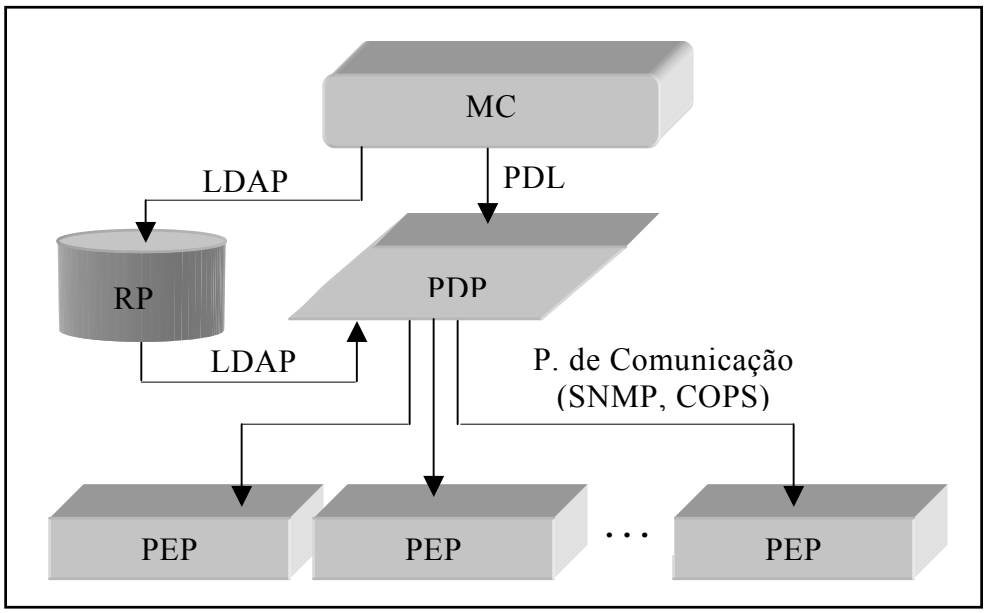

Figura 2.2 – Arquitetura de Gerência Paseada em Política.

2.3 O Ambiente RENATA

Um problema da gerência Internet é a centralização no gerente do processamento de informações e, conseqüentemente, do processo de tomada de decisão. Os agentes deste tipo de gerência têm capacidades computacionais limitadas e praticamente nenhuma autonomia. Uma alternativa para a solução deste problema é o uso de Agentes Inteligentes, uma vez que possuem maior autonomia e poder de processamento. Agentes Inteligentes podem também ser executados próximos ou no próprio objeto gerenciado, transformando em informação os dados coletados. Desta forma, há uma economia de banda passante, pois a troca de informação entre gerente e agente pode ser diminuída sensivelmente. Outro diferencial é a garantia de uma maior escalabilidade, pois parte da carga de tarefas destinadas aos gerentes, agora pode ser dividida entre os agentes.

Agentes pró-ativos tem como objetivo detectar situações anormais, comparando o estado atual da rede com um perfil de bom comportamento da mesma. Este perfil deve permitir ao sistema de gerência evitar problemas. Uma atitude pró-ativa compreende medidas preventivas ou reativas de menor impacto, dentro de uma ação planejada [16]. Técnicas de Inteligência Artificial podem ser usadas para viabilizar o caráter pró-ativo de um agente.

Dentre os mecanismos de Inteligência Artificial usados como máquina inteligente para agentes, as Redes Neurais têm se mostrado como uma tecnologia viável, motivadora de várias pesquisas [5] [8] [16] [17]. As exigências impostas pelas características dos agentes inteligentes, bem como pelas especificidades impostas pelo tráfego ATM, são requisitos solucionáveis pelas RNAs, devido às suas características de aprendizado, generalização, adaptabilidade, robustez e tolerância à falhas.

O RENATA é um ambiente de apoio a gerência ATM que utiliza agentes inteligentes baseados em RNAs com o intuito de prover ações pró-ativas. A Seção 2.3.1 fala sobre Agentes Inteligentes, uma vez que estão intimamente ligados ao RENATA. A Seção 2.3.2 descreve a arquitetura funcional do RENATA com explicação de cada um dos seus módulos. Nas Seções 2.3.3 e 2.3.4 é feito um breve relato de dois problemas solucionados pelo RENATA: a estimativa da capacidade requeri em comutadores ATM e apoio ao processo decisório no controle de admissão de conexões ATM.

2.3.1 Agentes Inteligentes

Agentes de softwares estão, cada vez mais, presentes nos vários tipos de aplicações. É o caso dos agentes aplicados ao gerenciamento de redes, bem como os agentes aplicados a ambientes de bases de dados Internet, sistemas de arquivos e automação. O ponto em comum entre esses vários tipos de agentes é que eles têm a capacidade de perceber eventos do ambiente (uso de sensores) em que estão inseridos e tomar alguma atitude em relação a um determinado evento. A Figura 2.3 traz o esquema de uma agente comum, em que o mesmo responde, de uma forma simples, a uma ocorrência de um evento predeterminado.

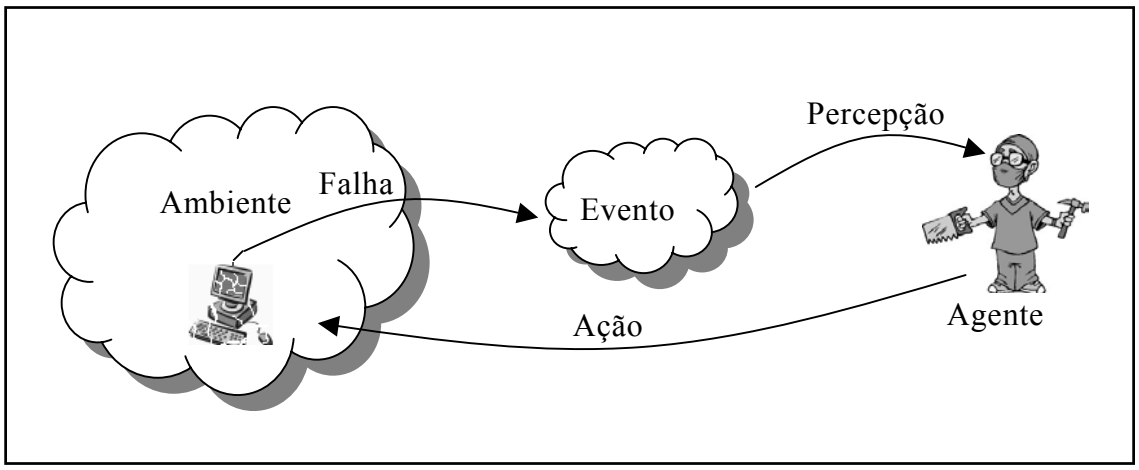

Figura 2.3 – Agente x Ambiente.

Há três dimensões que devem ser consideradas para caracterizar os agentes de softwares:

1. Ação (*Agency*) diz respeito ao grau de autonomia que o agente possui em relação ao usuário ou a outros agentes. Um agente pode representar o usuário, ajudá-lo, guiá-lo ou até mesmo tomar decisões em seu lugar;
2. Inteligência se refere a capacidade do agente assimilar ou adquirir conhecimento de domínio específico e utilizá-lo na solução de problemas de mesmo escopo. Assim, podem ser simples regras predefinidas ou técnicas avançadas de Inteligência Artificial;
3. Mobilidade trata da maneira de locomoção dos agentes em uma rede de computadores. Agentes móveis devem ter permissão para serem executados em diversos pontos da rede, o que acarreta preocupação com a segurança da mesma, bem como restrição nas permissões do agente em termos de acesso à rede e execução de tarefas.

Agentes aplicados ao modelo clássico de gerenciamento de redes são, por padrão, baseados em procedimentos e se assemelham a simples programas. Estes, na maioria das vezes, coletam informações dos objetos gerenciados e repassam aos gerentes, os quais realizam operações sobre os agentes segundo ordens de um ser humano. Tais agentes não possuem autonomia e estão vinculados a uma entidade maior responsável pelo mecanismo de decisão. O modelo de gerenciamento Gerente x

Agente utiliza os agentes simples e centraliza a tomada de decisão no gerente. Isto sobrecarrega o mecanismo de tomada de decisão do gerente na mesma proporção do crescimento da rede ou, em algumas vezes, do tráfego, influenciando assim, nas características de escalabilidade da mesma. Os protocolos utilizados nos modelos clássicos de gerenciamento de redes apresentam deficiências, uma vez que a capacidade computacional dos agentes é bastante limitada e o processo de extração de informação de gerência está centralizado nos gerentes. Estes gerentes são sobrecarregados facilmente. Além disso, há a questão do pouco poder das instruções dos agentes deste paradigma, quase limitadas a instruções de GETs e SETs.

Os sistemas convencionais de gerência de redes não parecem capazes de resolver os problemas de complexidade, custo e escalabilidade intrínsecos às redes de computadores. Um caminho para a solução destes problemas é a distribuição de inteligência entre os componentes da rede de computadores [38]. Tal abordagem permite que a tomada de decisão fique mais próxima do objeto gerenciado liberando do gerente parte de sua carga de tarefas e transferindo as mesmas para o agente. Isso também permite uma maior agilidade nas soluções dos problemas, pois a tomada de decisão é mais rápida. Outra vantagem é a diminuição do consumo de banda passante, ocasionada pela autonomia dos agentes inteligentes, os quais não necessitam de comunicação intensiva com o gerente. Isto faz com que os gerentes não necessitem realizar tarefas em demasia, diminuindo seu custo computacional e operacional.

Agentes Inteligentes usam técnicas de Inteligência Artificial para prover mecanismos de tomada de decisão, que podem se basear em regras pré-definidas (os mais simples) ou em máquinas de inferência (os mais complexos). Desta forma, ele age como um processo de software ativo ou autônomo capaz de resolver tarefas específicas em seu escopo de atuação. Agentes Inteligentes constituem uma solução para prover distribuição de mecanismos autônomos e inteligentes aos diversos pontos da rede de computadores a ser gerenciada.

A Inteligência Artificial e o paradigma de orientação a objeto oferecem facilidades para construção dos referidos agentes. O poder de encapsulação de um objeto do tipo agente permite que o mecanismo de inteligência fique bem protegido de operações indesejáveis e o processo de instanciação dos objetos agentes a partir de

uma classe facilita a criação dos mesmos. Normalmente, agentes inteligentes são implementados em linguagens orientadas objeto, podendo utilizar arquiteturas de Objetos Distribuídos. A Figura 2.4 mostra um esquema de agente inteligente. Percebe-se que a interação com o ambiente é feita de forma mais otimizada, devido a possíveis inferências realizadas e autonomia adquirida.

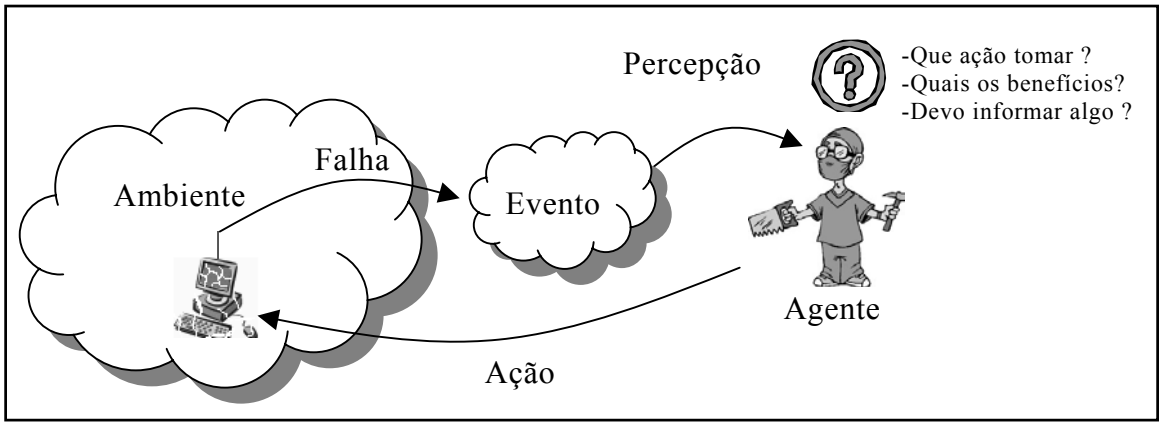

Figura 2.4 – Agente Inteligente x Ambiente.

Dentre as linhas da Inteligência Artificial que dão suporte a máquina inteligente dos agentes de software, se destacam o ramo simbólico e o ramo conexionista. O ramo simbólico possui como ferramenta básica para manipulação do conhecimento as regras de inferência da lógica clássica, enquanto a inteligência artificial conexionista usa redes neurais artificiais para este fim.

As RNAs se mostram como alternativa de resolução para problemas de difícil solução, ou seja, problemas que não possuem uma equação matemática que os governe ou possuem solução algorítmica de complexidade intratável para a tecnologia atual de computadores. Estes últimos problemas não completamente solucionados, mas as RNAs podem apresentar soluções de subproblemas (aplicados a um número restrito de casos) menores dos mesmos. Para que as RNAs possam atuar satisfatoriamente em tais problemas, eles devem ser bem definidos em termos de exemplos assim, estes exemplos poderão servir de base de aprendizado para as RNAs. Entre os campos de aplicações para as redes neurais artificiais destacam-se

reconhecimento e classificação de padrões, controle de processos industriais, robótica e predição de séries temporais. O Anexo A traz um tutorial que explica passo a passo como redes neurais artificiais podem ser aplicadas em diferentes domínios e deve ser consultado, caso haja necessidade de melhor entendimento sobre as mesmas.

2.3.2 Arquitetura Funcional do RENATA

Dentro do escopo de desenvolvimento de agentes inteligentes que possam detectar antecipadamente situações problemáticas de uma rede ATM, o ambiente RENATA propicia a criação de vários padrões, tanto de bom, quanto de mau funcionamento, de uma rede ATM em particular. Estes padrões são usados no treinamento das redes neurais artificiais, que serão as máquinas inteligentes destes agentes.

A Figura 2.5 traz a arquitetura funcional do ambiente RENATA, que possui basicamente três módulos: Módulos de Treinamento, Módulo Neural e Módulo de Gerência. Esta arquitetura propicia um mecanismo seqüencial para geração de agentes inteligentes de tecnologia conexionista.

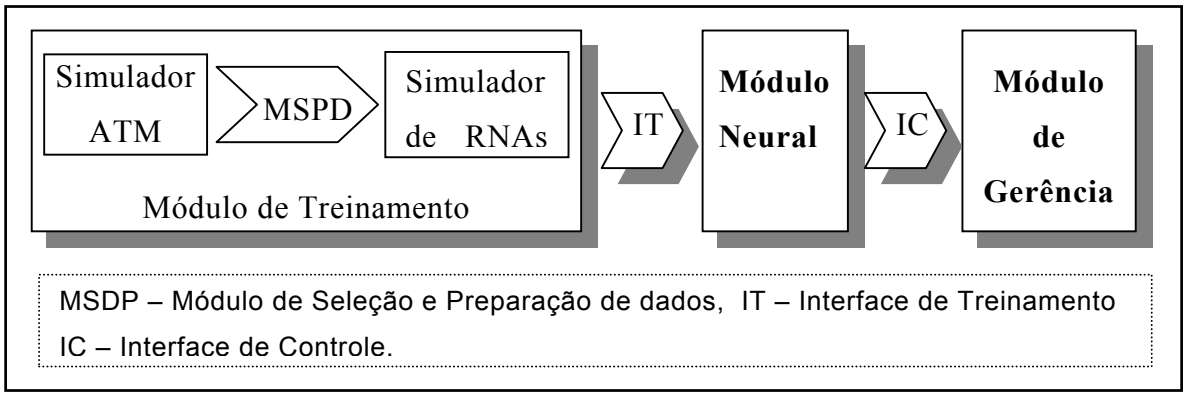

Figura 2.5 – Arquitetura Funcional do RENATA.

> "*No Módulo de Treinamento, a rede neural é projetada, treinada e validada. Para isto,* este *módulo é dividido em três componentes: o Simulador ATM, o MSPD (Módulo de Seleção e Preparação de Dados) e o Simulador de Redes Neurais. O Módulo Neural consiste da rede neural resultante da saída do Módulo de Treinamento e de informações sobre*

sua arquitetura e objetivo. O Módulo de Gerência é responsável pela integração e ativação do Módulo Neural, através do desenvolvimento de um agente que fornece dados para a predição da rede neural que, de acordo com sua saída, toma determinadas ações" [16].

2.3.2.1 Módulo de Treinamento – Simulador ATM

As redes neurais, necessitam de padrões de treinamento para que possam "aprender". Em redes de computadores, muitas vezes, há uma certa dificuldade de obtenção de tais padrões. Se não houver registros históricos do funcionamento da rede a ser gerenciada, fica praticamente inviabilizada a obtenção de padrões de treinamento. Não se pode justificar a parada de uma rede em produção para obtenção dos comportamentos críticos e anormais da mesma. Logo, se justifica a necessidade de um simulador de redes ATM. Simuladores deste tipo suprem a dificuldade de obtenção dos dados, pois proporcionam, de maneira muito simples, a simulação das mais variadas formas de topologia, carga, tipos de tráfego e falhas.

De acordo com o problema em questão, determinadas opções de *logs* do simulador ATM são habilitadas de modo que este gere os dados que servirão de base para o treinamento da rede neural. Estes dados devem corresponder aos parâmetros que serão posteriormente monitorados por um agente em uma rede ATM real.

Devido à grande complexidade para desenvolver uma ferramenta para simular redes ATM e também à necessidade de se economizar trabalho e tempo foi adotada uma ferramenta já existente para este fim. O ambiente RENATA usa o simulador de redes ATM denominado NIST, que foi desenvolvido pelo *National Institute of Standards and Technology* (NIST) [15]. Este simulador permite que o usuário crie diferentes topologias e configure parâmetros de operação de cada componente. Enquanto a simulação é executada, várias medidas de desempenho podem ser mostradas na tela ou salvas em arquivo para análise posterior.

2.3.2.2 Módulo de Treinamento – MSPD

Antes de serem submetidos à rede neural, os dados precisam ser selecionados, divididos, escalonados e testados. Essas funções são realizadas pelo MSPD (Módulo

de Seleção e Preparação de Dados). Segundo [5], aproximadamente, 98% destes dados devem ser de funcionamento normal e 2% devem caracterizar as situações que a rede neural deve detectar. Em seguida, os dados devem ser separados na proporção de 80% para treinamento e 10 % para testes e o restante para validar a rede neural. Esta partição dos dados de treinamento reflete a experiência adquirida por [5], assim tal modelo deve ser encarado como uma sugestão, pois não existe argumento científico para este fim. Há RNAs que só aceitam entradas binárias, outras, reais na escala de 0 a 1 ou de –1 a +1. Nestes casos, técnicas de normalização, escalonamento e codificação binária 1-N devem ser utilizadas.

2.3.2.3 Módulo de Treinamento – Simulador de Redes Neurais

A função do simulador de RNAs é especificar o modelo da rede neural e controlar seu treinamento. Parâmetros como taxa de aprendizado, função de ativação e outros parâmetros específicos de cada modelo devem ser configurados. Tais parâmetros determinam, por exemplo, a velocidade de treinamento e o grau de generalização. O aprendizado da rede deve ser acompanhado no sentido de verificar se a taxa de erro está posicionada dentro de um intervalo de valores adequados ao problema em questão ou se algum parâmetro de treinamento não foi bem escolhido. Por fim, a saída do Módulo de Treinamento é a rede neural treinada e testada. A comunicação entre o Módulo de Treinamento e o Módulo Neural é feita pela Interface de Treinamento, que transmite os dados resultantes do simulador de RNAs (matriz de pesos da RNA) para o Módulo Neural. Esta matriz de pesos representa o conhecimento adquirido durante o processo de treinamento.

O Ambiente RENATA usa o SNNS (*Stuttgart Neural Network Simulator*) [22], um simulador desenvolvido na Universidade de Stuttgart. Este simulador pode ser usado para criar, modelar, treinar, realizar podar, analisar e visualizar redes neurais de diversos modelos.

2.3.2.4 Módulo Neural

O Módulo Neural consiste da rede neural resultante do Módulo de Treinamento e de informações sobre sua arquitetura e objetivo. O desempenho da rede em questão

deve ser monitorado de tempos em tempos, com o intuito de verificar se seus resultados estão abaixo do limiar esperado. Se tal limiar não for atingido, possivelmente a dinâmica da rede de computadores tenha mudado (serviços e topologia), então a rede deve ser novamente treinada. Entretanto, pequenas alterações devem ser absorvidas pela capacidade de generalização das redes neurais.

A comunicação entre o Módulo Neural e o Módulo de Gerência é feita pela Interface de Controle, esta repassa informações relativas à rede neural (arquitetura e propósito), tais informações serão usadas no desenvolvimento do agente.

2.3.2.5 Módulo de Gerência

O Módulo de Gerência é responsável pela integração e ativação do Módulo Neural, através de um agente inteligente que acionará a rede neural e tomará as devidas ações de acordo com suas predições, visando uma atitude pró-ativa. Antes do desenvolvimento do agente é necessário que seja configurado no Módulo de Gerência o ambiente de gerência ATM em questão, ou seja, os dispositivos que serão monitorados e seus mecanismos (MIBs, SNMP, ILMI e Células OAM).

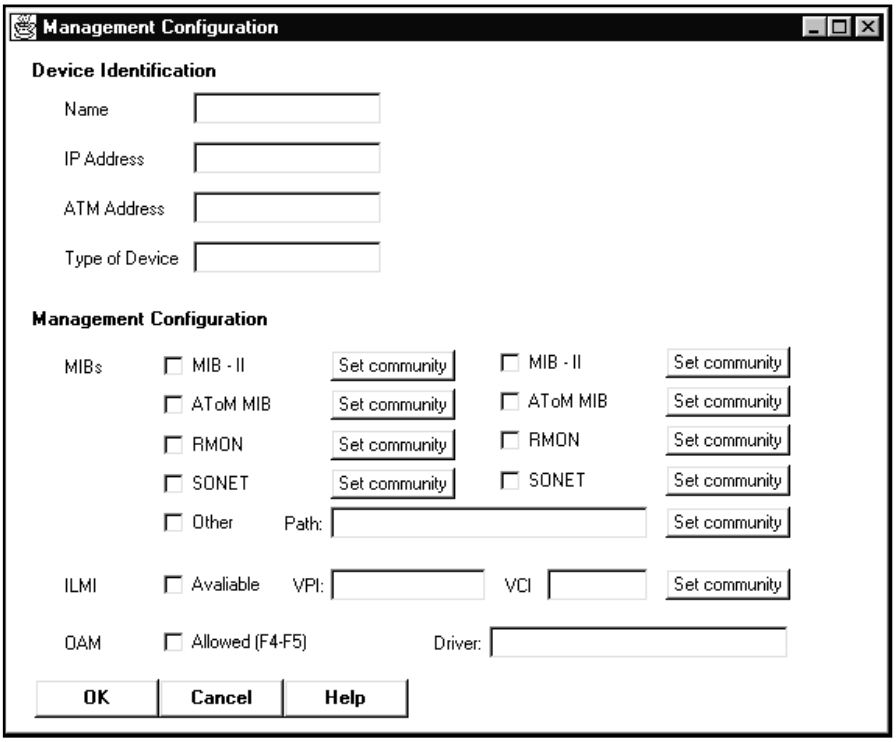

Figura 2.6 – Configuração da gerência do dispositivo.

A Figura 2.6 mostra a janela do Módulo de Gerência RENATA, sendo este módulo responsável pela configuração da gerência do dispositivo. Outras funções deste módulo são o monitoramento dos agentes e o gerenciamento da rede ATM. Os agentes devem informar se a rede neural está abaixo do limiar de aceitação para que o Módulo de Gerência tome as providências necessárias. Todas as informações fornecidas pelos agentes devem ser armazenadas, pois podem ser usadas para análises posteriores e também por documentarem a dinâmica do funcionamento da rede ATM. Os passos para o desenvolvimento de um agente RENATA são mostrados na Tabela 2.1.

Passo	Ação
1.	Definição do Problema;
2.	Simulação de Rede ATM;
3.	Preparação e Seleção dos Dados Gerados pelo Simulador ATM;
4.	Projeto, Treinamento e Validação da Rede Neural;
5.	Documentação da Rede Neural;
6.	Configuração dos Mecanismos de Gerência da Rede;
7.	Construção do Protótipo do Agente;
8.	Instalação e Ativação do Agente; e
9.	Monitoramento do Agente.

Tabela 2.1 - Passos para o desenvolvimento de um Agente RENATA.

2.3.3 Cálculo da Capacidade Requerida de Comutadores ATM

A tecnologia ATM propõe manter a qualidade dos serviços prestados em canais multiplexados, mesmo que as fontes dos dados que trafegam neste canal sejam de naturezas distintas. Desta forma, uma conexão só deve ser aceita se a rede dispuser de recursos suficientes para manter seu bom funcionamento. Fontes de tráfego constante, como as do tipo CBR (*Constant Bit Rate*), são bem conhecidas e seus comportamentos ao longo do tempo de transmissão podem ser previstos sem maiores problemas. Fontes

de tráfego variável, como as do tipo VBR (*Variable Bit Rate*), são mais complexas de serem caracterizadas devido a sua natureza estocástica. Algoritmos como o *Equivalente Bandwitdth* (EB) e a Aproximação Gaussiana tentam definir uma quantidade mínima de largura de banda que deve se alocada a uma fonte de tráfego de modo a satisfazer os parâmetros de QoS estabelecidos pela rede. Esta magnitude de banda é denominada de Capacidade Requerida.

É descrito em [8] como o RENATA foi usado para calcular o valor da capacidade requerida de uma fonte de tráfego VBR ON-OFF em comutadores ATM. O tipo de rede neural escolhida foi o **perceptron multicamadas** com algoritmo de aprendizagem *Backpropagation with Momentum*. A topologia consistia de três camadas de neurônios (entrada, intermediária e saída). O número de neurônios de entrada varia conforme o parâmetro história[11] e os valores de entrada estão baseados nos parâmetros citados abaixo:

- ✓ $N_{máx}$ – Capacidade máxima de conexões do comutador;
- ✓ ξ – Capacidade do buffer do comutador;
- ✓ N – Número de fontes de tráfego advindas dos enlaces de entrada;
- ✓ n – Fonte de Tráfego VBR ON-OFF caracterizada por:
- ✓ P_n – Taxa de Transmissão de Pico;
- ✓ T^{on} – Tamanhos Médios de Períodos Ativos e
- ✓ T^{off} – Tamanhos Médios de Períodos inativos.

O número de neurônios da camada intermediária deve variar em busca de uma maior precisão da rede neural. A camada de saída deve ter apenas um neurônio que fornece o valor da capacidade requerida.

2.3.4 Controle de Admissão de Conexões de Redes ATM

Em redes ATM quando se tenciona estabelecer uma nova conexão deve-se decidir se esta será aceita. O *controle de admissão de conexões* se baseia em certo critério para bloquear o estabelecimento de uma nova conexão quando a sua

[11] O parâmetro história se refere à janela de tempo durante o qual o sistema é observado para que a rede neural possa coletar informações suficientes para compor seu vetor de entrada.

característica de tráfego e qualidade de serviço oferecem perigo de congestionamento para a rede [39]. A definição do critério de bloqueio é de responsabilidade da operadora da rede. A abordagem baseada na soma da taxa de pico de transmissão gerada por aplicações que usam um determinado *link* garante o não congestionamento do mesmo, embora possa subutilizá-lo. O critério de aceitação deve tentar encontrar um ponto de equilíbrio entre a probabilidade de bloqueio e o uso eficiente dos recursos da rede.

Este problema foi tema do estudo de caso da dissertação de mestrado em [16]. No escopo desta dissertação os agentes gerados decidem sobre a aceitação de uma conexão VBR em redes ATM. Tais agentes usam como máquina de inteligente uma rede neural 5-20-1 (5 neurônios de entrada, 20 intermediários e um na saída) onde os parâmetros de entrada são: número da VCCs ativos, número máximo de VCCs, banda requisitada pela conexão, banda disponível na porta e taxa de ocupação do buffer. A saída da rede indicava se a conexão deveria ou não ser aceita. Tal estudo de caso visou testar a viabilidade e as funcionalidades do protótipo RENATA.

2.4 Conclusões

A complexidade para manter uma rede em bom funcionamento é diretamente proporcional ao binômio crescimento da rede / necessidade de novas aplicações. Crescimento este, em relação tanto aos meios físicos quanto ao tráfego. Novos modelos de gerenciamento, tais como, WBEM e PBNM têm sido propostos para suportar os novos desafios da gerência de redes. Antes do surgimento do WBEM e do PBNM, o Frame Relay e posteriormente o ATM eram tecnologias que tinham sido desenvolvidas para possibilitar avanços na infra-estrutura de transmissão das redes de computadores. O ATM trouxe avanços significativos em termos de largura de banda e garantias de qualidade de serviço e com isso teve início a popularização de serviços isócronos. Neste contexto, novas técnicas de gerenciamento eram necessárias para atender as especificidades do ATM, pois o modelo clássico de gerenciamento não se adequava satisfatoriamente às peculiaridades do mesmo. Assim, o ambiente RENATA surgiu como uma proposta inovadora para a gerência ATM.

As redes ópticas estão se popularizando e assim mais capacidade de transmissão de dados está sendo disponibilizada, trazendo novos benefícios e novos desafios. A arquitetura RENATA pode ser expandida para dar suporte a outras tecnologias de redes, tais como, IP, MPLS e GMPLS. O Capítulo 3 trata sobre Sistemas de Fibras Ópticas e o Capítulo 4 explica o problema do re-roteamento rápido em redes IP+GMPLS sobre DWDM.

Capítulo 3

Sistemas de Fibras Ópticas

Os sistemas de comunicação são responsáveis pela transmissão de informação de longo e curto alcance. Com o advento dos sistemas ópticos houve um avanço substancial nos meios de comunicação. Os sistemas ópticos operam em alta freqüência (aproximadamente 100 THz), no espectro da luz visível ou próximo da região infravermelha, o que lhes permite uma grande largura de banda e taxas de transmissão altas. Os sistemas ópticos de comunicação são também chamados de sistemas de transmissão de ondas de luz, em oposição os sistemas de microondas que operam em freqüências menores, próximos de 1 GHz. Fibras ópticas são cabos condutores ópticos que permitem o confinamento da luz em seu interior, proporcionando a transmissão de dados em certos tipos de sistemas ópticos, denominados de **Sistemas de Fibras Ópticas.**

Este capítulo apresenta o resultado de um estudo teórico sobre engenharia de sistemas de fibras ópticas, necessário para o desenvolvimento do Gerador de Perturbações (GDP). O GDP é uma ferramenta aplicada a simulação de falhas em *links* ópticos, que será descrita no Capítulo 6. É feita uma explanação dos elementos básicos dos sistemas de fibras ópticas onde os fatores mais relevantes que degradam a geração, transmissão e recepção dos sinais que trafegam nestes sistemas serão aborda  Na Seção 3.1 é descrito como um laser transmissor gera o sinal luminoso usado na transmissão de dados, com enfoque no comportamento deste sinal em termos de potência luminosa. As fibras ópticas, com seus tipos e fatores atenuantes do sinal que trafega por elas, são expostas na Seção 3.2. Na Seção 3.3 é feita uma explanação

sobre os receptores ópticos e como estes se comportam no reconhecimento dos dados recebidos. As considerações finais para este capítulo estão na Seção 3.4.

3.1 Lasers Transmissores

Os lasers são capazes de produzir luz com espectro de freqüência muito estreito, uma luz muito próxima da monocromática. Materiais como, o gasoso He-Ne (Hélio-Neônio) e o sólido rubi com 0,05% de cromo, possibilitam a emissão estimulada de radiação. Os lasers usam materiais desses tipos para controlar tal emissão e assim gerar sua luz peculiar. No seu interior, os lasers podem concentrar, por um período curto de tempo, luz com potencial energético muito elevado e controlar sua emissão de forma a produzir feixes de luz direcionados, com comprimentos de onda específicos e potências variadas. A maioria dos sistemas ópticos de comunicação usa lasers semicondutores como fontes ópticas, devido às características relacionadas acima, como também, por causa da sua superioridade em relação aos LEDs (Diodo Emissor de Luz) [2] .

3.1.1 Fatores que Influenciam no Sinal Emitido por um Laser Semicondutor

A estabilidade do comprimento de onda e da amplitude ou potência do sinal de um laser semicondutor é um importante fator para a transmissão eficiente de dados. Os principais fatores que influenciam na estabilidade do comprimento de onda e a potência do sinal de luz emitido por esses lasers são corrente, temperatura e tipo de material usado no laser. O gráfico da Figura 3.1 mostra a atenuação sofrida por um sinal óptico e o deslocamento do comprimento de onda, ambos devido a variação de temperatura. Geralmente, a estabilização da freqüência do sinal emitido por um transmissor laser é feita por meio de técnicas termoelétricas de resfriamento. Estas técnicas usam equipamentos que mantém a temperatura adequada ao funcionamento do equipamento e manipulam valores da ordem de fração de graus Celsius [10].

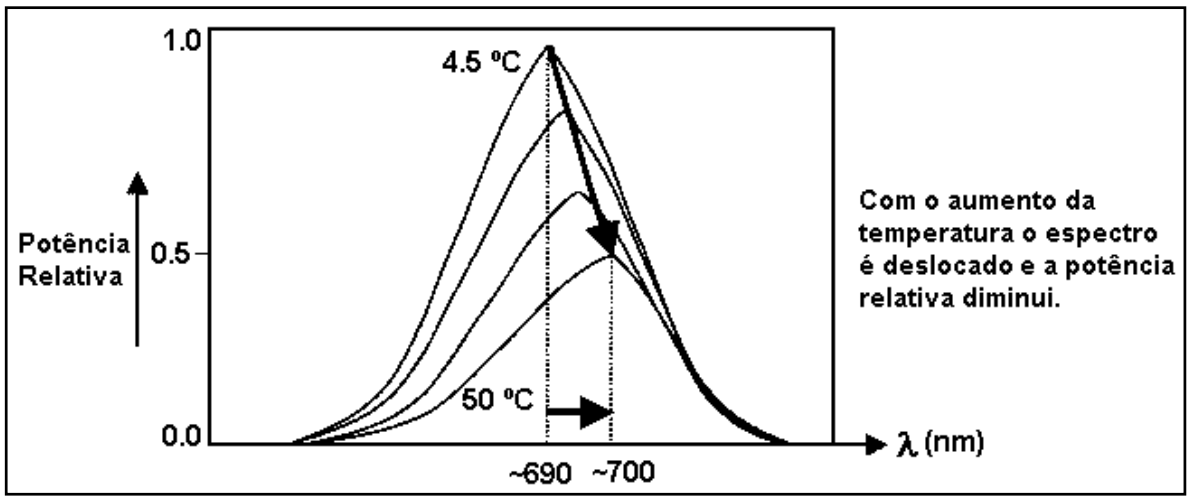

Figura 3.1 – Efeitos da variação de temperatura em um transmissor laser[12].

É necessário ultrapassar um certo limiar de corrente para que um transmissor laser possa iniciar seu ganho de potência. Verifica-se que o valor da corrente limiar é diretamente influenciado pela temperatura do equipamento. A potência emitida por um transmissor laser semicondutor é caracterizada por suas curvas L-I (*Light – Current*). A Figura 3.2 mostra as curvas L-I de um laser InGaAsP (Índio+Gálio+Arsênio+Fósforo) com comprimento de onda de transmissão 1,3 μm em uma faixa de temperatura de 10^o a 130^o C. À temperatura ambiente, o laser em questão necessita receber uma corrente superior a 20 mA para iniciar a emissão do sinal. Neste mesmo caso, fornecendo uma corrente de 100mA o laser emite um sinal de saída de aproximadamente 10 mW. O valor da corrente limiar I_{th} aumenta exponencialmente com a temperatura T, podendo ser descrito pela equação 3.1 de natureza empírica [2] [12] [14]

$$I_{th}(T) = I_0 \, EXP \, (T/T_0), \qquad (3.1)$$

na qual I_0 é uma constante, T_0 é o valor da temperatura característica dependente da estrutura do laser e dos seus materiais constituintes.

[12] Figura traduzida e retirada da página 20 de [10] .

Para lasers InGaAsP T_0 varia entre 50-70K e excede 120K para lasers GaAs (Gálio+Arsênio). Devido à alta sensibilidade à temperatura**,** lasers InGaAsP necessitam de equipamentos termoelétricos para manter a temperatura adequada. Tipicamente lasers desse tipo deixam de transmitir a 100 graus Celsius. A potência emitida P_e por um transmissor laser é descria pela equação 3.2 [2]

$$P_e = \frac{\hbar w}{2q} \frac{\eta_{\text{int}} \alpha_{mir}}{\alpha_{mir} + \alpha_{\text{int}}} (I - I_{th}), \tag{3.2}$$

na qual $\hbar w$ é a energia do fóton, η_{int} é a eficiência quântica interna, ou seja, a fração de elétrons que é convertida em fótons pelo laser (aproximadamente 100% para a maioria dos lasers semicondutores), I é a corrente de funcionamento do transmissor, I_{th} é a corrente limiar e o termo $\alpha_{mir}/(\alpha_{int}+\alpha_{mir})$ depende do tipo de material usado na confecção do laser e caracteriza as perdas sofridas no processo de geração e emissão de luz, tais como, absorção e dispersão, sendo α_{mir} relativo às perdas causadas pelo espelho do laser e α_{int} relativo às perdas totais.

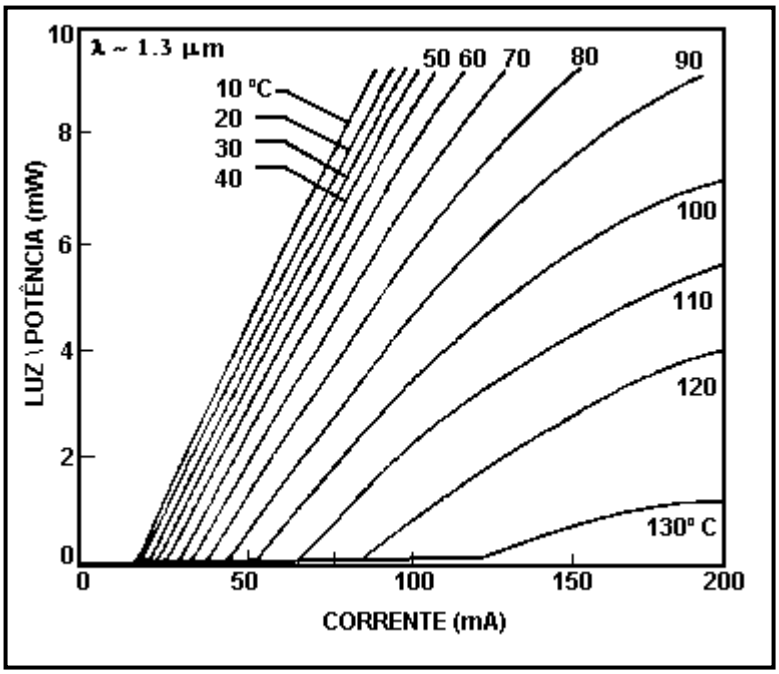

Figura 3.2 – Curvas L-I (Luz-Corrente) de um laser InGaAsP.

3.2 Fibras Ópticas

Uma fibra óptica consiste em um filamento de sílica ou plástico, por onde é feita a transmissão de um sinal de luz codificado no espectro infravermelho de freqüência (10^{12} a 10^{14} Hz). Ao redor desse filamento existem outras substâncias de menor índice de refração que provocam a reflexão interna do feixe de luz conduzido pelo mesmo. Isso garante a transmissão do feixe de luz, pois não o deixa escapar para fora do filamento, ou seja, do núcleo da fibra. A Figura 3.3 mostra o comportamento do sinal luminoso que trafega pelos tipos de fibras ópticas atuais.

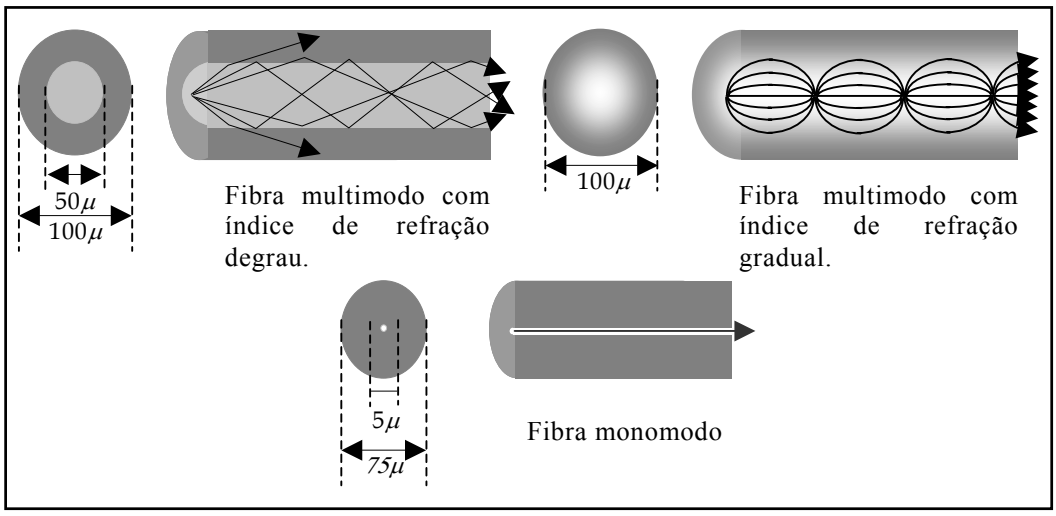

Figura 3.3 – Tipos de fibras ópticas com a representação física do sinal transmitido.

Há vários tipos de fibras ópticas otimizadas para diferentes taxas de transmissão e distâncias. As fibras podem ser classificadas em duas categorias principais: monomodo e multimodo. As fibras monomodo apresentam dimensões reduzidas, maior largura de banda e menor atenuação do sinal, se comparadas com as multimodo. O termo monomodo significa que o feixe de luz possui um único modo de propagação, sendo este paralelo ao eixo horizontal da fibra. As fibras multimodo propagam o sinal de luz em vários modos, de acordo com a variação de índices de refração do invólucro com relação ao núcleo. Estas se classificam em: índice degrau e

índice gradual. As fibras multimodo com índice gradual apresentam bandas passantes superiores as de índice degrau.

3.2.1 Atenuação Causada por Fibras Ópticas

A atenuação sofrida por um sinal óptico que trafega por um cabo de fibra óptica limita o tamanho deste cabo. Isto acontece devido à redução da potência média recebida pelo receptor óptico encarregado de receber o sinal luminoso da fibra óptica e transformá-lo novamente em sinal elétrico. Desta forma, deve haver um certo limite no tamanho do cabo para assegurar a correta interpretação dos dados pelo receptor. A atenuação causada pela fibra depende principalmente de mecanismos de dispersão, absorções causadas pelo material de confecção da fibra e flutuações no seu índice de refração. A forma de mensurar as perdas de um sinal que atravessa uma fibra óptica é feita através do seu coeficiente de atenuação α, o qual indica quantos decibéis do sinal é perdido por quilômetro que este percorre na mesma. O coeficiente de atenuação não inclui somente a absorção do material, mas também outras fontes de atenuação da potência [2]. Tais fontes de atenuação relacionadas com α são perdas causadas pela impureza presente na sílica do material de fibra e vibrações ressonantes associadas a moléculas específicas. Se P_{in} é a potência do sinal de entrada de uma fibra óptica de comprimento L e com coeficiente de atenuação α, o sinal de saída P_{out} é governado pela equação 3.3, mostrada abaixo

$$P_{out} = P_{in} EXP(-\alpha L) . \tag{3.3}$$

Como o coeficiente de atenuação α é geralmente expresso em unidade de dB/Km a relação 3.4 deve ser usada.

$$\alpha(dB/Km) = -\frac{10}{L}\log_{10}\left(\frac{P_{out}}{P_{in}}\right) = 4{,}343\alpha . \tag{3.4}$$

3.3 Receptores Ópticos

Fotodetectores são equipamentos que alteram suas características conforme a intensidade de luz que incide sobre eles. A principal função dos receptores ópticos é converter o sinal óptico de sua entrada em sinal elétrico, possibilitando a recuperação do sinal inicial e, por conseguinte, a transformação deste sinal em bits. Os requisitos necessários ao bom funcionamento de um equipamento receptor são semelhantes ao dos equipamentos transmissores, tais como, alta sensibilidade, resposta rápida, baixo ruído, alta confiabilidade e tamanho da entrada compatível com o diâmetro do núcleo da fibra óptica. Tais requisitos são mais fielmente atendidos pelos fotodetectores feitos de materiais semicondutores. Fotodetectores devem ser suficientemente rápidos para responder com uma saída elétrica confiável a excitação luminosa incidente sobre o mesmo. Em termos atuais a velocidade de transformação do sinal de luz em sinal elétrico pelos receptores está na faixa de picosegundos, o que vem atender os requisitos para a transmissão de dados nas redes de alta velocidade atuais.

Fotodetectores são caracterizados por certos parâmetros. Entre eles pode-se citar: espectro de resposta, fotosensibilidade, eficiência quântica, corrente gerada sem excitação luminosa (*dark current*), ruído, tempo de resposta, faixa de freqüência na qual o fotodetector é sensível (*frequency bandwidth*) [10].

A performance de um receptor óptico é mensurada pela sua taxa de bits errados BER (*Bit Error Rate*), sendo definida como a probabilidade de ocorrer um erro na identificação dos bits recebidos pelo circuito receptor. Assim, uma BER de 4 x 10^{-6} corresponde a uma média de 4 erros por milhão de bits. A maioria dos sistemas ópticos requerem uma BER de menos de 10^{-9}. A fotossensibilidade de um receptor óptico é definida como a potência média mínima requerida para operar a uma BER de 1 x 10^{-9} [2]. A expressão que calcula a taxa de bits errados (*BER*) de um receptor óptico ideal (sem ruído térmico, sem *dark current* e com 100% de eficiência quântica) é exposta na equação 3.5 [2]

$$BER = EXP\left(-N_p\right)/2, \tag{3.5}$$

sendo N_p definido como o número de fótons contidos no sinal recebido pelo receptor. N_0 significa o número de fótons contidos no sinal de bit em 0 e N_1, o número de fótons contidos no sinal de bit em 1. Como o receptor em questão se trata de um equipamento ideal não há *dark current,* tal fato permite que N_0 possa assumir o valor zero, ou seja, o bit 0 será reconhecido na ausência de sinal. N_1 deve, no mínimo, ser igual ao valor 20 para que o receptor possa garantir uma BER menor que 10^{-9}. Este valor para BER é requisitado pela maioria dos sistemas ópticos atuais. Esta restrição é resultado de flutuações quânticas relacionadas com a flutuação da incidência de luz no receptor, sendo denominado limite quântico. Para que um certo número de fótons possa excitar um fotodetector é necessário que a potência média do sinal excitante atinja um certo valor, Tal restrição é capturada pela equação 3.6

$$\overline{P}_{rec} = \frac{N_p hvB}{2} \ , \tag{3.6}$$

na qual $\overline{P}_{rec}$ é a potência média recebida pelo receptor, ou seja, $\overline{P}_{rec} = (p_1 + p_0)/2$, sendo p_1 e p_0 as potências representativas dos bits em 1 e 0 respectivamente, hv é a energia do fóton e B a taxa de bits por segundo a ser recebida. A partir da equação 3.6 é possível encontrar a potência média que deve excitar um fotodetector para prover um N_p = 20 (mínimo quântico). Em um receptor ideal que trabalhe com comprimento de onda de 1,55 μm (hv = 0,8 eV) e operando a uma taxa de 10 Gbit/s é necessário um sinal com potência de 13nW para obtenção de um N_p = 20. Em sistemas ópticos reais N_p deve exceder os dois mil fótons para produzir uma BER menor que 10^{-9}, uma vez que a performance desses sistemas é severamente afetada pelo ruído térmico [2]. A BER em função da potência média recebida e dada pela equação 3.7, que pode ser deduzida a partir das equações 3.5 e 3.6.

$$BER = EXP\left(-\frac{2\overline{P}_{rec}}{hvB}\right)\Big/2 \, . \tag{3.7}$$

3.4 Conclusões

Há diversos fatores que afetam o desempenho de um sistema óptico, por exemplo, fatores externos (temperatura e ruído), natureza do material usado na fabricação dos componentes e dimensões dos componentes. Foram descritos matematicamente os três principais componentes de um sistema de fibra óptica: transmissores, receptores e fibras ópticas. Este modelo matemático importante para engenharia de sistemas de fibras ópticas, pois descreve o comportamento dos mesmos. Outra aplicação para o modelo matemático, em questão, seria o uso de suas fórmulas em softwares que simulam sistemas ópticos. Simuladores deste tipo podem descrever o comportamento de um sistema óptico mediante fatores que afetam seu desempenho, gerando material de análise para engenheiros, bem como para os administradores destes sistemas.

Para a simulação de um sistema óptico real são necessárias variáveis que descrevam com precisão os vários tipos de ruídos relacionados a um sistema de fibra óptica. Estas variáveis devem ser aplicadas a fórmulas não expostas aqui. A investigação sobre um sistema óptico matemático que refletisse fielmente o mudo real e sua posterior implementação demandariam um esforço incompatível com o tempo disponível para a conclusão deste trabalho. Preferiu-se trabalhar com um sistema mais próximo do ideal sem fugir do escopo principal que é a busca por um modelo para sistemas ópticos. O Capítulo 4 trata do problema de re-rotemanto rápido em redes IP+GMPLS sobre DWDM e como as RNAs podem ajudar na solução do mesmo. Um simulador de sistemas de fibras ópticas pode fornecer os padrões de treinamento necessários ao aprendizado de tais RNAs.

Capítulo 4

Re-roteamento em Redes IP+GMPLS sobre DWDM

O IP tem sido estendido principalmente depois do crescimento quantitativo e qualitativo de aplicações multimídia. Segundo [11], MPLS é uma extensão do IP. A combinação do MPLS com redes ópticas provê mecanismos de engenharia de tráfego, tunelamento e criação de classes de serviços associados a uma largura de banda suficiente para transportar com facilidade informações isócronas. Um problema remanescente em redes ópticas é prover restauração rápida, especialmente, em ambientes distribuídos. Soluções proprietárias têm sido propostas para este problema, mas há dificuldades de interoperabilidade entre as diferentes implementações apresentadas pelos fabricantes [4].

DWDM e sua precursora WDM são tecnologias de transmissão de dados em ondas de luz através de fibras ópticas. A diferença básica entre elas é que DWDM usa amplificadores ópticos que operam a 1550 nm, o que lhe permite multiplexar mais comprimentos de onda em uma mesma fibra óptica (*Dense Wavelength).* Desta forma, estas duas tecnologias possuem o mesmo princípio, sendo apenas DWDM mais evoluída.

Este capítulo faz considerações sobre o re-roteamento de redes MPLS/GMPLS, sendo descrito na Seção 4.1. Os trabalhos correlatos estão na Seção 4.2 e as explicações de como redes neurais podem atuar na predição de falhas de *links* ópticos estão na Seção 5.3, ficando a conclusão deste capítulo na Seção 4.4.

4.1 Re-roteamento em Redes MPLS/GMPLS

Uma variante do MPLS é o MPL(*ambda*)S, o qual consiste de uma coleção de protocolos distribuídos usados para configurar caminhos e gerenciar conexões ópticas. GMPLS (*Generalised Multiprotocol Label Switching)* trata da generalização do MPLS, sendo aplicado em diferentes planos de controle de diversos tipos de redes e suportando comutação no domínio do tempo, comprimento de onda e espaço. O protocolo MPL(*ambda*)S está centrado no estabelecimento de conexões ópticas, porém não apresenta a funcionalidade de restauração das mesmas. O termo GMPLS é uma nomenclatura mais nova e engloba o termo MPL(*ambda*)S [21].

O grande desafio é proporcionar formas de restauração rápida de redes GMPLS, tal que sua performance seja melhor que a do SONET (*Synchronous Optical Network*) ou pelo menos equivalente. Uma das formas como o MPLS provê restauração para a camada IP é feita através de um LSP pré-alocado, que serve de *backup* para o LSP principal. Detectada a falha, o nó-fonte do fluxo de dados (fonte do LSP) usa seu mecanismo de re-roteamento. Em primeiro lugar, é descartado temporariamente o LSP *default*, sendo o fluxo de dados deste canal desviado para o LSP de *backup*. A Figura 4.1 exemplifica a forma de restauração proposta pelo MPLS. O tráfego entre D e A flui normalmente pelo LSP D-B-A (rota *default*), ao passo que o LSP de restauração (*backup*) é estabelecido através dos roteadores D-C-A. Enquanto a dinâmica do tráfego se dá normalmente, o LSP de *backup* permanece inativo, sem ocupar largura de banda, pois o IP sobre MPLS é necessariamente uma rede de comutação de pacotes. Quando ocorre alguma falha no *link* que liga D e B, o roteador D determina que o LPS *default* falhou e redireciona os pacotes para LSP de *backup.*

Outra forma de restauração oferecida pelo MPLS é o mecanismo de alocação redundante de subcaminhos*,* onde não é pré-alocado um LSP inteiro, mas somente um subcaminho que desvie o ponto onde ocorreu a falha. No caso da restauração de um LSP com rota D-B-A, uma falha ocorrida no subcaminho D-B seria contornada com o uso do subcaminho D-C-B, então o novo LSP seria D-C-B-A.

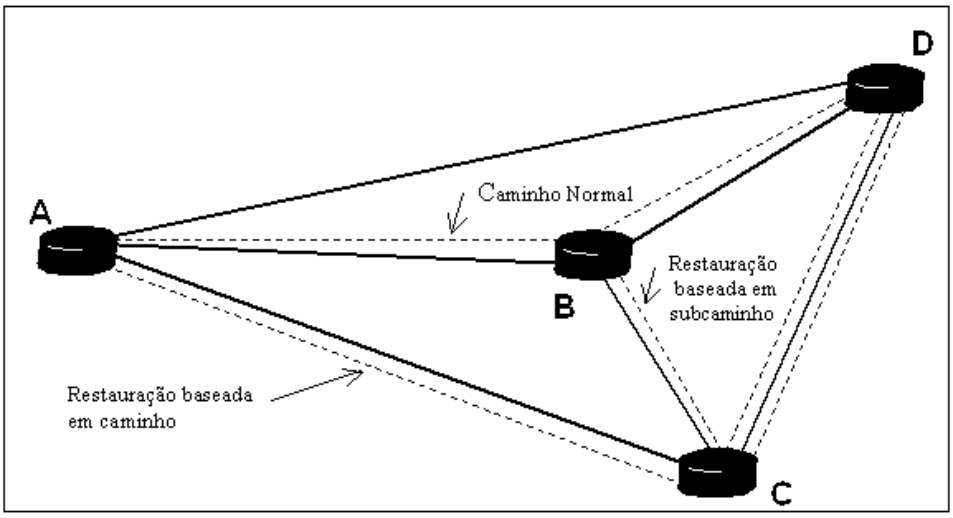

Figura 4.1 – Restauração de uma rede IP sobre MPLS.

4.1.1 Comutação de Pacotes em Redes Ópticas

Uma rede óptica consiste de múltiplas camadas. A Figura 4.2 mostra uma rede óptica de múltiplas camadas, são elas: camada IP, camada OXC (*Optical Cross-Connects),* camada WDM (*Wavelength-Division Multiplexing*) e a camada de fibra óptica. Na realidade, um *link* em uma camada superior se dá através de conexões em sua camada imediatamente inferior e assim sucessivamente. Um *link* óptico se dá através de um OTS (*Optical Transport System)*, neste caso, um par de terminais WDM que multiplexa/demutiplexa sinais ópticos transmitidos em diferentes comprimentos de onda ou freqüência [7]. Em um OTS também estão mecanismos de amplificação do sinal, onde se dá a ligação de dois sistemas deste tipo. Veja na Figura 4.2 que o *link* A-D na camada IP pode ser disponibilizado por uma conexão A-B-D da camada OXC, assim como a infra-estrutura da camada OXC é feita via camada WDM. Veja que o *link* A-C na camada de OXC e mapeado na camada WDM pela rota A-B-C e não há a interferência do OXC B. Este *link* é dito como sendo expresso via *office* B. Por fim a camada WDM é implementada sobre a topologia física real, ou seja, a camada de fibra óptica. Nesta camada, estruturas denominadas *spans* agrupam todas as fibras, sejam em cabos, conduítes, ou subestruturas entre dois pontos consecutivos. Observe que

apesar dos *links* na camada de fibra serem concentrados, estes são dispersos na camada IP.

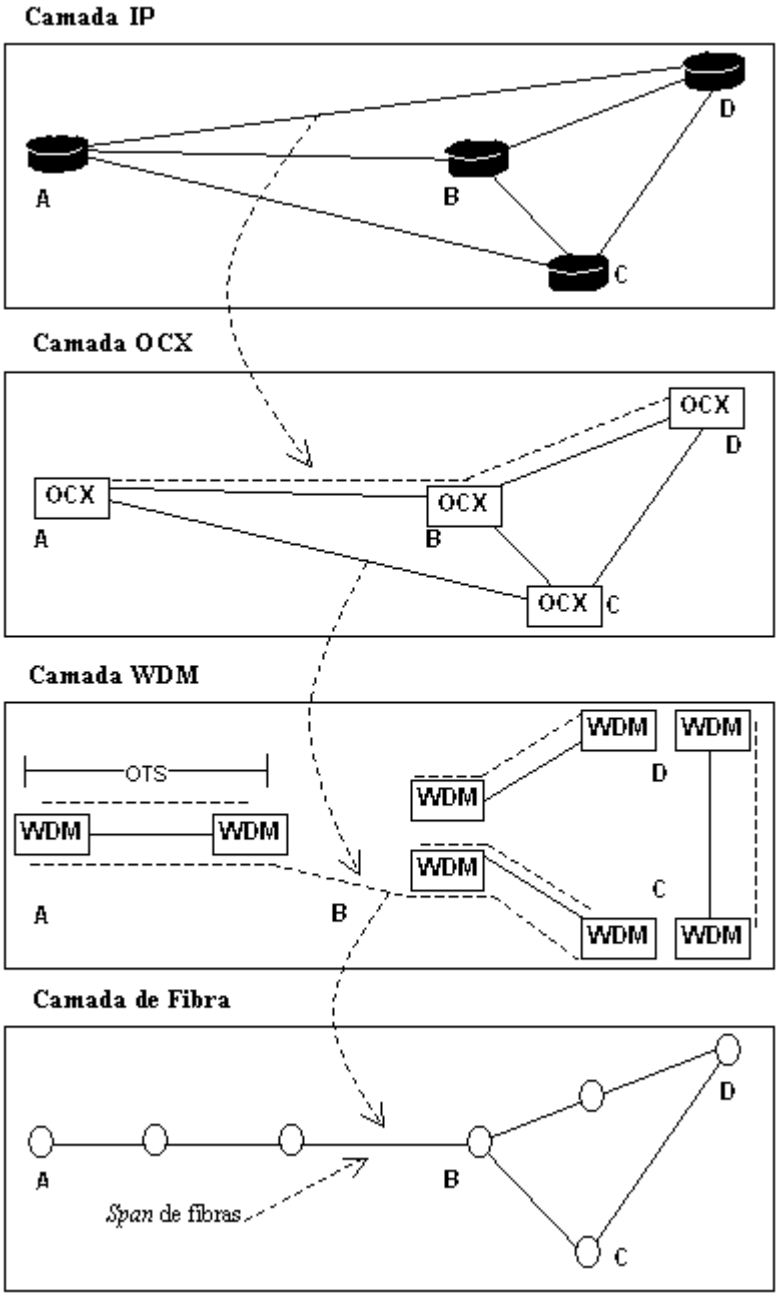

Figura 4.2 – Uma rede óptica em múltiplas camadas[13].

[13] Figura traduzida e retirada da página 91 de [7].

4.1.2 Seleção de Canal

Um OTS multiplexa vários sinais ópticos em uma única fibra, ou seja, diferentes canais são multiplexados em uma simples fibra. Logo não há possibilidade de se alocar uma conexão entre dois pontos sem que uma fatia de banda passante seja consumida, mesmo que não haja transmissão de dados. Depois do estabelecimento de um LSP em redes IP+MPLS, se nenhum pacote for transmitido, praticamente, não haverá consumo de banda. Esta é uma diferença importante entre redes ópticas e redes IP, que influencia de sobremaneira na capacidade de restauração de redes ópticas.

Na Figura 4.3.a tem-se a representação de um *link* WDM onde três canais foram alocados. Dois canais foram alocados para transmissão de dados (círculos e quadrados) e o terceiro para ser usado como um canal de *backup* de um *link* qualquer. Note que o canal de *backup* permanecerá em silêncio enquanto o canal *default* correspondente não necessitar do uso de uma rota alternativa. Neste caso, esta reserva implicaria em uma perda de 33,33% da banda passante total do *link*. No caso em que a comutação é feita via pacotes (Figura 4.3.b) não há segmentação do *link,* isso significa que, enquanto o canal de *backup* não for usado, 100% do *link* pode ser destinado aos demais canais.

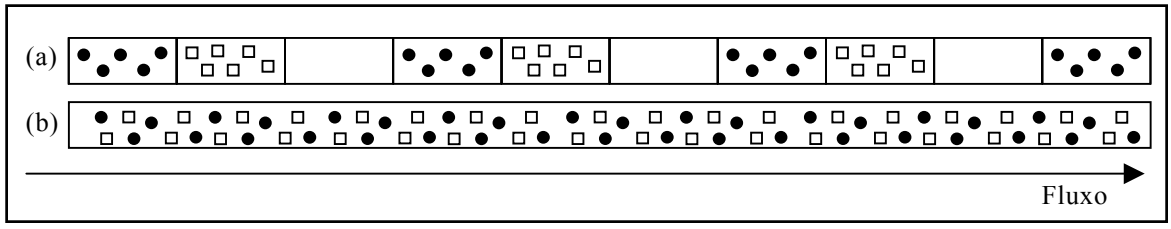

Figura 4.3 – WDM x Comutação de Pacotes.

4.1.3 Estabelecimento de Conexões Bidirecionais x Conexões Unidirecionais

A maioria das conexões orientadas a pacotes é unidirecional. Entretanto, terminais WDM, cuja função pioneira era prover suporte à tecnologia SONET/SDH requerem OTSs que operem bidirecionalmente. Extensões de MPLS para redes ópticas

deve trabalhar de forma a fornecer tal funcionalidade. Um problema típico em redes ópticas é conhecido como *glare*. Tal problema pode ser definido como a impossibilidade de um nó de fim de *link* associar ao mesmo canal bidirecional duas conexões provenientes de direções antagônicas. Esta situação não ocorre em conexões unidirecionais, pois qualquer nó controla todos os canais que passam por ele. O *glare* é mais um complicador para a restauração rápida de um *link*, pois sua ocorrência, na maioria das vezes, é verificada enquanto o processo de restauração está em andamento. Estabelecendo um dos nós de fim de *link* como o nó de controle e o outro como um nó de não-controle pode-se sugerir algumas soluções para este problema [7]:

- ✓ Solução 1: O nó não-controlador do canal deve abdicar do canal e tentar outro, deixando o canal corrente para o nó controlador da conexão. Isso pode ocasionar retornos e avanços sucessivos da mensagem associada à conexão do nó não-controlador.
- ✓ Solução 2: O nó de controle deve procurar por um canal de maior ordem possível, enquanto o nó não-controlador deve procurar por um canal de menor ordem possível. Conflitos ocorrerão se não houver pelo menos dois canais disponíveis, os quais devem ser ordenados.
- ✓ Solução 3: O nó não-controlador deve pedir ao nó de controle um canal para sua conexão.

4.1.4 Categorias de Restauração em Redes Ópticas

Existem diferentes métodos para prover restauração em redes ópticas, fato este que dificulta bastante a padronização dos mesmos. A Tabela 4.1 divide os diferentes métodos de restauração em quatro grandes categorias.

Segundo [4], a Categoria 1 seria melhor definida como uma técnica de proteção contra falhas e não como uma técnica de restauração de falhas. Nesta categoria, 100% dos recursos possuem mecanismos redundantes e a reação contra falhas ocorre quase que instantaneamente. Este tipo de proteção é chamada de 1+1, na qual os dados são transmitidos em paralelo, cabendo apenas ao nó de destino selecionar o melhor sinal. O único problema com este tipo de restauração é o seu altíssimo custo, pois cada canal de transmissão primário deve possuir o seu canal redundante.

Categoria	Cálculo do Caminho	Cálculo de Associações dos Canais	Conexões Cruzadas
1	Antes	Antes	Antes
2	Antes	Antes	Depois
3	Antes	Depois	Depois
4	Depois	Depois	Depois

Tabela 4.1 - Métodos para prover restauração em redes ópticas.

Nas demais categorias, os canais devem ser reservados para garantir que o tratamento de restauração será adequado. Estes canais podem dar suporte de reserva para vários caminhos de restauração, mas um canal não pode ser requerido de forma simultânea. Estes mecanismos de restauração são chamados de M:N, em que M é o número de canais de *backups* pré-alocados que podem ser compartilhados entre N caminhos primários ou de restauração. Há subcasos de restauração M:N bastante usados como o 1:N, no qual um canal de *backup* é compartilhado entre N caminhos primários e 1:1 em que um canal de backup é reservado para um único caminho primário. Note que a diferença entre os mecanismos 1+1 e 1:1 é que no primeiro o canal redundante é um espelho do principal, ou seja, os dados são transmitidos em paralelo, enquanto que no segundo caso, o canal de *backup* ficará ocioso até ser solicitado. A diferenciação entre os métodos é determinada pelo momento em que são realizadas as ações, antes ou depois da falha. As principais ações a serem tomadas são:

- ✓ cálculo do caminho (seqüência de OXCs);
- ✓ cálculo da associação do canal ao longo do caminho de restauração; e
- ✓ cruzamento das conexões ao longo no caminho de restauração.

O gerenciamento de falha consiste em identificar de forma inteligente as quatro etapas importantes do processo de tratamento de falhas: detecção, localização, notificação e atenuação. A melhor forma de detecção de falhas é fazer seu tratamento na camada mais próxima da ocorrência da mesma. Em redes ópticas deve-se procurar

sinais de falha na camada física. Parâmetros que podem medir falhas nestas redes são LOL (*Loss of Light),* OSRN *(Optical Signal-to-Noise Ratio)*, BER (*Bit Error Rate)*, *crosstalk*, dispersão e atenuação do sinal [4].

A localização da falha requer comunicação entre os nós para determinar onde a falha ocorreu. O protocolo LMP (*Link Management Protocol*) inclui um procedimento para localização de falhas, tanto em redes ópticas, como em redes eletro-ópticas. Isto é feito enviando mensagens entre nós adjacentes através de um canal de controle disjunto dos canais de dados. Tal separação permite a localização da falha independente do plano de controle de dados.

Uma vez detectada e localizada uma falha deve-se procurar mecanismos de atenuação da mesma, dentre estes mecanismos pode-se citar a proteção e restauração de falhas, já comentadas anteriormente.

4.2 Redes Neurais para Predição de Falhas em *Links* de Redes Ópticas

Como mencionado no capítulo introdutório, no final da década de 90, o modelo típico de rede de dados era baseado em quatro camadas: a camada IP para aplicações e serviços, ATM para engenharia de tráfego, SONET/SDH para transporte e DWDM como infra-estrutura física [3]. Para cada camada citada há várias camadas de gerenciamento, cada uma contendo suas próprias rotinas. Em particular, o gerenciamento de falhas é das tarefas mais redundantes, por ser tratado em todas as camadas. Nos equipamentos de transmissão pode-se obter informações, tais como, potência óptica e temperatura do equipamento. Na camada WDM, obtém-se informações sobre a qualidade do sinal óptico, como SNR (*Signal-to-Noise Ratio) e crosstalk* através de algum equipamento de teste como, por exemplo, um analisador de espectro. Em camadas superiores obtêm-se informações mais detalhadas acerca da qualidade do sinal, como a taxa de bits errados BER especifica para cada tecnologia de transmissão em uso.

A arquitetura em múltiplas camadas não tem se mostrado como uma solução satisfatória em relação a escalabilidade. Geralmente, há uma camada que limita a

expansão de toda a rede [1]. Outro fator determinante é o custo, seja de implantação e/ou de manutenção da rede. Uma tendência confirmada por [3] é a simplificação do modelo de quatro camadas, no qual IP/GMPLS utilizam o DWDM como infra-estrutura física de transmissão. Neste contexto, será feito um estudo dos equipamentos de uma rede de computadores com seus respectivos parâmetros, os quais podem ser candidatos a variáveis de entrada de uma RNA. O propósito da RNA é predizer um estado de falha em redes IP/GMPLS sobre DWDM. Tal rede neural servirá de máquina inteligente para agentes inteligentes do RENATA 2 que, por sua vez, servirão de monitores do tráfego e sinalizadores de eventuais problemas. Segundo [13], há duas classes de componentes de redes que podem sinalizar possíveis problemas: **componentes de hardware**, cujas falhas têm de ser identificadas, uma vez que degradam ou interrompem o sinal de transmissão e **equipamentos de monitoramento** que estão presentes em várias camadas para prover informações adicionais sobre a qualidade do sinal. Na próxima seção será feita uma breve descrição dos componentes de hardware com as variáveis candidatas à entrada da rede neural sublinhadas.

4.2.1 Equipamentos de Transmissão Ópticos

- ✓ **Fibras Ópticas**: meio físico usado para transmitir sinais ópticos entre dois pontos. Elas oferecem baixa atenuação, baixo custo e capacidade de transmissão simultânea de vários canais, usando para isso, diferentes comprimentos de onda. Não há parâmetro que uma fibra possa fornecer ao agente.
- ✓ **Transmissores**: são lasers ou vetores de lasers que convertem sinais elétricos em sinais ópticos de comprimentos de onda determinados. A resolução de um laser é o fator que determina o número de canais disponíveis de um meio WDM. Transmissores enviam alarmes nos quais temperatura e potência podem ser lidos, sendo parâmetros de seu estado.
- ✓ **Receptores**: convertem sinais ópticos de diferentes comprimentos de onda e os transforma em sinais elétricos. Estes enviam alarmes quando o sinal óptico de entrada está abaixo do nível aceitável.

- ✓ **Filtros *Add*/*Drop***: são capazes de adicionar ou retirar certos comprimentos de onda de sinais ópticos sem interferir nos demais componentes do sinal.
- ✓ **3Rs (*Regenerator*/*Reshaper*/*Retimer*):** amplificam o sinal de modo a evitar atenuações. Enviam alarmes se o sinal de entrada não puder ser restaurado.
- ✓ **Comutadores de Proteção:** recebem sinais ópticos replicados de dois canais diferentes (o canal primário e o canal de proteção) e selecionam um deles conforme o nível de potência. Caso uma comutação seja realizada, um alarme é acionado.
- ✓ **Multiplexadores / Demultiplexadores**: responsáveis por transformar vários sinais ópticos de diferentes comprimentos de onda em um sinal que contém todos os comprimentos de onda e vice-versa.
- ✓ **Comutadores**: responsáveis pelo cruzamento de conexões, conectam uma entrada em particular com uma saída específica. Sinalizam se uma conexão não puder ser estabelecida.
- ✓ **Amplificadores**: amplifica o sinal de saída, isto é, o sinal de saída é entregue a seu destino com um nível de potência maior que o sinal de entrada.

Os alarmes providos pelos equipamentos da camada física apontam falhas graves e muitas destas sem a menor possibilidade de predição como, por exemplo, o roubo ou corte de um cabo de fibra óptica, entretanto o aumento gradual de temperatura de um laser pode indicar uma possível falha deste equipamento.

4.2.2 Equipamentos de Transmissão WDM

Equipamentos que checam a qualidade do sinal de uma rede WDM podem ser divididos em duas categorias:

- **Equipamentos de Teste Individual**: podem fornecer parâmetros de teste de um simples canal, como por exemplo, fornecer a taxa de erro de bits (BER) de um canal. A desvantagem de se usar BER é que o equipamento de teste deve estar ciente da tecnologia de transmissão das camadas superiores (ATM, SONET, SDH, IP). Uma das vantagens do WDM é a transparência em relação às camadas superiores.

- **Equipamentos de Teste de Grupo**: qualifica de forma completa o sinal óptico, o que inclui comprimento de onda, potência máxima do sinal de cada comprimento de onda e o OSNR de cada canal.

4.2.3 Análise da Qualidade do Sinal a Partir da Camada IP

Na camada IP, roteadores podem ser considerados como equipamentos monitores, desde que os mesmos realizem o *checksum* dos pacotes antes dos mesmos serem transmitidos. Como candidatos a parâmetros de predição na camada IP pode-se citar:

- **Taxa de Erro de Bits (BER)** – O cabeçalho IP possui um campo chamado *header checksum* que permite verificar coerência erros no cabeçalho [19]. O *bit header checksum* não checa os dados anexados ao cabeçalho IP, ou seja, não checa todo o pacote, mas apenas o próprio cabeçalho. Antes da transmissão do pacote é feita a soma dos campos do cabeçalho IP e armazenado seu complemento no campo *header checksum*. Quando o cabeçalho passar por um roteador é feito novamente o cálculo do complemento do cabeçalho, então este é comparado com o *header checksum* já armazenado. Caso tal comparação não seja exata, implica que houve uma modificação no cabeçalho e o datagrama será descartado. Nenhuma mensagem de erro é gerada, ficando por conta das camadas superiores detectarem a perda do datagrama e requisitarem sua retransmissão. Descartes sucessivos de pacotes podem indicar falha nos equipamentos transmissores, logo o BER desta camada pode ser usado como um forte indicador de falhas na rede.
- **Tamanho da Fila de Roteamento** – de forma isolada, mudanças abruptas no tamanho da fila de roteamento não indicam uma possível falha dos meios de transmissão da rede. Uma parada brusca pode ser simplesmente ocasionada pelo final da transmissão e implicar na diminuição repentina do tamanho da fila. Mas analisada dentro de um contexto maior, por exemplo, a diminuição do tamanho da fila em adição aos descartes sucessivos de pacotes pode indicar falhas.

- **Tempo Médio de Chegada de Pacotes** - da mesma forma que o tamanho da fila de roteamento, o tempo médio de chegada de pacotes pode ser usado contextualmente para indicar falhas de transmissão na rede. Se os pacotes chegam ao roteador de forma constante e repentinamente, uma variação brusca no tempo de chegada é verificada, tal fato pode apontar problemas com o *link*, desde que outras variáveis como BER e tamanho da fila de roteamento também se comportem de forma anormal. Este fato pode ser verificado monitorando-se a utilização do enlace de transmissão por onde trafegam os pacotes. Se um canal é muito utilizado e passa a ficar em silêncio por um certo período de tempo, dependo do contexto (verificação de outras variáveis envolvidas), tal evento pode ser um indicativo de falha.

Conjugado a todos estes parâmetros, o fator tempo, que definido como o instante no qual os parâmetros supracitados foram coletados, deve ser levado em consideração. Os estados anteriores da rede formam o mecanismo que fornece o histórico de acontecimentos da mesma, tal histórico é vital para a predição proposta neste trabalho.

As RNAs [8] [16] são elementos computacionais que têm a capacidade de processar e extrair conhecimento através de experiência. Há uma necessidade de fornecer padrões do funcionamento do experimento que as redes neurais possam aprender a respeito do mesmo. Tais padrões devem fornecer os parâmetros de entrada da rede com os seus respectivos resultados, caso o algoritmo de aprendizagem destas seja do tipo supervisionado.

Para predizer situações problemáticas em um *link* de uma rede de computadores, os parâmetros de entrada da rede neural devem descrever o funcionamento do *link* em questão, ou seja, devem descrever o estado dos dispositivos de transmissão, bem como o tráfego. A saída da rede deve indicar se uma falha no *link* está prestes a ocorrer. A definição dos parâmetros usados deverá levar em consideração a disponibilidade dos mesmos. Dependendo dos recursos financeiros e de infra-estrutura disponíveis para a captura dessas variáveis ou parâmetros, a descrição do *link* será mais fiel na medida que as melhores vaiáveis forem escolhidas.

A potência do sinal óptico de entrada de um receptor da camada física de uma rede WDM seria um candidato forte a parâmetro de entrada da rede neural. Em termos de pesquisa básica, a captura de tais parâmetros pode ser feita através de simuladores, os quais devem fornecer arquivos de *log* com o resultado das simulações e posteriormente ested arquivos devem ser adequados ao formato de entrada requerido pelas redes neurais.

4.2.4 Predição de Séries Temporais

A tarefa de analisar uma série de parâmetros coletados em instantes diferentes e inferir o próximo valor pode ser realizada por um mecanismo de predição. As RNAs têm se mostrado efetivas quando usadas para predição de séries temporais [17]. Nesta seção será feita uma descrição geral do problema de predição. Maiores detalhes sobre este assunto podem ser encontrados em [45] [46].

Predição de séries temporais é um problema de processamento de sinais em que se tem uma seqüência de N amostras de uma determinada variável escalar, {y(t), y(t-1),...,y(t-N+1)}, uniformemente espaçadas no tempo, e cuja meta é obter uma estimativa, y'(t+1), para o próximo elemento da série. Uma possível formulação matemática para este problema supõe que a série observada foi gerada por um processo cuja evolução temporal pode ser descrita por um modelo ***auto-regressivo não-linear***:

$$y(t+1) = f[y(t), y(t-1),...,y(t-n_y+1)]+ e(t), \qquad (4.1)$$

no qual $1 \leq n_y << N$ é a ordem do modelo, e(t) é uma variável aleatória i.i.d., de média zero e variância σ_e^2, representando as imprecisões de medida e/ou modelagem. A função $f(\bullet)$ é o mapeamento matemático responsável pela dinâmica de geração da série. Em outras palavras, $f(\bullet)$ relaciona o passado e o presente da série com o seu futuro. As dificuldades com esta formulação são basicamente duas:

1. Não se conhece $f(\bullet)$ que pode ser inclusive não-liear.
2. Não se conhece a distribuição de probabilidade do ruído e(t).

Para efeito de simplificação, a abordagem mais comum assume que e(t) é uma seqüência de ruído branco gaussiano de média zero e variância σ_e^2. Portanto, o objetivo do problema de predição de séries temporais se restringe a encontrar modelos matemáticos que forneçam uma aproximação para a função $f(\bullet)$. Neste caso, tem-se a seguinte representação:

$$y'(t+1) = f'[y(t), y(t-1),...,y(t-n_y+1); M], \tag{4.2}$$

na qual $f'(\bullet)$ simboliza uma aproximação da função $f(\bullet)$ e M é o vetor ou matriz de parâmetros do modelo matemático sendo ajustado à série.

Assim, defini-se o *erro de predição* ou *resíduo* como a diferença entre o valor realmente observado para a próxima amostra da série e a estimativa y'(t+1) produzida pelo modelo em questão com base nas amostras passadas, ou seja:

$$e(t) = y(t) - y'(t), \tag{4.3}$$

tal que a variância dos resíduos, σ_e^2, pode ser vista como uma estimativa da variância do ruído brando, ou seja, $\sigma_e^2 = \sigma'^2_e$.

A seqüência de resíduos, {e(t)}, t = 1,..., N, é utilizada para avaliar a precisão do modelo por meio da raiz quadrada do "Erro Médio Quadrático" (MSE, em inglês), dado pela seguinte expressão:

$$MSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y(t) - y'(t))^2} = e(t), \tag{4.4}$$

na qual N é o tamanho da seqüência de resíduos.

4.2.5 Técnicas de Modelagem Global de Séries Temporais

Os modelos utilizados em predição de séries temporais divide-se em dois grupos principais: *modelos globais* e *modelos locais*. Nos métodos globais, apenas um modelo matemático é construído e utilizado para caracterizar a série observada. O

método global mais simples tem como protagonista o conhecido modelo *linear auto-regressivo* (LAR):

$$\text{y'(t+1)} = a_0 + a_1 y(t) + \ldots + a_{n_y} y(t - n_y + 1) = a_0 + \sum_{i=1}^{n_y} a_i y(t - i + 1), \tag{4.5}$$

em que a_i são os coeficientes do modelo, que juntamente com a ordem da memória, n_y, constituem os parâmetros do modelo. Geometricamente falando, predição de séries temporais via modelo LAR consistem em ajustar um hiperplano n_y-dimensional aos dados em $\Re^{n_y+1}$, sendo por este motivo classificado como um modelo linear global.

Existem várias técnicas para calcular os coeficientes a_i de um modelo LAR, sendo a mais comum a dos *Mínimos Quadrados* (MQ) [47]. De acordo com a técnica MQ, os coeficientes são calculados pela seguinte expressão:

$$a = (Y^T Y)^{-1} Y^T p, \tag{4.6}$$

em que $a = [a_0 \;\; a_1 \ldots \;\; a_{n_y}]^T$ é o vetor de coeficientes, p é o vetor de predição e Y é a matriz de regressão. Estes dois vetores e a matriz Y são dados por:

$$p = \begin{pmatrix} y(n_y+1) \\ y(n_y+2) \\ \vdots \\ y(N) \end{pmatrix}, \quad Y = \begin{pmatrix} 1 & y(ny) & \cdots & y(1) \\ 1 & y(ny+1) & \cdots & y(2) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & y(N-1) & \cdots & y(N-n_y-1) \end{pmatrix}. \tag{4.7}$$

Uma vez calculados os coeficientes, este são utilizados na equação 4.5 para estimar novos valores para a série. Apesar de sua simplicidade, este método pode apresentar problemas de instabilidade numérica devido à inversão de matrizes, principalmente para valores elevados de n_y. De qualquer modo, o uso do Modelo LAR com coeficientes calculados pelo método MQ está amplamente disseminado, servindo sempre como referência para estudos comparativos com outros modelos auto-regressivos, lineares ou não-lineares [48].

Uma rede MLP consiste em um conjunto de neurônios dotados de funções de transferência, geralmente sigmoidais, que operam sobre o produto interno do vetor de

entrada da camada com o respectivo vetor de pesos. Conforme o Anexo A, a rede MLP pode aproximar qualquer função suave com um grau de precisão arbitrário, desde que exista um número suficiente de camadas escondidas com número suficiente de neurônios não-lineares em cada uma delas. Em princípio, uma rede MLP pode ter qualquer número de camadas e qualquer número de neurônios em cada camada, mas não há uma técnica geral para calcular estes números. Em geral, eles são encontrados por tentativa e erro, após muita experimentação com a rede e os dados.

Segundo [17] a função não-linear $f(\bullet)$ pode ser estimada usando-se redes neurais do tipo MPL (*Multi-Layer Prceptron*) com algoritmo de aprendizagem *Recurrent Backpropagation.* Mas nesta dissertação obteve-se resultados satisfatórios usando-se o mesmo tipo rede neural, mas com algoritmo de aprendizagem *Backpropagation* simples, conforme exposto no Capítulo 6 e no Anexo A. Há uma vasta literatura a respeito deste tipo de redes neurais em [6] [9], o que facilitou a obtenção dos resultados desta dissertação. A Figura 4.4 mostra uma rede neural com os parâmetros de entrada e a resposta pressuposta.

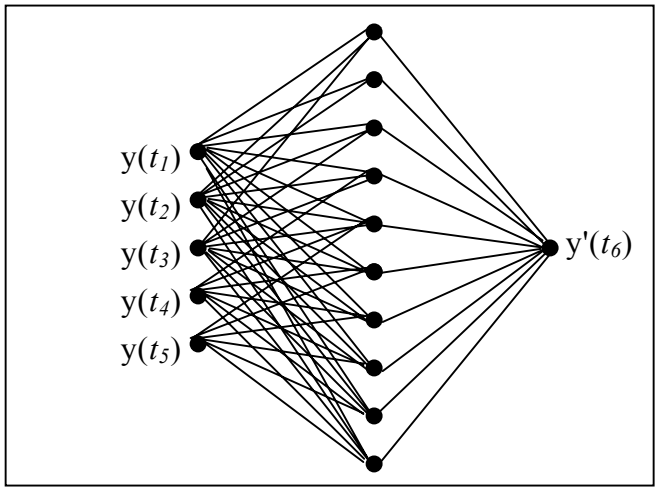

Figura 4.4 – Rede Neural 5-10- (entrada $y(t_1)$,..., $y(t_1)$; e saída y'(t6)).

O exemplo exposto na Figura 4.4 trabalha apenas com uma variável representada pela letra *y*. Se *y* fosse uma variável binária que significasse o descarte de pacotes (1 = pacote aceito, 0 = pacote descartado), então baseado nos valores de *y* nos instantes t_1, t_2, t_3, t_4 e t_5, o valor de *y* no instante t_6 poderia ser previsto com uma

margem de erro *e(t)*. A topologia da rede, a quantidade de valores de entrada e o valor da janela de tempo do histórico dos parâmetros são feitos de forma empírica até que se obtenha um grau de acerto satisfatório do ponto de vista do erro, seja de predição de valores futuros ou inferência de valores atuais. É necessária a generalização deste modelo para permitir o uso de uma função de várias variáveis, onde dois modelos podem ser usados: uma rede maior, que receba todas as variáveis ou um conjunto de redes menores interconectadas, onde cada rede trataria de um tipo de variável.

Os pesos sinápticos das redes MLP não se relacionam linearmente com as entradas, exigindo assim um método mais sofisticado de estimação dos pesos da rede. Na maioria dos casos, isto é feito pelo algoritmo de "retropropagação de erros" (*error backpropagation*), que é um método de busca estocástica via gradiente descendente do erro [49] [50]. Como a grande maioria dos algoritmos de aprendizagem que utilizam gradiente descendente, muito raramente a rede MLP encontra uma solução ótima global para os pesos, mas sim soluções subótimas (ou locais).

4.3 Trabalhos Correlatos

O uso de RNAs para detecção antecipada de falhas em enlaces de comunicação é uma tarefa complexa e possivelmente pouco explorada pela comunidade científica, uma vez que não são comuns artigos que discutam o assunto. De qualquer forma, foram encontrados trabalhos relacionados com RNAs que tentam predizer falhas em diferentes domínios tecnológicos.

Em [41], foi desenvolvido um sistema baseado em RNAs para predizer e detectar falhas, em estado inicial, de sistemas elétricos (linhas de transmissão). Nesse trabalho são feitas simulações do circuito equivalente de uma linha de transmissão e redes neurais artificiais *feed-forward* foram usadas como componente detector de falhas. O trabalho se concentrou em detectar ocorrências de pequenas falhas em uma rede de transmissão e alertar o operador do sistema em questão. Para isso, são capturadas informações sobre o *status* dos principais componentes da rede elétrica, por exemplo, transformadores e distribuidores. As informações coletadas são convertidas em parâmetros de entrada das RNAs. As falhas destacadas pelas RNAs se

apresentam mais relevantes em certas ocasiões climáticas. Por exemplo, a fuga de corrente de uma linha de transmissão em um inverno rigoroso pode progredir para uma falha mais grave se houver o acúmulo gradual de neve nesta linha.

Se as condições climáticas são tão importantes para o bom funcionamento dessa proposta, então seria relevante a inserção de variáveis relacionadas com o tempo climático (por exemplo, temperatura ambiente) na entrada destas RNAs. Tal fato poderia deixar o processo mais autônomo e por conseguinte, haveria menor necessidade de intervenção humana na interpretação dos resultados deste sistema. Um fato não mencionado em [41] foi o modo como o tratamento temporal dos dados coletados foi aplicado ao banco de conhecimento das redes neurais usadas. Esta é uma questão fundamental para predição de falhas usando esta abordagem. Apesar do domínio desse trabalho não estar diretamente relacionado com redes de computadores, o mesmo consiste em uma abordagem interessante (95% de acerto segundo o autor) para predição de falhas através RNAs e, neste aspecto, se correlaciona com o trabalho proposto por esta dissertação.

Abraham em [42], apresentou um comparativo de performance entre dois sistemas de monitoramento *online* de circuitos e sistemas eletrônicos. O objetivo de tais sistemas era predizer o momento da ocorrência de falha de equipamentos eletrônicos em ambiente de condições adversas (*stressor*). Tal ambiente influenciava diretamente no tempo de vida útil dos equipamentos eletrônicos estudados. O primeiro sistema usava ANFIS (*Adaptive Network Based Fuzzy Inference System*) e o segundo MLPs com algoritmo de aprendizagem *backpropagation.* A performance de aprendizagem do ANFIS foi superior a das RNAs e os resultados apresentados também. Entretanto, para a montagem do sistema ANFIS é necessária a intervenção de um especialista humano para construção das regras de inferência necessárias a metodologia *neuro-fuzzy* apresentada. Apesar de Abraham comprovar que o sistema ANFIS foi superior ao sistema baseado em RNAs, há aplicações em que o modelo ANFIS não poderá ser aplicado em virtude da falta de subsídios que caracterizem a natureza estatística do objeto modelado. No trabalho [42] a abordagem *neuro-fuzzy* foi possível devido a modelagem estatística realizada sobre o experimento.

O artigo [51] descreve o progresso de combinação de técnicas computacionais "*soft"* e "*hard"* aplicadas à identificação e prevenção de falhas em sistemas de telecomunicações. Técnicas de tratamento de falhas *soft* tem alvo na busca de soluções que possibilitem a um sistema de telecomunicação ser de baixo custo, de fácil uso e robusto. As técnicas *soft* estão baseadas em várias teorias, sendo que estas teorias podem ser combinadas para fornecer uma solução melhor para problema em questão. Dentre tais teorias, pode-se citar: RNAs, Lógica Fuzzy, Redes Probabilísticas (*Bayesian Belief Networks* - BBNs), Algoritmos Genéticos, Teoria do Caos e seções da Teoria da Aprendizagem. As técnicas de tratamento de falhas *hard* (convencional) usam basicamente os alarmes dos equipamentos físicos, tais como, multiplexadores e roteadores para diagnosticar o estado da rede de telecomunicações.

Uma falha pode disparar alarmes em cascata, gerando uma grande quantidade de novos alarmes que, possivelmente, podem ocultar o problema que realmente causou a falha. É essencial que uma rede possa prover mecanismos de *correlação de alarmes* para que o gerente da mesma possa identificar ou, até mesmo, prevenir falhas. Idealmente, o gerenciamento de falhas deve ser capaz de facilitar a tarefa de predição das mesmas. Segundo o trabalho [51], técnicas *soft* de tratamento de falhas podem ser aplicadas tanto para identificação como para predição de falhas. Dentre tais técnicas, esse trabalho explica como RNAs, BBNs, Algoritmos Genéticos e Técnicas de Computação *Soft* para o descobrimento de regras (*Soft Computing for Rule-Descovery)* podem realizar a tarefa de predição. BBNs e Algoritmos Genéticos são explicados com mais detalhes se comparados à explicação sobre RNAs e *Soft Computing for Rule-Descovery*, mas em todos os casos nenhum resultado concreto é mostrado, sendo o artigo em questão importante para descrever como estas tecnologias podem ser aplicadas ao gerenciamento de falhas.

Quanto a falhas em sistemas DWDM/WDM, Kartalopoulos em [10] apresenta um livro[14] que trata especifica e exclusivamente de falhas em redes DWDM. Carmen Mas em [13], desenvolveu uma algoritmo para localização de falhas de sistemas WDM. Ambos os trabalhos [10] e [13] apresentam técnicas de localização e

[14] Observa-se mais uma coletânea de artigos que propriamente um livro.

tratamento de falhas, mas não de predição. Vale ressaltar que estes dois últimos trabalhos foram elucidativos e inspiradores desta dissertação por apontarem aspectos importantes relacionados com sinais e equipamentos ópticos, sendo citados em diversos pontos desta dissertação.

Em particular, detecção de falhas em *links* ópticos envolve várias variáveis. O relacionamento entre estas variáveis é importante para predição de tais falhas. Encontrar uma forma de correlacionar as variáveis citadas é uma tarefa complexa, em que a modelagem através de métodos analíticos é de solução difícil. RNAs são capazes de capturar o relacionamento desconhecido das variáveis e fazer predições de falhas de um enlace óptico de uma rede de computadores.

4.4 Conclusões

O GMPLS poderá atuar melhor no plano de controle de redes IP sobre DWDM se melhorias em seu modo de re-roteamento após falhas forem realizadas. Segundo [17], RNAs podem operar satisfatoriamente em predição de séries temporais. Se o histórico do tráfego de um *link* óptico puder ser considerado como uma série temporal não-linear, então RNAs poderão prever falhas em um dado *link* desde que as variáveis que descrevem o comportamento do *link* possam ser fielmente capturadas. O trabalho conjunto de RNAs e agentes de *software* pode agilizar o processo de re-roteamento rápido de uma rede óptica, na qual os agentes seriam responsáveis por sinalizar eventuais falhas de acordo com os subsídios fornecidos por sua RNA associada. Como proposta de solução para esse problema, o ambiente RENATA foi estendido para suportar redes IP, MPLS e GMPLS. Esta nova versão recebeu o nome de RENATA 2 e será apresentada a seguir no capítulo 5.

Capítulo 5

RENATA 2 – um Ambiente para Gerência Inteligente

É proposta nesta dissertação a arquitetura RENATA 2, que é a evolução natural da RENATA original. O objetivo da nova arquitetura é simplificar e atualizar a arquitetura antiga, de forma a adequá-la às novas tendências da área de redes de computadores. Ferramentas de terceiros foram substituídas por *softwares* mais novos e atuais. Isso trouxe benefícios tanto em termos de melhoria de interface do usuário, como também abriu o leque de suporte a protocolos de redes, tais como, IP, MPLS, GMPLS e GPRS. Entretanto, tecnologias *wireless* não são discutidas nesta dissertação, apesar de serem possíveis de uso. A grande contribuição da arquitetura RENATA 2 é a ferramenta Gerador de Perturbações (GDP). O GDP é um simulador de *links* ópticos que agrega funcionalidades de tratamento de erros ao ns. Outra ferramenta desenvolvida foi o Módulo de Seleção e Preparação de Dados (MSPD). O MSPD é responsável pela comunicação entre o *ns* e JNNS, dois simuladores de código aberto presentes na nova arquitetura.

Este capítulo explica como o ambiente RENATA 2 pode gerar agentes inteligentes baseados em redes neurais aplicados ao gerenciamento de falhas e contém apenas dois tópicos. A arquitetura RENATA 2 é apresentada de forma detalhada na Seção 5.1 e as considerações finais deste capítulo são relatadas na Seção 5.2.

5.1 Arquitetura do RENATA 2

Em relação à arquitetura RENATA 1, houve, na sua versão 2, modificações tanto nas ferramentas de terceiros, como nas ferramentas propostas. Merece destaque o GDP, uma ferramenta desenvolvida nesta dissertação, que tem como função auxiliar o simulador de redes *ns* no processo de simulação de falhas de *links* ópticos. A Figura 5.1 mostra arquitetura do ambiente RENATA 2. O uso do *ns* permitiu que o ambiente RENATA 2 trabalhasse com vários tipos de redes, uma vez que este simula desde redes IP até redes de protocolos sem fio. O MSPD também foi modificado uma vez que deve ser personalizado ao problema específico a ser resolvido. Nesta nova versão, o MSPD também atua na crítica de resultados gerados pelas RNAs. O Módulo de treinamento composto pelo simulador de redes neurais JNNS (*Java Neural Network Simulator*) trouxe uma facilidade maior ao processo de desenvolvimento de redes neurais devido a sua interface amigável. A IT (Interface de Treinamento) e o Módulo Gerador de Agentes tratam da maneira como o agente deve interagir com o protocolo GMPLS para agir em eventuais falhas, sendo ambos ainda uma proposta conceitual e deverão ser implementados em trabalhos futuros.

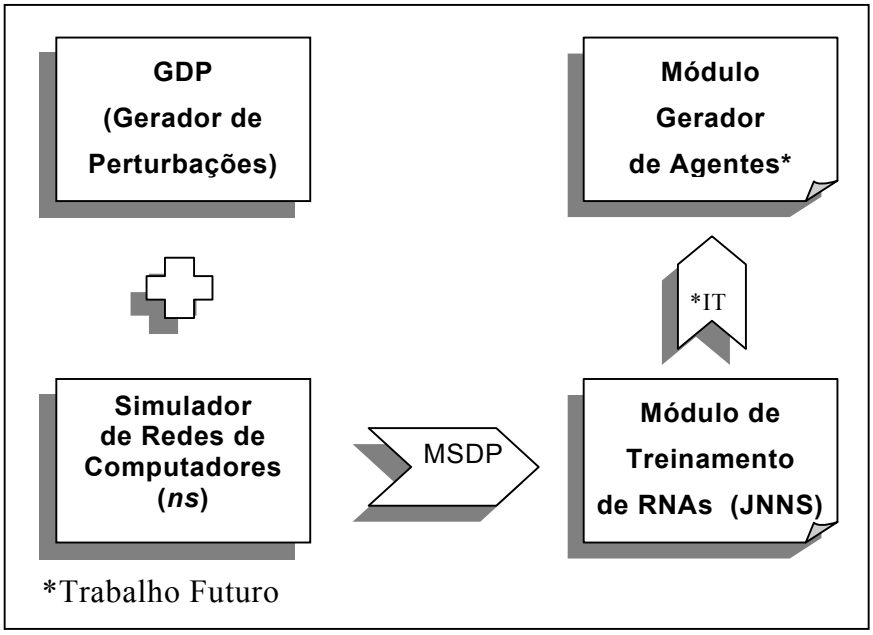

Figura 5.1 – Arquitetura RENATA 2.

5.1.1 *ns - Network Simulator*

O simulador de redes de computadores escolhido para a versão 2 do RENATA foi o *ns* (*Network Simulator)* [31]. Tal escolha foi feita por que o *ns* é uma ferramenta aberta, de amplo uso na comunidade científica e trabalha com diversos tipos de tecnologias de redes. Dentre as simulações que o *ns* trabalha diretamente pode-se relacionar TCP, *mullticast*, roteamento, difusão de dados, redes locais e metropolitanas, redes de longo alcance e redes sem fio. Existem ainda extensões de terceiros, como é o caso do MNS (MPLS *Network Simulator*) que dá ao *ns* a capacidade de trabalhar com redes MPLS. Tal extensão será descrita na Seção 5.1.1.2. O desenvolvimento do *ns* teve início em 1989 com base no *Real Network Simulator* [31]. A versão utilizada no desenvolvimento do trabalho exposto nesta dissertação foi a de número 2.1b8a. Atualmente o desenvolvimento deste simulador é feito através do projeto VINT (*Virtual InterNetwork Testbed*), que tem como objetivo construir um simulador de redes que permita o estudo e a interação de protocolos atuais e futuros. O VINT é um projeto colaborativo entre USC/ISI, Xerox PARC, LBNL e UC Berkeley [33].

O *ns* foi implementado em C++ e usa como *frontend,* uma versão orientada a objetos da linguagem Tcl denominada de Otcl. A linguagem C++, por ser robusta, atinge os requisitos necessários para implementação de protocolos em geral, uma vez que disponibiliza manipulação eficiente de bytes, pacotes e cabeçalhos; implementação de algoritmos que manipulem conjuntos de dados e rapidez de processamento. Para a construção do script da simulação, o uso de Otcl se mostra mais adequado, por ser de mais alto nível que a linguagem C++ e permitir modificações mais facilmente, além de ser mais amigável com o usuário. Realmente, o uso de duas linguagens de programação facilita o desenvolvimento de um *script* de simulação pelo usuário final, entretanto o processo de desenvolvimento de extensões do *ns*, por parte de terceiros, se torna mais complexo.

O ponto marcante para a escolha do *ns* neste trabalho foi o fato de ele é uma ferramenta de domínio público e trabalha com diversos tipos de redes. Isto permite ao RENATA 2 ser acessível a qualquer pesquisador, além de abrir um leque de possibilidades de simulações no que tange às tecnologias de rede. Outra vantagem do

ns é o seu amplo uso na comunidade acadêmica e conseqüentemente uma constância de atualizações e extensões. Os pontos fracos ficam por conta de sua documentação, inexistente em certos pontos, e pelo seu modo de tratamento de erro imperativo, que dificultou bastante este trabalho, sendo necessário o desenvolvimento do GDP para melhorar esta funcionalidade.

5.1.1.1 *Nam - Network Animator*

Para a visualização das simulações o pacote *ns* fornece a ferramenta *nam* (*Network Animator*) [32]. Ela possui uma interface amigável com o usuário, sendo capaz de fornecer visualização dos passos da simulação. São também características do *nam* a animação do fluxo do tráfego e o fornecimento de informações sobre as entidades simuladas (links, pacotes e nós) durante o processo de visualização. A Figura 5.2 mostra a interface do *nam*.

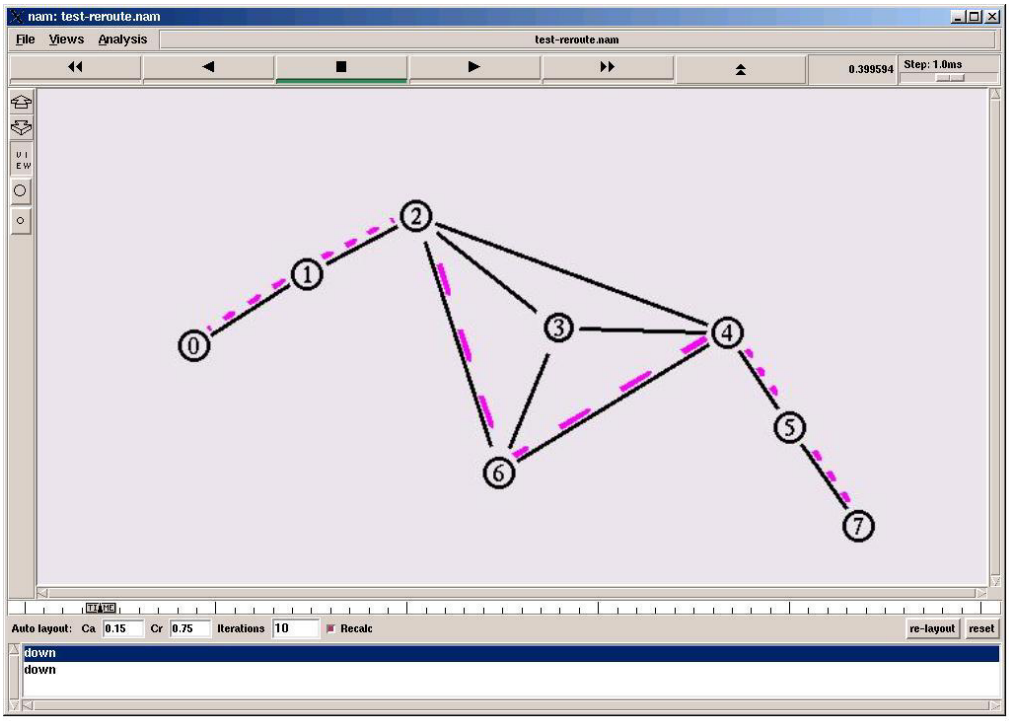

Figura 5.2 – Interface do *Network Animator.*

Para o *Network Animator* ser usado é necessário gerar no *script* (arquivo *.tcl) da simulação um arquivo de *trace.* Este arquivo terá todas as informações da simulação, tais como, topologia, nós, *links*, tráfego dos pacotes e geradores de tráfego.

Após ler o arquivo de *trace* o *nam* está apto a executar a animação de acordo com os comandos fornecidos pelo usuário: velocidade animação, pausa, prosseguir simulação, retroceder quadros e mudar o *layout* das estações.

5.1.1.2 MNS - MPLS *Network Simulator*

O MNS (MPLS *Network Simulator*) é uma extensão do *ns* que permite a simulação de redes MPLS. Apesar de ter sido originalmente desenvolvido para ambientes Unix da Sun, o MNS se comporta bem em ambiente Linux Conectiva 8, sendo este último usado neste trabalho. Essa extensão foi desenvolvida na Coréia pelo Departamento de Engenharia da Computação da Universidade Nacional de Chungnam. [30]. Em princípio, o MNS foi criado para funcionar na versão 2.1b6 do *Network Simulator*, mas pode ser adaptado para versões mais recentes**.** O MNS possui duas versões. A versão 1.0 já vem com *ns* e implementa as seguintes funções do MPLS:

- ✓ Comutação de rótulos - operações de troca e empilhamento de rótulos, tratamento do campo TTL e *penultimate hop popping*;
- ✓ Protocolo LDP - tratamento de mensagens LDP, tais como, mapeamento, remoção, liberação e notificação;
- ✓ Protocolo CR-LDP - tratamento de mensagens de requisição e mapeamento;
- ✓ Agregação de fluxo de pacotes;
- ✓ Controle e configuração de LSPs - distribuição e alocação de rótulos e configuração de LSPs pré-definidos pelo usuário (ER-LSP);

A segunda versão agrega as seguintes melhorias:

- ✓ Implementação da *Constraint-based Routing* que permite estabelecer CR-LSPs usando informações sobre reserva de recursos da rede;
- ✓ Re-roteamento (Restauração de caminhos) com suporte a esquemas de restauração de rota pré-negociada.

- ✓ Melhorias proporcionadas pelo CR-LDP que permitem o estabelecimento de CR-LSPs baseados em parâmetros, tais como, taxa de transmissão e tamanho do buffer, foi também adicionado o mecanismo de preempção de recursos.

O MPLS *Network Simulator* ainda não atingiu sua versão final e está pouco documentado. Apesar de possuir algumas falhas, como seus próprios desenvolvedores reconhecem [30], as funcionalidades requeridas nas simulações descritas nesta dissertação funcionaram em plenitude e não causaram problema algum.

5.1.2 GDP - Gerador de Perturbações

Apesar do *ns* oferecer facilidades ao trabalho proposto nesta dissertação, o mesmo não oferece um tratamento de erro apropriado à resolução do problema de se gerar RNAs aplicadas a agentes inteligentes para gerenciamento de falhas. Logo, foi necessário o desenvolvimento de uma ferramenta que suprisse esta funcionalidade, tal software foi denominado de GDP (Gerador de Perturbações).

O tratamento de erro proposto pelo *ns* é imperativo. Por exemplo, para simular a queda de um *link* entre os LSRs 4 e 6 no instante 4,5s da simulação de uma rede MPLS, deve-se executar o seguinte comando: ***$ns rtmodel-at 4.5 down $LSR6 $LSR4*.** Neste exemplo, nenhum fator relacionado com a simulação contribui para a queda do *link*, o comando em questão reflete apenas a vontade do usuário, ou seja, de forma imperativa tal comando é passado ao *ns*, que por sua vez o executa sem nenhuma restrição e sem nenhum relacionamento com o mundo real. Para se treinar uma rede neural é necessária uma base de conhecimento que se aproxime do mundo real ou o reflita fielmente. Logo, o tratamento de erros proposto pelo *ns* não é adequado a geração de um banco de exemplos necessário ao aprendizado das redes neurais propostas neste trabalho.

O GPD simula o comportamento dos principais dispositivos de uma rede de fibra óptica, levando em consideração as variáveis físicas que ditam o comportamento de tais dispositivos. Os equipamentos simulados pelo Gerador de Perturbações são: transmissor óptico, fibra óptica e receptor óptico. Estes são ditos principais por estarem obrigatoriamente presentes em redes de fibra óptica sejam diretamente

(próprio equipamento), ou indiretamente, por exemplo, um transmissor que faz parte de um multiplexador DWDM. A Figura 5.3 mostra um *link* óptico e demonstra os principais equipamentos de um rede de fibra óptica. O gerador de perturbações foi desenvolvido em código Java puro o que lhe permite ser multiplataforma, sendo baseado na teoria física exposta no Capítulo 3, logo deve-se recorrer a este capítulo em caso de dúvida.

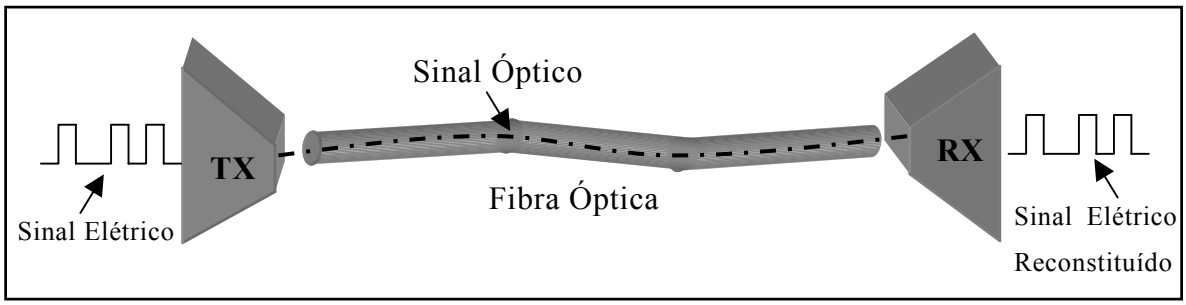

Figura 5.3 – Link Óptico.

5.1.2.1 Transmissor Óptico

Há dois tipos de transmissores ópticos usados em redes de fibra óptica, os baseados em LEDs e os baseados em lasers. O gerador de perturbações abstrai apenas os transmissores do segundo tipo, por serem mais precisos e terem maior capacidade de transmissão em relação aos primeiros. Estes são também mais adequados às transmissões DWDM, que são o alvo das simulações do Capítulo 6. Os parâmetros do GDP relacionados com o transmissor são os seguintes: temperatura ambiente, corrente de operação e comprimento de onda de operação. Não há restrição no domínio destas variáveis, exceto temperatura ambiente. O parâmetro relacionado com a temperatura ambiente pode assumir valores do intervalo fechado [0,60] graus Celsius, entretanto quando o comprimento de onda de operação do transmissor assume o valor de 1300 nanômetros, o intervalo de aceitação passa a ser de [0,130] graus Celsius. Isto ocorre devido à dificuldade de obtenção das curvas L-I, que são obtidas empiricamente, e mostram o comportamento de um transmissor específico em temperaturas elevadas. Tal fato não altera a funcionalidade do transmissor, uma vez que a maioria dos

transmissores degrada o sinal emitido por eles em ambientes de temperatura elevada[15] ou a corrente de operação não é suficiente para atingir a corrente limiar[16], também em virtude da temperatura. Se estas duas possibilidades ocorrerem um receptor não poderá decodificar o sinal emitido pelo transmissor. A Figura 5.4 mostra a janela de entrada de dados do transmissor do gerador de perturbações com as respectivas unidades, há janelas semelhantes deste *software* para o receptor e a fibra óptica.

Figura 5.4 – Janela de entrada de dados do transmissor.

5.1.2.2 Fibra Óptica

Quanto à fibra óptica, o gerador de perturbações captura apenas a atenuação causada pela mesma em um sinal óptico. Para isso, é necessária a entrada de valores relacionados com o coeficiente de atenuação da fibra dado em dB/Km (decibéis por quilometro) e o comprimento da fibra dado em Km. Não há restrição de valores para estas variáveis, entretanto é de responsabilidade do usuário fornecer valores que reflitam a realidade.

---

[15] Temperatura elevada, neste caso, se refere a valores superiores a 60º C, como mostra a Figura 3.2

[16] Corrente limiar é definida pela Equação 3.1.

5.1.2.3 Receptor Óptico

Para que o receptor possa interpretar os sinais recebidos pelo transmissor, deve-se conhecer as potências representativas dos bits em 1 e em 0 (zero), logo estes valores devem ser passados ao gerador de perturbações em miliwatts. Ainda é necessário o fornecimento da taxa de recepção dos bits (em Gbps), de forma que se possa calibrar a freqüência de recebimento do sinal luminoso. Quanto ao comprimento de onda de operação, o receptor simulado por esta ferramenta está restrito aos que operam a 1550 nanometros. Tal fato ocorre devido à dificuldade de se obter a energia do fóton em ondas de comprimento variáveis e que trafegam no material do transmissor. Entretanto, para o problema modelado neste trabalho esta restrição não o afeta, pois a grande maioria dos receptores ópticos DWDM operam nesta faixa de comprimento de onda.

5.1.2.4 Tipos de Simulação

As variáveis que sofrem ação do gerador de perturbações são a temperatura do laser e a corrente de operação do mesmo, sendo "perturbadas" isoladamente ou em conjunto e de três maneiras diferentes: Linear, Aleatória Linear e Aleatória, o que totaliza nove simulações distintas. Para qualquer tipo de simulação é necessário fornecer ao gerador de perturbações o nome para identificar a simulação, o tempo total da simulação, o intervalo de perturbação e o tempo de amostragem (de quanto em quanto tempo o GDP deve capturar o estado do *link* simulado).

A Figura 5.5 apresenta um gráfico gerado a partir dos dados de uma simulação linear. Este tipo de simulação requer que o usuário indique o valor inicial e final que a variável perturbada deve assumir. A Figura 5.6 mostra um gráfico de uma simulação aleatória linear e a Figura 5.7, um gráfico de uma simulação aleatória. Ambas as simulações, aleatória e aleatória linear, requerem um parâmetro chamado de variação máxima, que dita qual o valor máximo que uma variável pode assumir em um tempo de amostragem. Isso impede que um parâmetro simulado assuma uma variação de magnitude exagerada em um curto intervalo de tempo. Por exemplo, não é razoável que para uma massa considerável, como é caso de um transmissor laser, possa

acontecer uma variação de 60^{o} C em um intervalo de um centésimo de segundo. Isso até pode ocorrer para massas muito diminutas como é o caso de uma microgota de tinta aquecida em uma impressora jato de tinta. O gráfico da Figura 5.5 e o gráfico da Figura 5.6 mostram simulações relacionadas com valores lineares crescentes. Se o usuário do GDP desejar que as variáveis envolvidas em tais simulações assumam valores decrescentes, basta fornecer um valor negativo para a variação máxima no caso da simulação aleatória linear e um valor final menor que o valor inicial, para a variável a ser perturbada, no caso da variação linear.

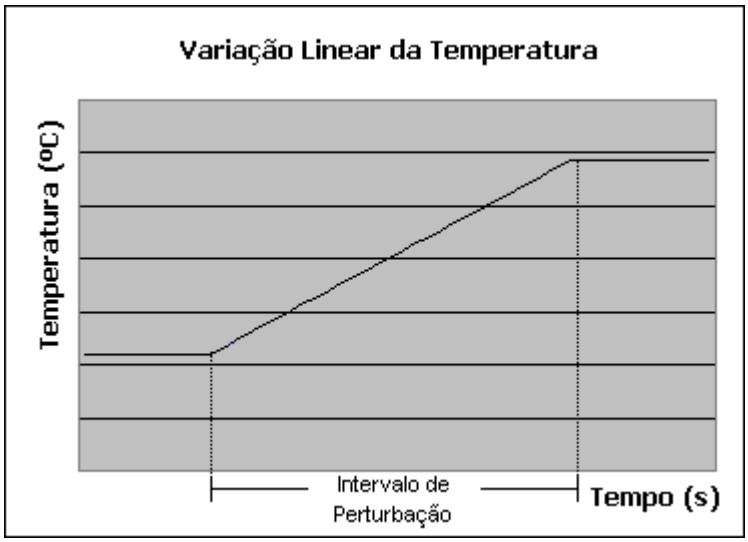

Figura 5.5 – Simulação Linear.

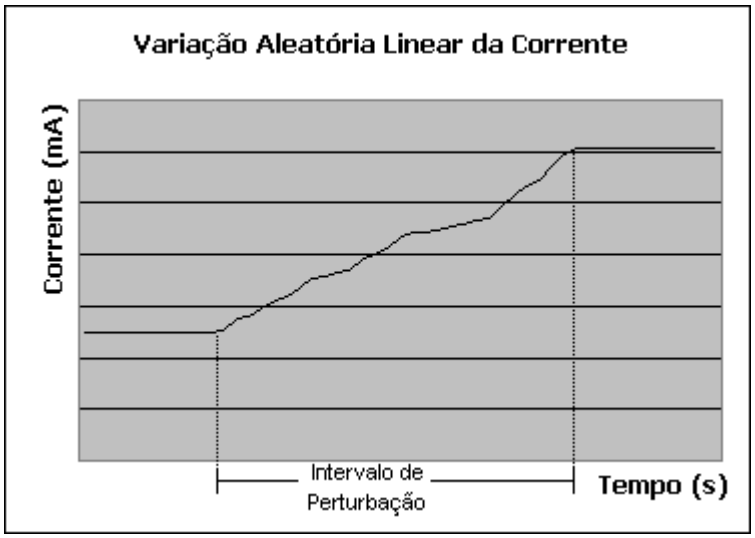

Figura 5.6 – Simulação Aleatória Linear.

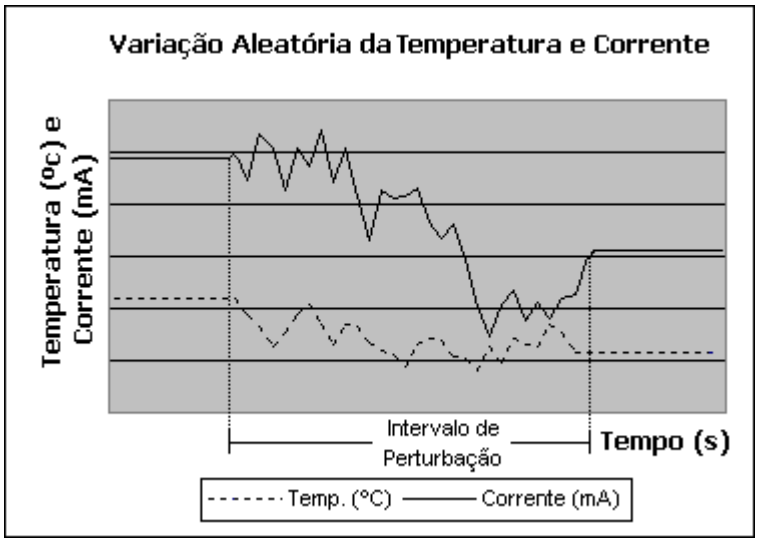

Figura 5.7 – Simulação Aleatória.

5.1.2.5 Saída da Simulação

Como saída da simulação, o GDP gera a parte responsável pelo tratamento de erro do *script* TCL do *ns*. Para que este código seja inserido em uma simulação do *ns* o usuário deve colocar no *script* TCL (simulação do *ns*) a *tag* #GDP e fornecer o local de armazenamento desse *script* ao gerador de perturbações, que irá colocar o código referente ao tratamento de erro da camada física logo abaixo do *tag*. Um exemplo de código gerado é mostrado na Figura 5.8. Por exemplo, o comando ***$ns at 3,6000 "$em set rate_ 0,932528525"*** indica que no instante 3,6s da simulação deve ser gerada uma taxa de perdas de pacotes de 93%.

Além da saída para o *ns* e das variáveis simuladas (tempo e corrente) o gerador de perturbações fornece a potência de saída do laser transmissor, a potência de recebida pelo receptor, a BER da camada física relativa ao link simulado e informações de documentação da simulação, conforme exemplificado na Figura 5.9, a qual mostra a janela principal do GDP.

5.1.3 MSPD – Módulo de Seleção e Preparação de Dados

O MSPD tem como principal função preparar os *logs* do *ns* com o intuito de colocá-los no formato aceito pelo simulador de redes neurais. Além desta missão principal, o MSPD atua na escolha e ordenação dos *logs* de forma a não fornecer os

padrões em uma seqüência que “vicie” o treinamento das redes neurais, como também faz comparações entre os arquivos de padrão de treinamento com os resultados gerados pelas redes neurais, a fim de calcular o erro gerado em ocasiões especiais da simulação.

```
$ns at 0,0000 "$em set rate_ 0,000000000"
$ns at 2,3000 "$em set rate_ 0,000000497"
$ns at 2,4000 "$em set rate_ 0,000001318"
$ns at 2,5000 "$em set rate_ 0,000003571"
$ns at 2,6000 "$em set rate_ 0,000009883"
$ns at 2,7000 "$em set rate_ 0,000027955"
$ns at 2,8000 "$em set rate_ 0,000080845"
$ns at 2,9000 "$em set rate_ 0,000239159"
$ns at 3,0000 "$em set rate_ 0,000724056"
$ns at 3,1000 "$em set rate_ 0,002244509"
$ns at 3,2000 "$em set rate_ 0,007127776"
$ns at 3,3000 "$em set rate_ 0,023200298"
$ns at 3,4000 "$em set rate_ 0,077440541"
$ns at 3,5000 "$em set rate_ 0,265223378"
$ns at 3,6000 "$em set rate_ 0,932528525"
$ns rtmodel-at  3,7000 down $LSR6 $LSR4
```

Figura 5.8 – Saída típica do Gerador de Perturbações.

Como cada problema a ser solucionado pelo RENATA requer diferentes tipos de parâmetros e, conseqüentemente, *logs* distintos, o MSPD foi personalizado (codificado) de forma a atender as especificidades do problema tratado neste trabalho.

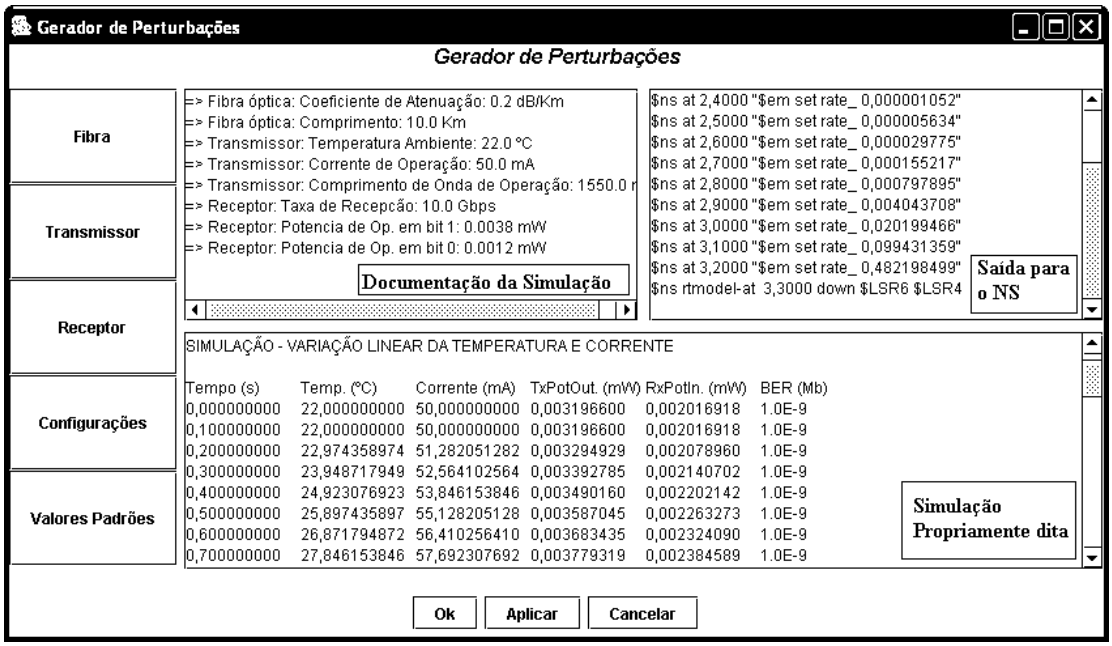

Figura 5.9 – Janela principal do Gerador de Perturbações.

5.1.4 JNNS – *Java Neural Network Simulator*

Para propósitos de concepção e treinamento das redes neurais que serviram de máquina inteligente dos agentes RENATA 2 foi usada a ferramenta JNNS (*Java Neural Network SimulatorI*). O uso desta ferramenta é justificado pelo alto custo e complexidade do desenvolvimento de um *software* próprio para manipulação de redes neurais, bem como pela facilidade de obtenção de ferramentas abertas deste tipo. O JNNS foi desenvolvido no WSI (Instituto de Ciência da Computação *Wilhelm-Schickard*) em Tübingen, Alemanha, sendo baseado no núcleo do SNNS (*Stuttgart Neural Network Simulator*). A grande inovação do JNNS em relação ao SNNS é a nova interface gráfica baseada em tecnologia Java, o que lhe conferiu maior portabilidade e facilidade de uso. Ambas ferramentas se equivalem quanto à usabilidade principal (desenvolvimento de redes neurais de vários tipos) e diferem em pontos específicos como exibição tridimensional das redes e geração automática de código C, tais pontos estão presentes apenas na versão SNNS.

5.2 Conclusões

O módulo para gerar os agentes inteligentes está fora do escopo deste trabalho, pois esta dissertação se concentra no processo de desenvolvimento da máquina inteligente dos agentes RENATA 2. Ficando a interface de treinamento e o módulo gerador de agentes como sugestão de trabalhos futuros. Outro trabalho futuro seria a construção de uma ferramenta que substituísse o simulador de redes neurais de uso geral (JNNS), por uma ferramenta mais especifica voltada para redes neurais MLP. Com o uso de tal ferramenta os usuários do RENATA 2 (gerentes e/ou administradores de redes) poderiam se abstrair do uso de redes neurais, obtendo mais tempo para se concentrar na modelagem do problema (simulações) e não na forma como resolve-lo. Isto facilitaria sobremaneira o trabalho desses usuários.

O ambiente RENATA 2 pode atuar em diversos problemas das redes IP, MPLS e GMPLS, mostrando-se mais geral e com melhor interface que o RENATA original. Este bom desempenho se deve ao uso de simuladores mais recentes: *ns* e JNNS. O *ns* se apresentou como um simulador de redes que se aplica a diversos domínios e de

amplo uso na comunidade científica, enquanto o JNNS se mostrou de mais fácil uso que o seu precursor, o SNNS. As novas ferramentas desenvolvidas nesta dissertação, em destaque o GDP, são importantes tanto para integrar, como para completar as funcionalidades dos simuladores supracitados.

Uma extensão convencional do *ns* necessita de um estudo profundo de sua hierarquia de classes, além do entendimento de um código pouco documentado e complexo que combina duas linguagens de programação o C++ e o Otcl [34]. O GDP é uma proposta inovadora e simples para a extensão do *ns* e de outras ferramentas semelhantes, pois pode ser desenvolvido em qualquer linguagem de programação, dispensa o entendimento do código do *ns* por parte do desenvolvedor e tem um custo de desenvolvimento reduzido em termos financeiro e de tempo. Aplicações semelhantes podem capturar o comportamento de equipamentos como roteadores, switches e multiplexadores.

O ambiente RENATA 2 mostrou-se capaz de resolver o problema do re-roteamento rápido em redes IP+GMPLS sobre DWDM, como verificado no próximo capítulo, onde os resultados desta dissertação serão expostos.

Capítulo 6

Utilizando o RENATA 2 para Predição de Falhas de *Links* Ópticos

A forma lenta de re-roteamento provida por redes GMPLS sobre DWDM é um dos problemas que limitam o uso deste protocolo neste tipo de enlace. Uma solução para tal problema é o uso de mecanismos de proteção 1+1, em que um *link* é utilizado como espelho para prover recuperação imediata de falhas. Tal proposta é bastante onerosa, pois um canal deve ser alocado de forma exclusiva para fornecer a segurança necessária. Os agentes RENATA 2 podem predizer situações problemáticas em um *link*, o que permite a tomada de atitudes pró-ativas antes que o problema se agrave. Isto poderá possibilitar que um mecanismo de restauração 1:n possa se aproximar do modelo 1+1, uma vez que, levantada a possibilidade de ocorrência de falha, o canal de *backup* passaria a funcionar de imediato como espelho do *link* supostamente problemático.

Este trabalho proporcionou o desenvolvimento da máquina inteligente de um agente de software que monitorará um *link* óptico de uma rede em particular. A forma de sinalização desse agente poderá ser simples, indicando apenas um instante futuro com seu respectivo percentual probabilístico de falha. Ela também pode ser mais robusta para interagir com o protocolo GMPLS com o intuito de melhorar o processo de re-roteamento rápido. Esta comunicação do agente com o protocolo GMPLS e/ou

com o gerente da rede não foi contemplada neste trabalho e deverá ser concretizada em trabalhos futuros.

A Seção 6.1 apresenta uma descrição do sistema óptico simulado. Tal sistema foi baseado em especificações reais de equipamentos DWDM de diversos fabricantes [35] [36] [37]. Tais especificações serviram de entrada para o GDP que simulou um *link* óptico (camada física). Em seguida, o *ns* foi usado para simular a camada de enlace e de rede. A Seção 6.2 apresenta explicações sobre as RNAs desenvolvidas neste trabalho, as quais podem ser usadas como máquina inteligente para agentes de *software*. Os resultados obtidos pelas RNAs propostas, com os devidos comentários, estão na Seção 6.3. Na Seção 6.4 está presente a conclusão deste capítulo.

6.1 O Sistema Simulado

O processo de simulação é necessário para definição do banco de exemplos que servirá de base de conhecimento para o aprendizado das redes neurais. Para isso foram necessárias simulações das camadas física, de enlace e de rede. Neste processo, o GDP simula um *link* da camada física (transmissor, fibra óptica e receptor) e fornece o comportamento desta camada para o *ns*, responsável pelas simulações da camada de enlace e rede.

6.1.1 A Camada Física

A responsabilidade pela simulação da camada física é do software Gerador de Perturbações o qual simula um *link* óptico. O modo de operação deste *link* é *full-duplex*. Em sua entrada há um transmissor laser semicondutor e em sua saída há um fotoreceptor também semicondutor. Este enlace não contempla nem amplificadores nem regeneradores de sinal. Os parâmetros de configuração destes equipamentos foram escolhidos segundo um sistema DWDM típico considerado pela empresa Altamar [35]. A Tabela 6.1 traz as especificações de componentes ópticos segundo o modelo proposto por tal empresa.

COMPONENTE ÓPTICO	ESPECIFICAÇÕES	
Fibra óptica	Coeficiente de Atenuação	Comprimento
	0,25 db/km	44km
Transmissor	Potência de Saída (max.)	Perda de Inserção
	-3dBm	6dBm
Receptor	Potência de Entrada (min.)	Perda de Inserção
	-26dBm	6dBm

Tabela 6.1 - Especificações do sistema Mux/Demux DWDM Altamar.

Quanto à vazão do canal óptico, foi assumido o valor de 10 Gbps e o valor 1550 nm para o comprimento de onda de operação do transmissor e do receptor. Tais valores são baseados no Sistema DWDM Alcatel 1640 OADM [37], também estando em conformidade com o sistema Multiplexador/Demultiplexador DWDM da Scientific Atlanta [36]. Houve a necessidade de juntar especificações de sistemas DWDM de diferentes fabricantes, não porque tais especificações fossem divergentes, mas devido a documentações incompletas. Por exemplo, no modelo Altamar apresentado na Tabela 6.1 não foi mencionado o comprimento de onda de operação dos equipamentos transmissores e receptores, já as especificações do Sistema DWDM Alcatel 1640 OADM relacionam o comprimento de onda de operação e a taxa de transmissão, mas não apresentam nenhum dado sobre a sensibilidade do fotodetector. Apesar do sistema óptico simulado ser hipotético, ele se aproxima bastante de um sistema DWDM real.

6.1.2 As Camadas de Enlace e de Rede

A Figura 6.1 mostra uma janela do *Network Animator* com a topologia da rede simulada no *Network Simulator*. O *ns* foi a ferramenta usada para simular as camadas

de enlace e rede. Os protocolos usados na rede simulada foram o IP, protocolo da camada de rede, e o MPLS, cujo cabeçalho MPLS é colocado entre a camada de enlace e a camada de rede. Não foi usado nenhum pacote ou extensão do *ns* que possibilitasse a simulação de uma rede GMPLS, e sim o MNS[17] que agrega funções do MPLS ao *ns*. Tal fato não compromete a integridade da simulação, pois o *link* óptico foi simulado no GPD e o GMPLS é apenas a forma generalizada do MPLS para ser executada em diversos tipos de redes.

De acordo com [40], *"a premissa utilizada pelo GMPLS é que um rótulo pode ser generalizado para qualquer coisa que pode identificar um fluxo de dados"*. Por exemplo, em uma fibra óptica, cuja banda é dividida em comprimentos de onda (*lambdas*), um *lambda* poderia ser alocado para servir implicitamente de rótulo usado no transporte de dados.

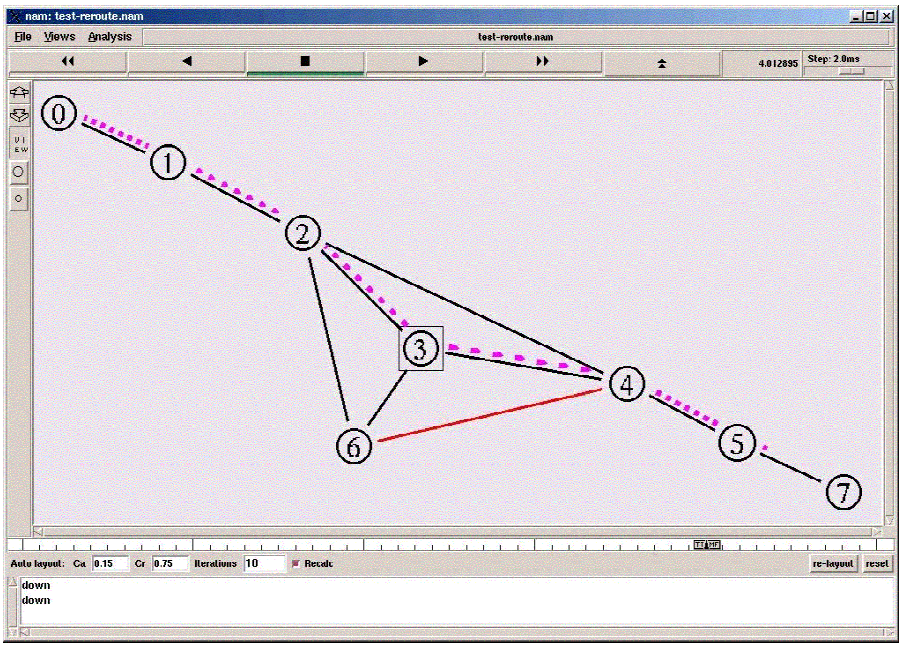

Figura 6.1 – Topologia da rede simulada.

[17] O uso do MNS foi necessário, uma vez que até o presente momento não há um pacote do ns para moldá-lo ao GMPLS.

Na rede simulada, os nós 0 e 7 são roteadores IP, enquanto os demais são LSRs, sendo os nós 1 e 5 *Edge* LSRs (roteadores de borda). Foram estabelecidos dois LSP: o LSP de identificador 1000 (LSP 1000) com caminho 1 – 2 – 6 – 4 – 5 e o LSP de identificador 2000 (LSP 2000) com caminho 1 – 2 – 3 – 4 – 5. O LSP 1000 é o principal, ou seja, por onde o tráfego deve fluir normalmente e o LSP 2000 é a rota alternativa e pré-negociada a ser usada em caso de falha do LPS 1000. O LSR 6 é o protetor do tráfego (definição usada no MNS), cuja função é indicar se há necessidade de re-roteamento ocasionado pela falha do *link* 6 – 4. O link modelado na Seção 6.1.1 se refere ao enlace que liga os LSRs 4 e 6, isso significa que o Gerador de Perturbações se comunica diretamente com o *Network Simulator* para especificar o comportamento desse enlace. Para facilitar o entendimento chamar-se-á, a partir desse ponto, o enlace que liga os LSRs 4 e 6 de ***link* principal**.

Para que a simulação das camadas 2 (enlace) e 3 (rede) fiquem de acordo com a camada 1 (física), a banda passante suportada pelo *link* principal é de 10 Gbps[18] e os demais *links* suportam até 12 Gbps. Tal disposição de banda passante foi necessária para permitir que o link principal fosse congestionado em certas situações e, desta forma, gerar uma melhor base de conhecimento, evitando que as redes neurais pudessem confundir congestionamento com falhas. Houve necessidade de se trabalhar em escala para dimensionar a banda passante e o tráfego simulado, pois o simulador *ns* não obteve um desempenho satisfatório (mostrou-se muito lento, cerca de 1,5 minuto para cada simulação) quando operou com tráfego dimensionado em gigabits por segundo. Para solucionar o problema foi necessária a seguinte convenção: cada bit transmitido na simulação equivale a 10000 bytes transmitidos no mundo real. Então, o *link* principal (enlace 6 – 4, também simulado na camada física) foi configurado na simulação do *ns* com o valor 1 Mbps para banda passante, o que deve equivaler a 10 Gbps no mundo real. Como supracitado, os demais *links* foram configurados propositalmente com o valor 1,2 Mbps, com o intuito de, em certas ocasiões, sobrecarregar o *link* principal com congestionamento e não com falhas.

---

[18] A melhor forma de quantificar banda passante é usar hertz e não bits por segundo. Entretanto, a segunda forma é mais comumente usada.

O nó 0 é um roteador IP e também assume a função de gerador de tráfego, enquanto o nó 7 , que também utiliza o protocolo IP, é o destino final dos dados gerados pelo nó 0. O tráfego gerado na simulação é do tipo CBR UDP com pacote de 53 bits e rajada de 5 Kb a uma taxa de transmissão que assume os seguintes valores: 0.0, 0.2, 0.4, 0.6, 0.8 1.0 e 1.2 Mbps dependendo da simulação. Para os demais valores de vazão de tráfego espera-se que as redes neurais possa generalizar estes valores a partir dos valores simulados. O significado no mundo real dos parâmetros de tráfego supracitados são aproximadamente: 64KB para o tamanho de pacote, rajada de 6MB e vazão que varia de 0 a 12 Gbps.

6.2 A Máquina Inteligente dos Agentes

O Módulo de Seleção e Preparação de Dados (MSPD) foi responsável pelo ajuste dos arquivos de *log* gerados nas simulações do *ns* de forma que estes pudessem compor o banco de conhecimento para treinamento e validação da rede neural que servirá de máquina inteligente para os futuros agentes. O processo de treinamento e validação da rede neural em questão foi desenvolvido no software de simulação de redes neurais JNNS (*Java Neural Network Simulator*).

Para prever a falha do *link*, a rede neural deve consultar um histórico relativo aos 20 (vinte) instantes anteriores ao momento presente e sugerir um valor para a BER (imediatamente futura) da camada física. Basicamente, cada simulação do *ns* gera um *log* que contém valores para o seguinte quinteto ordenado: *status* do *link*, quantidade de banda passante ocupada, pacotes recebidos, pacotes perdidos e BER (*Bit Error Rate)* da camada física. O parâmetro *status* do *link* indica se o enlace está em funcionamento ou não e sua necessidade é diferenciar o silêncio de uma falha. Os demais parâmetros já possuem mnemônicos bem descritivos e dispensam comentários.

A cada décimo de segundo foi feita uma coleta de dados das simulações (quinteto ordenado). Desta forma, para a predição de uma falha, as RNAs devem receber as variáveis de entrada e sugerir ou não a necessidade de re-roteamento no próximo décimo de segundo. Caso se verifique a possibilidade de uma falha, a rede de computadores terá um décimo de segundo para realizar o re-roteamento. Simulações

feitas no ns constataram que para a topologia de rede óptica proposta neste trabalho o tempo médio necessário para o re-roteamento é de aproximadamente 0.025671s. Considerando no máximo 20.000 instruções de máquina para efetuar todos os cálculos de uma RNA 20-20-1, em um processador de 1GHz, seriam necessários 0.00002s. Desta forma, um décimo de segundo é mais que suficiente para que a rede de computadores possa realizar o re-roteamento sugerido pelas RNAs antes de se verificar perdas de dados. Foram desenvolvidos três tipos de redes neurais MLP:

- ✓ **Tipo 1** usa variáveis de entrada relacionadas com os equipamentos físicos e se relaciona com a predição de falhas da camada física;
- ✓ **Tipo 2** usa variáveis relacionadas com o tráfego do link óptico (exceto o *status* do *link*) e trabalham no âmbito da inferência de falhas no tempo presente;
- ✓ **Tipo 3** usa as mesmas variáveis das redes neurais Tipo 2 e tentam obter a saída das redes neurais Tipo 1

Estes três tipos de redes neurais são candidatas a máquina inteligente dos agentes inteligentes futuros.

6.3 Resultados

O processo de busca pela melhor topologia de uma rede neural é um processo empírico e complexo. No contexto deste trabalho, foram feitos vários testes (tentativa e erro) com diferentes topologias, parâmetros de treinamento, variáveis de entrada e janelas de tempo das simulações para a escolha dos melhores valores para estas entidades. Julga-se ser desnecessário o comentário de todos os testes feitos, assim falar-se-á apenas sobre os resultados mais importantes, sem deixar de mencionar certos insucessos que, uma vez não repetidos, possam vir a facilitar futuras evoluções do RENATA 2. Por coincidência, a topologia 20-20-1 com algoritmo de treinamento *backpropagation,* foi a mais adequada para os três tipos de redes neurais citados anteriormente, bem como os parâmetros de treinamento citados na Tabela 6.2.

A taxa de aprendizagem (η) atua no algoritmo *backpropagation* e fornece uma "aproximação" para a trajetória no espaço dos pesos calculada pelo método de descida

mais íngreme. Quanto menor for o parâmetro da taxa de aprendizagem η, menores serão as variações dos pesos sinápticos da rede, de uma iteração para outra, e mais suave será a trajetória no espaço de pesos. Esta melhoria é obtida à custa de uma taxa de aprendizado muito lenta. Por outro lado, se fizermos o parâmetro da taxa de aprendizagem η muito grande, para acelerar a taxa de aprendizagem, as grandes modificações nos pesos sinápticos resultantes podem tornar a rede instável (oscilatória).

O máximo erro tolerado ou delta **(d_{max})** Corresponde à máxima diferença tolerada $d_j = t_j - o_j$ entre o valor de aprendizagem e uma saída o_j de uma unidade de saída. Por exemplo, se valores acima de 0.9 devem ser considerados como 1, e valores abaixo de 0.1 devem ser considerados como 0, então d_{max} deve ser configurado como 0.1. Isso previne o *overtraining* da rede. Valores usuais para o d_{max} são 0.1 ou 0.2.

Parâmetro	Valor
η (taxa de aprendizagem)	0.2
d_{max} (delta)	0.1
Ciclos	1000
Passos	1

Tabela 6.2 - Parâmetros de treinamento das redes MPL escolhidas.

O banco de conhecimento usado no treinamento de cada tipo de rede neural foi composto de três conjuntos de **padrões de treinamento** (termo usado no JNNS) com cada conjunto armazenado em um arquivo que possui uma função bem definida:

- ✓ `treina.pat`, responsável pelo treinamento propriamente dito e possui um maior número de padrões: 29.760 (vinte e nove mil setecentos e sessenta), ou seja, 78% dos padrões do banco de conhecimento
- ✓ `valida.pat`, também usado no processo de treinamento e serve para validar os resultados do arquivo `treina.pat, possuindo` 4.216

(quatro mil duzentos e dezesseis) padrões, 11% do total de padrões do banco de conhecimento.

- ✓ `testa.pat`, este não foi apresentado à rede neural durante o processo de treinamento. Se a rede neural aprendeu corretamente e se suas generalizações forem satisfatórias, os resultados apresentados pela mesma em função do uso deste arquivo devem estar condizentes com o esperado. Quanto à quantidade de padrões, `testa.pat` e `valida.pat` são idênticos.

A divisão do banco de conhecimento sugerida por [16] deveria ser feita na forma 80%, 10%, 10%, respectivamente, para cada arquivo (*.pat) citado. Entretanto, uma simulação envolve um certo número de padrões e estes devem ser mantidos em seqüência de modo a não alterar o caráter temporal da simulação. Logo, a divisão do banco de conhecimento teve que seguir também este critério.

O MSE (*Mean Square Error*) ou Erro Médio Quadrático é a forma mais comum de verificar se a rede MLP estão se comportando de forma adequada, este é definido pela fórmula 7.1 mostrada logo abaixo.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(S_{des} - S_{obt})^2 \quad , \tag{7.1}$$

na qual S_{des} e S_{obt} representam respectivamente as saídas desejadas e obtidas da rede neural e *n,* o número total de padrões. O JNNS já fornece o MSE para os arquivos `treina.pat` e `valida.pat`, bem como o gráfico de andamento do processo de treinamento. Os valores para a variável S_{des} podem ser obtidos dos próprios arquivos de padrões (*.pat), enquanto os valores para a S_{obt} são fornecidos pelo JNNS em forma de arquivos de resultado ou saída gerada pela rede (*.res).

Apesar do MSE ser usado pelo JNNS, este relata o erro da RNA levando em consideração todos os padrões de treinamento. Tal fato pode mascarar o erro dos pontos considerados críticos para este experimento. Por exemplo, as saídas dos padrões onde ocorrem transições de um estado normal para um estado de falha do

enlace simulado podem não ser fielmente descritas pelo MSE por se tratar de um conjunto reduzido[19] de casos imersos em um universo bem maior.

Para resolver este problema foi implementado no MSPD uma forma de calcular o MSE nos Pontos de Picos Suaves[20], comparando cada um dos arquivos `.pat` com seu respectivo `.res`. Para um melhor entendimento da definição de Pontos de Picos deve-se observar a Figura 6.2.

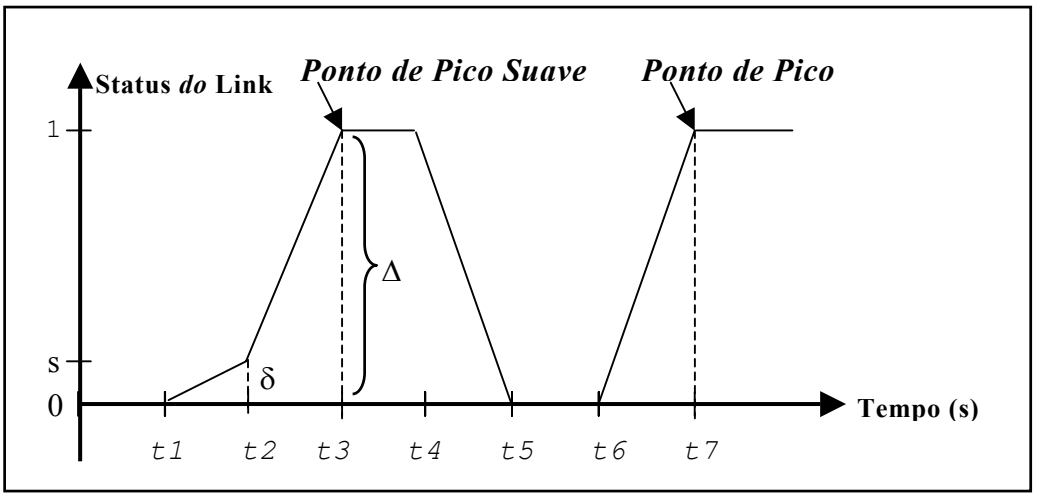

Figura 6.2 – Exemplo de pontos de picos.

Considere *TA* na Figura 6.2 como sendo o triângulo definido pelos pontos *(t1,0) (t2,0) (t2,s)* e *TB* o triângulo *(t1,0) (t3,0) (t3,1)*, ambos sem perda de generalidade. Defini-se δ como sendo o comprimento do cateto de *TA* perpendicular ao eixo Tempo e Δ como sendo o comprimento do cateto de *TB* também perpendicular ao eixo Tempo. Um ponto de pico é caracterizado quando $\Delta \geq 0{,}8$, este será um Ponto Pico Suave (PPS) se $\delta \geq 0{,}001$ e um Ponto Pico Brusco (PPB) se $\delta < 0{,}001$. Durante

[19] Para um melhor entendimento da palavra reduzido pode-se comparar o numero de padrões do arquivo `valida.pat` (4216) com seus respectivos pontos de picos descritos na Tabela 6.3, na Tabela 6.4 e na Tabela 6.5 , os quais são bastantes reduzidos em relação ao arquivo original.

[20] Ponto ou instante da simulação final em que ocorre uma transição "gradual" do *status* normal para o *status* de falha em relação ao *link* simulado.

os experimentos feitos neste trabalho verificou-se que quanto menor o δ, maior o MSE de predição (rede neural tipo 1) e também de inferência (rede neural tipo 2). Os pontos de pico brusco são de difícil predição, pois a mesma se baseia em fatos históricos, os quais muitas vezes são idênticos em casos bruscos.

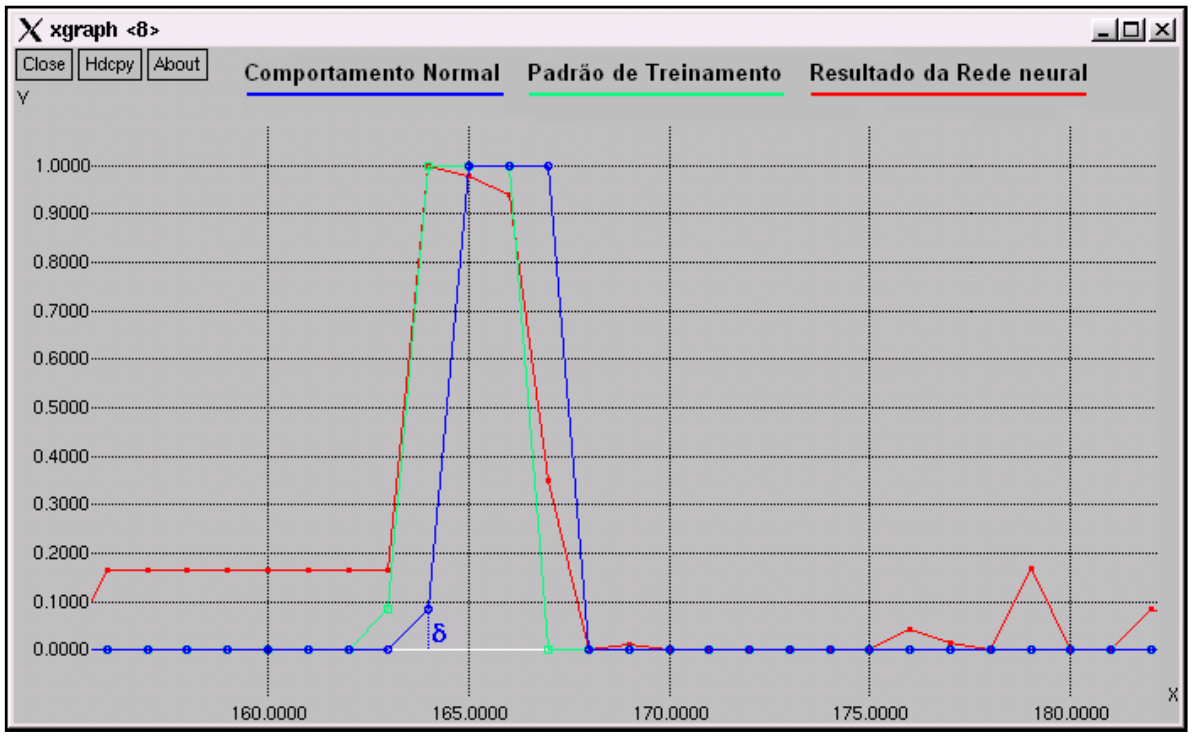

Figura 6.3 – Exemplo de acerto na predição de falhas em pontos de picos.

A Figura 6.3 mostra um PPS na comparação dos arquivos `.res` e `.pat` do banco de conhecimento `testa.pat`. Nota-se que as linhas relativas ao padrão de treinamento (verde) e ao resultado da RNA (vermelho) assumem o mesmo comportamento. Estes padrões se antecipam ao padrão de comportamento real (azul). Nota-se ainda que o δ deste pico suave gira em torno de 0.1. A Figura 6.4 traz um exemplo de PPB, onde os resultados de predição são certas vezes falhos.

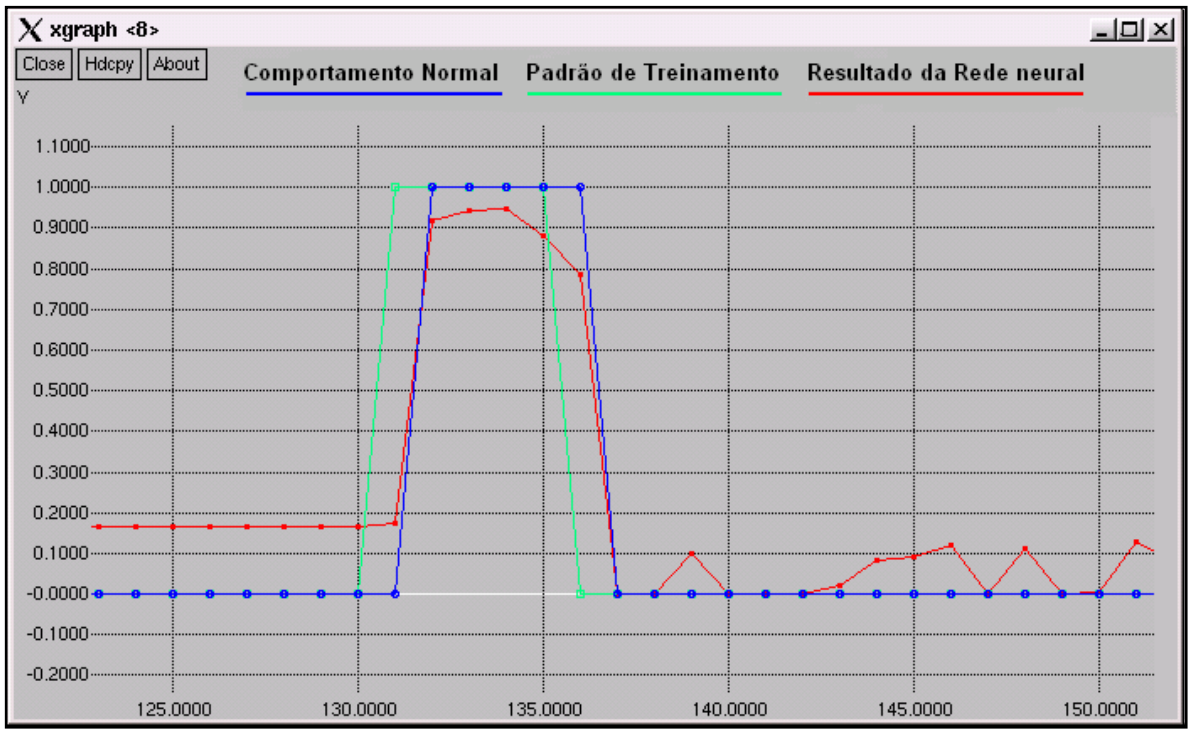

Figura 6.4 – Exemplo de erro na predição de falhas em pontos de pico.

Apesar do processo de formação dos PPSs e PPBs levar em consideração apenas três pontos, ou seja, os três últimos padrões em relação ao ponto em que se quer obter informações prévias, a topologia 12-20-1 (3 tempos x 4 parâmetros = 12 neurônios de entrada) para as redes neurais preocupadas com a predição ou inferência da BER não foi a mais indicada, pois não cumpriu o seu propósito. Como já exposto na Tabela 6.2 a topologia 20-20-1 foi a mais adequada para os três tipos de redes neurais. Os vinte neurônios de entrada desta topologia de RNAs devem ser necessários devido a todos os estados do *link* simulado e não somente em relação aos pontos de falhas.

6.3.1 Rede Neural Tipo 1

Este tipo de rede deve ter como entrada os valores advindos dos equipamentos da camada física, os quais estão relacionados apenas com o GDP, ficando o *ns* como gerador final dos *log*s, os quais formaram o banco de conhecimento. Os parâmetros de

entrada refletem o histórico da BER nos 2 (dois) últimos segundos, ou seja, o histórico da BER contém os valores dos últimos 19 (dezenove) décimos de segundos mais a BER do tempo presente. Esta janela de tempo foi determinada de forma empírica como citado no início da Seção 6.3. A saída da rede neural apresenta o valor da BER para o décimo de segundo imediatamente futuro. Assim, os valores da BER devem ser coletados nos instantes: *t-0,19, t-0,18, t-0,17,..., t-0,3, t-0,2, t-0,1* e *t*, e através da rede neural foi obtido o valor da BER no tempo *t+0,1,* onde *t* indica o tempo presente dado em segundos. No mundo real não é possível mensurar a BER diretamente de um sistema em operação [2], mas uma alternativa seria amplificar o sinal óptico ($\overline{P}_{rec}$) antes que o mesmo incida no fotodetector e calcular a BER segundo a fórmula:

$$BER = EXP\left(-\frac{2\overline{P}_{rec}}{hvB}\right)\Big/2,$$

a qual já foi explicada no Capítulo 3 (Fórmula 3.7). Atente que esta fórmula calcula a BER no limite quântico, ou seja, modela um receptor ideal. A escolha desta fórmula é um artifício de abstração, pois modelos mais precisos necessitam de variáveis relacionadas com o ruído de equipamentos ópticos, as quais estão além do escopo de atuação do GDP. A Tabela 6.3 apresenta os resultados obtidos em termos do MSE. A Figura 6.5 apresenta o gráfico do MSE do processo de treinamento.

Este tipo de rede apresenta um bom comportamento[21] em relação ao MSE de validação, que considera todos os pontos do banco de conhecimento `valida` (0,0437). O desempenho em relação ao banco de conhecimento `treina` (0,2612) pode ser considerado como razoável[22]. Quanto ao confronto entre os arquivos de resultado da rede (`*.res`) e os padrões de treinamento (`*.pat`) os resultados se mostram ora bons (`Testa.res x Testa.pat`), ora razoáveis (`Treina.res x Treina.pat` e `Valida.res x Valida.pat`) para $0,2 \geq \delta \geq 0,005$. Se o δ ficar fora desta faixa de valores, a predição fica comprometida devido à natureza brusca da falha.

---

[21] Bom comportamento: diz-se de uma rede neural que apresenta MSE ≤ 0,15.

[22] Comportamento razoável: diz-se da rede neural que apresenta MSE ≤ 0,25.

Tipo de Erro	**Resultados**		
MSE de Treinamento	0,2612		
MSE de Validação	0,0437		
MSE nos pontos de pico	Pontos de Pico	δ	MSE
Treina.res x Treina.pat	118	$0,2 \geq \delta \geq 0,01$	0,0407
	171	$0,2 \geq \delta \geq 0,005$	0,1628
	229	$0,2 \geq \delta \geq 0,001$	0,2950
Valida.res x Valida.pat	16	$0,2 \geq \delta \geq 0,01$	0,0437
	25	$0,2 \geq \delta \geq 0,005$	0,1816
	37	$0,2 \geq \delta \geq 0,001$	0,3228
Testa.res x Testa.pat	20	$0,2 \geq \delta \geq 0,01$	0,0401
	28	$0,2 \geq \delta \geq 0,005$	0,1389
	35	$0,2 \geq \delta \geq 0,001$	0,2586

Tabela 6.3 - Resultados da Rede Tipo 1.

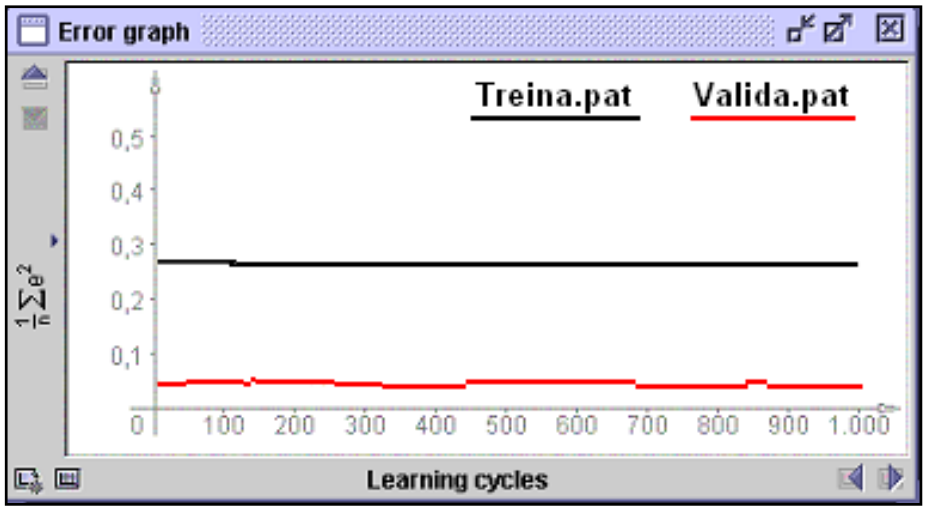

Figura 6.5 – Gráfico MSE de treinamento da rede neural tipo 1.

6.3.2 Rede Neural Tipo 2

Para esta rede também foi utilizada a topologia 20-20-1 e o algoritmo de aprendizagem *backpropagation.* Os parâmetros de entrada foram: o *status* do *link*, a quantidade de banda passante ocupada, os pacotes recebidos e os pacotes perdidos, enquanto a saída fez referência a taxa de erro de bits (BER) do tempo presente em forma percentual, ou seja, uma BER de 0,4 indica que a taxa de bits errados gerados é da ordem de 40%. No Capítulo 4 foram indicadas variáveis candidatas a entrada da rede neural, tais como, o comprimento de onda, a potência máxima do sinal de cada comprimento de onda e o OSNR de cada canal DWDM. Parâmetros como estes são mais adequados à rede do tipo 1 e não foram considerados devido a falta de infra-estrutura laboratorial disponível para este trabalho. As variáveis de entrada da rede tipo 2 foram coletadas nos instantes: *t-0,4, t-0,3, t-0,2, t-0,1* e *t*, sendo o tempo *t* é dado em décimo de segundo. A saída da rede indica a BER no instante *t*.

A rede neural tipo 2 apresentou resultados excelentes para a inferência da BER da camada física do instante presente. O MSE, tanto em relação ao conjunto total dos parâmetros de treinamento, como em relação aos pontos de pico suaves apresentou bons valores como demonstram a Tabela 6.4 e o gráfico da Figura 6.6.

A clareza minoritária dos valores dos dois tipos de erros calculados demonstra claramente que é possível inferir a BER da camada física a partir do comportamento do tráfego da rede de computadores. Entretanto, este tipo de rede não obteve resultados satisfatórios quando se tentou usá-la como mecanismo de predição de falhas. A Figura 6.7 mostra quanto o resultado fornecido pela rede neural (vermelho) se aproxima do padrão de treinamento (verde). Os dados envolvidos no gráfico da Figura 6.7 são relativos ao banco de conhecimento `testa.pat`.

Tipo de Erro	**Resultados**		
MSE de Treinamento	0,0105		
MSE de Validação	0,0015		
MSE nos pontos de pico	Pontos de Pico	δ	MSE
Treina.res x Treina.pat	118	$0,2 \geq \delta \geq 0,01$	0,0123
	171	$0,2 \geq \delta \geq 0,005$	0,0562
	229	$0,2 \geq \delta \geq 0,001$	0,0420
Valida.res x Valida.pat	16	$0,2 \geq \delta \geq 0,01$	0,0004
	25	$0,2 \geq \delta \geq 0,005$	0,0544
	37	$0,2 \geq \delta \geq 0,001$	0,0367
Testa.res x Testa.pat	20	$0,2 \geq \delta \geq 0,01$	0,0280
	28	$0,2 \geq \delta \geq 0,005$	0,0726
	35	$0,2 \geq \delta \geq 0,001$	0,0581

Tabela 6.4 - Resultados da Rede Tipo 2.

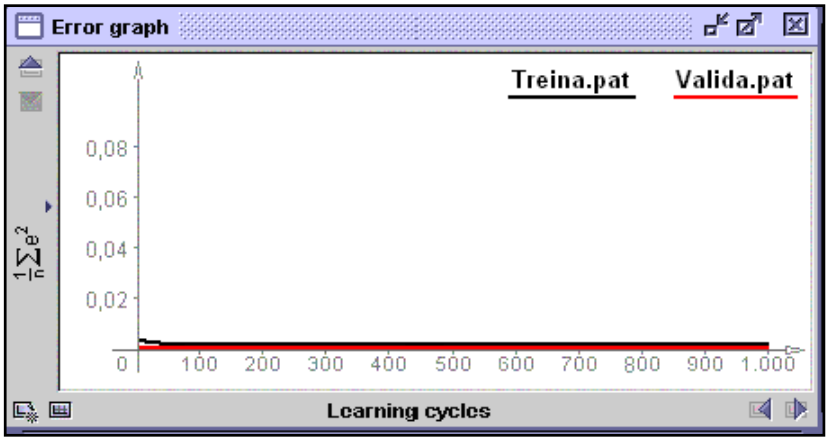

Figura 6.6 – Gráfico MSE de treinamento da rede neural tipo 2.

Um dos parâmetros de entrada para este tipo de rede foi o *status* do *link.* Como já foi dito, este impede a confusão pelas redes neurais entre os padrões de silêncio e de falha, entretanto, tal parâmetro pode influenciar excessivamente no resultado da rede. Para verificar este fato, foram retirados tanto o parâmetro *status do link,* como os padrões de silêncio do banco de conhecimento. Os resultados obtidos foram piores que os apresentados anteriormente, mas não deixaram dúvidas quanto ao bom desempenho da mesma, ainda que o parâmetro em questão fosse suprimido, como comprovam para este caso o MSE de treinamento de 0,1362 e o MSE de validação 0,0234.

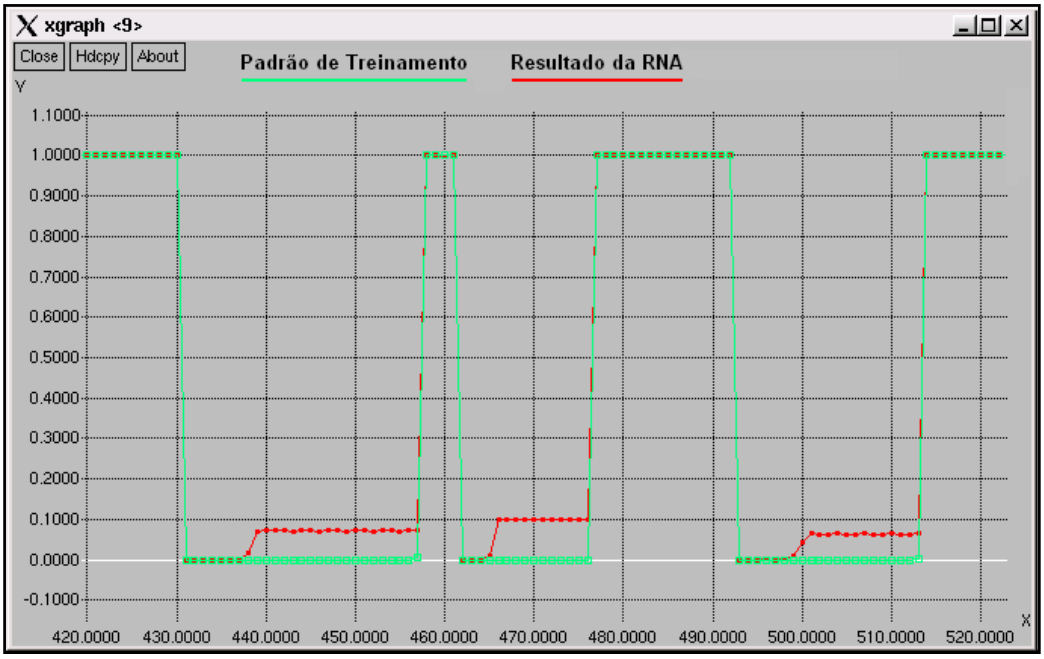

Figura 6.7 –Saídas dos arquivos Testa.res x Testa.pat sob a rede neural tipo 2.

6.3.3 Rede Neural Tipo 3

Inicialmente, pretendia-se usar a topologia da rede tipo 3 para a predição de falhas do *link* simulado. Este tipo de rede tenta inferir a próxima taxa de erro a partir dos instantes anteriores da simulação. Então de posse do comportamento do tráfego (*statusLink, UsedBand, PktLost, PktOk*) nos instantes *t-0,4, t-0,3, t-0,2, t-0,1 e t* pretendia-se obter a BER da camada física do instante *t+0,1*. O MSE de validação apresentado para redes deste tipo pode ser caracterizado como razoável, entretanto, o

erro nos pontos de pico suaves se mostrou totalmente falho. A Tabela 6.5 e o gráfico da Figura 6.8 documentam o resultado obtido por este tipo de rede.

Tipo de Erro	**Resultados**		
MSE de Treinamento	0,4103		
MSE de Validação	0,0536		
MSE nos pontos de pico	Pontos de Pico	δ	MSE
Treina.res x Treina.pat	118	$0,2 \geq \delta \geq 0,01$	0,8518
	171	$0,2 \geq \delta \geq 0,005$	0,8609
	229	$0,2 \geq \delta \geq 0,001$	0,8609
Valida.res x Valida.pat	16	$0,2 \geq \delta \geq 0,01$	0,8573
	25	$0,2 \geq \delta \geq 0,005$	0,8399
	37	$0,2 \geq \delta \geq 0,001$	0,8514
Testa.res x Testa.pat	20	$0,2 \geq \delta \geq 0,01$	0,8876
	28	$0,2 \geq \delta \geq 0,005$	0,8574
	35	$0,2 \geq \delta \geq 0,001$	0,8630

Tabela 6.5 - Resultados da Rede Tipo 3.

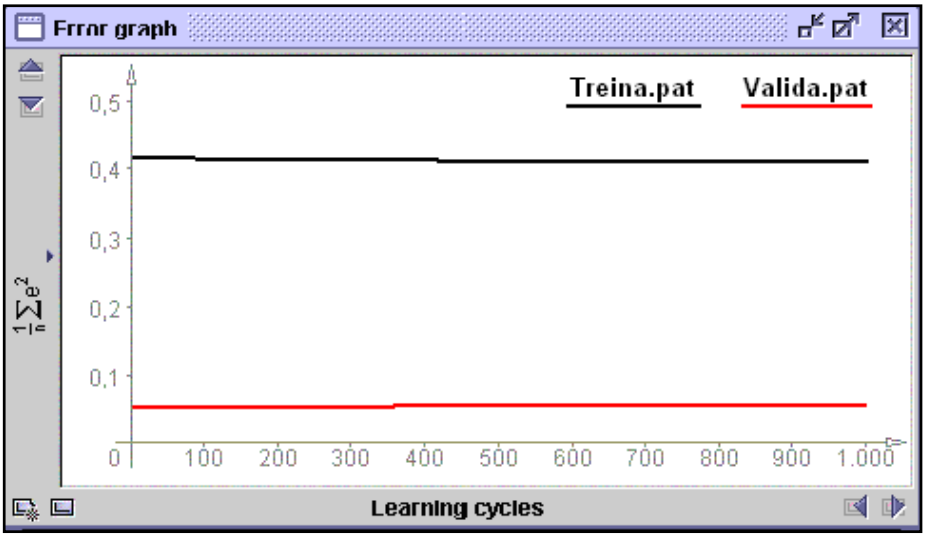

Figura 6.8 – Gráfico MSE de treinamento da rede neural tipo 3.

O estado falho evidente desta rede comprova que os parâmetros de entrada status do link, banda passante ocupada, pacotes recebidos e pacotes perdidos são insuficientes para realizar a predição desejada neste contexto. Futuramente, pode-se tentar melhorá-los de forma que o tráfego seja mais precisamente descrito e, conseqüentemente, melhorar os resultados finais esperados.

6.4 Conclusões

A forma mais comum de se verificar se uma RNA, depois de treinada, atingiu seu propósito é através do MSE. Para este experimento, além de um bom comportamento do MSE relacionado com todo o banco de conhecimento, este teve também de apresentar bons resultados em relação aos PPSs e, se possível, aos PPBs, uma vez que estes são os pontos de uma eventual transição entre os estados de funcionamento normal e de falha do *link* estudado.

O processo de experimentação demonstrou que é plausível uma investigação de como se prever ou inferir falhas da camada física. Três tipos de RNAs (tipo 1, 2 e 3) foram construídas, segundo ferramentas de simulações, para atuar no problema de re-roteamento de redes IP+GMPLS sobre DWDM. É possível usar RNAs que utilizem tanto informações sobre o tráfego (tipo 2), bem como informações relacionadas com o próprio equipamento físico (tipo 1). A RNA tipo 2 pode detectar falhas em um plano de mais alto nível e fazer inferências sobre a camada física, para isso usa variáveis da camada de rede, mas não consegue indicar a entidade que causou a falha. A RNA tipo 1 faz predições sobre a BER da camada física e utiliza variáveis de entrada relacionadas com esta mesma camada. Com a melhoria da RNA tipo 1, esta também poderá verificar qual é a entidade física causadora de falhas, pois suas variáveis de entrada modelam os próprios equipamentos físicos em questão. Para isso, é necessária uma interface de comunicação entres as RNAs e o equipamento físico. Outra possibilidade seria a implementação de um *hardware* embarcado que implementasse as RNAs diretamente no equipamento desejado. Ambas opções, interface e *hardware* embarcado, são deixados como trabalhos futuros.

A RNA do tipo 2 estimou a BER em um tempo presente e a RNA tipo 1 faz a inferência em relação ao futuro. A RNA tipo 2 poderia fornecer a entrada da RNA tipo 3 para fazer inferências sobre a BER da camada física usando variáveis da camada de rede. Tal disposição não é viável pois tal sistema é "equivalente" a concatenação das estruturas das RNAs. Isso acarreta a propagação do erro camada a camada em ambas as redes neurais além de gerar um excesso de camadas na estrutura supostamente equivalente. A RNA tipo 3 trabalha com as mesmas variáveis de entrada da RNA tipo 2 e fornece o mesmo tipo de resultado de saída da RNA tipo 1. A estrutura conexionista do tipo 3 não funcionou satisfatoriamente. Conclui-se que, operando separadamente, as redes neurais tipo 1 e tipo 2 atuam bem, conforme seus propósitos, mas em conjunto não obtiveram resultados satisfatórios, seja na forma de cascata (tipo 1 alimentando tipo 2), seja na forma encapsulada (tipo1 e tipo 2 na mesma estrutura) Desta forma não foi possível prever falhas de um *link* óptico segundo o tráfego da rede, mas somente em função de informações fornecidas pela própria camada física. Esta não é uma questão definitiva, pois uma falha causada por um equipamento físico acarreta um crescimento exponencial da BER o que requer variáveis de tráfego extremante precisas (da ordem de 10^{-9}). Logo, se o banco de conhecimento for melhorado de forma a fornecer mais subsídios ao treinamento das RNAs, o objetivo de fazer predições de falhas com informações a respeito do tráfego pode ser melhorado.

O processo de simulação influi diretamente no resultado final apresentado pelas RNAs. A preocupação em se aproximar o banco de conhecimento do mundo real é de suma importância para a qualidade dos mecanismos de prevenção de falhas. Quanto mais próxima do mundo real for a simulação, mais confiável serão os resultados obtidos pelas RNAs. O próximo capítulo descreve em detalhes como o experimento foi desenvolvido, bem como a seqüência de passos a ser seguida para gerar tais redes neurais.

Capítulo 7

Cenário de Implementação

Este capítulo tem por finalidade descrever o processo que foi utilizado para gerar as simulações necessárias ao desenvolvimento das RNAs aplicadas ao ambiente RENATA 2, comentar os resultados obtidos e documentar o trabalho feito de forma a facilitar futuras evoluções.

7.1 Implementação do GDP

O software GDP (Gerador de Perturbações) teve um papel fundamental para gerar com qualidade as informações do banco de conhecimento usado no treinamento das RNAs usadas na predição e inferência de falhas. Ele complementa o *ns* no âmbito da simulação de falhas em redes ópticas. Para implementação do GDP utilizou-se Linguagem Java, de forma a lhe garantir o máximo de portabilidade. O *ns* é um software livre com versões para ambientes Unix/Linux e Windows. Neste trabalho, foi usada a versão 2.1b8a do *ns* para ambiente Linux. No contexto do RENATA 2, o *ns* recebe informações diretamente do GDP, logo é oportuno que os usuários do RENATA 2 possam executar o GPD em um sistema operacional que o *ns* aceite.

No projeto e na implementação do GPD teve-se a preocupação de tornar sua manutenção e futuras evoluções simples e rápidas. Assim, foi adotado o paradigma de orientação a objetos nestas duas etapas, como também optou-se em separar o software em duas camadas: a camada de interface com o usuário e a camada de implementação

propriamente dita. As 2747 linhas de código do GDP foram divididas entre 16 classes Java. As classes abaixo estão relacionadas com a interface gráfica do usuário:

- ✓ `FrComp,`
- ✓ `FrConfiguracoes,`
- ✓ `FrFibraOp,`
- ✓ `FrMain,`
- ✓ `FrReceptor, e`
- ✓ `FrTransmissor.`

As demais classes estão relacionadas com as funcionalidades propriamente ditas do gerador de perturbações e podem ser definidas como sendo o núcleo do GDP:

- ✓ `FibraOp,`
- ✓ `GeradorPerturbacoes,`
- ✓ `PerturbAleatoria,`
- ✓ `PerturbAleatoriaLinear,`
- ✓ `PerturbLinear,`
- ✓ `Perturbacao,`
- ✓ `Receptor,`
- ✓ `Reta,`
- ✓ `Trans, e`
- ✓ `Transmissor.`

A Figura 7.1 mostra o diagrama classes em UML (*Unified Modeling Language*) do GDP, onde o relacionamento entre tais classes é exposto, ficando o diagrama de seqüência na Figura 7.2. A Tabela 7.1 traz comentários relacionados com cada uma das classes desse software que explicam as principais especificidades do código e o propósito de cada documento de código fonte.

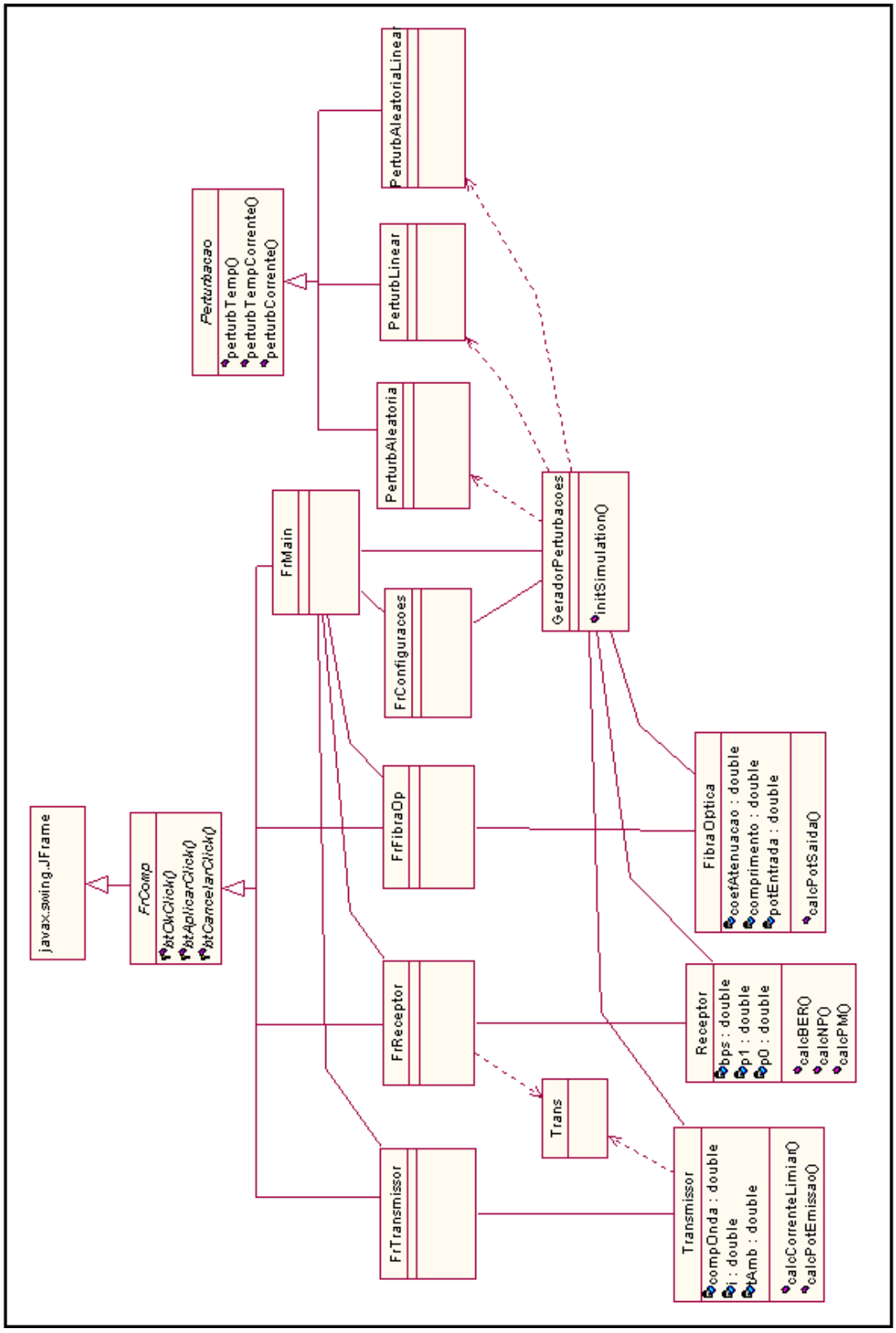

Figura 8.1 – Diagrama de Classes em UML do GDP.

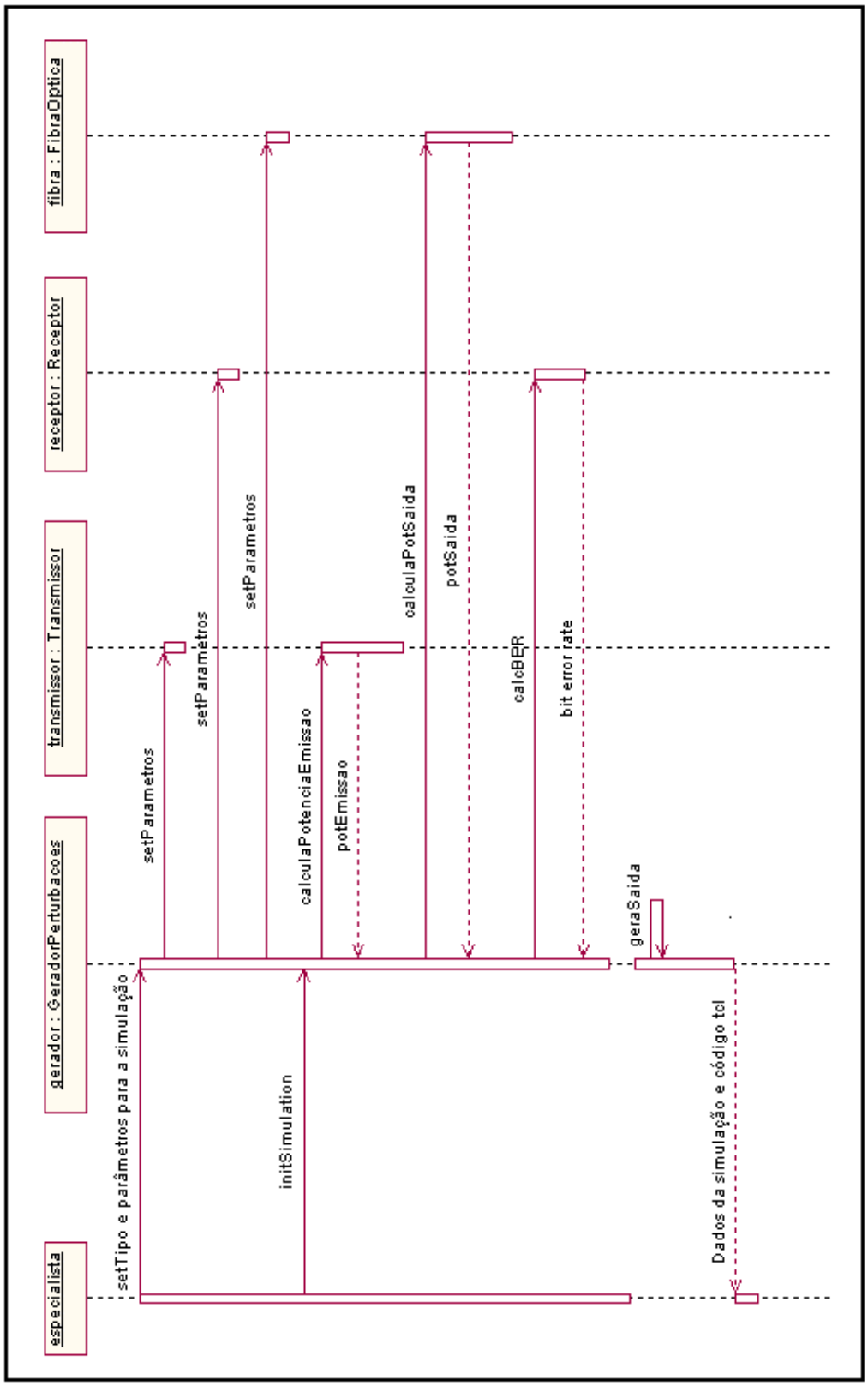

Figura 8.2 – Diagrama de Seqüência do GDP.

Classe GDP	Função
FrComp	Classe pai, responsável pela codificação comum das janelas relacionadas com os componentes ópticos: Fibra, Transmissor e Receptor e também a janela de configurações.
FrConfigurações	Janela onde o usuário define os parâmetros gerais da simulação.
FrFribra	Local de definição dos parâmetros da fibra óptica.
FrMain	Janela de execução inicial do sistema.
FrReceptor	Local de definição dos parâmetros do receptor.
FrTransmissor	Janela de definição dos parâmetros do transmissor.
FibraOp	Classe que abstrai um enlace de fibra óptica.
GeradorPerturbacoes	Classe que trata o gerador de perturbações como um todo.
PerturbAleatória	Trata de perturbações do tipo aleatória.
PerturbAleatóriaLinear	Abstrai as perturbações do tipo aleatória linear.
PerturbLinear	Idem ao anterior para perturbações do tipo linear.
Perturbação	Classe pai na hierarquia das perturbações.
Receptor	Abstrai um receptor óptico.
Reta	Usada para dar suporte as perturbações que têm um caráter linear e abstrai o comportamento dos pontos de uma reta.
Trans	Classe que trata algumas transformações de unidades, por exemplo, mW em dB.
Transmissor	Classe que abstrai um Transmissor Óptico.

Tabela 7.1 - Descrição das Classes do *GDP*.

7.2 Simulações do GDP - Perturbações

No GDP foram feitas 176 simulações no total. Tais simulações tiveram o objetivo de validar o ambiente RENATA 2 em situações de falhas de um transmissor óptico. Para isso, provocou-se a variação de grandezas físicas que influenciam no comportamento da potência do sinal emitido por um laser transmissor, ou seja, dentro de cada simulação foram geradas perturbações nos valores de temperatura e corrente que interferem no comportamento do laser transmissor. As simulações feitas no GDP foram divididas em três categorias:

I. Simulações de perturbação na temperatura;
II. Simulações de perturbação na corrente; e
III. Simulações de perturbação simultânea na temperatura e na corrente.

Dentre as 176 simulações, 62 foram de perturbações na temperatura, 52 estavam relacionadas com corrente e as demais 62 com ambas grandezas. Inicialmente cada uma destas categorias deveria possuir 65 simulações, entretanto, algumas simulações de perturbações não agregaram conhecimento relevante ao processo de aprendizagem das redes neurais e foram descartadas por serem redundantes. As 176 simulações geraram uma carga de simulações finais (1232 simulações do *ns*) bem razoável para a execução das mesmas nos recursos computacionais disponíveis (equipamentos Pentium 4 com 192 MB de RAM). O processo de transformação de simulações do GDP em simulações do *ns* será explicitado com maior detalhamento no decorrer desse capítulo. Cada categoria de simulação do GDP foi dividida em cinco subcategorias, nas quais cada grandeza relacionada variava de cinco modos diferentes:

1. Linear Crescente;
2. Linear Decrescente;
3. Aleatória Linear Crescente (valores aleatórios, mas sempre crescentes);
4. Aleatória Linear Decrescente (valores aleatórios, mas sempre decrescentes);
5. Aleatória (valores aleatórios limitados a uma faixa de valores).

Para maiores informações, sobre as formas de variação já citadas, consulte a Seção 5.1.2 GDP - Gerador de Perturbações. A Seção citada também é importante

para o entendimento da Tabela 7.2 que apresenta um histórico das simulações feitas no GDP. Nesta tabela, os termos decremento e incremento presentes, respectivamente, nas categorias de perturbação Linear Decrescente e Linear Crescente, indicam a razão de uma PA (Progressão Aritmética) de a_n termos, na qual o termo a_1 é descrito na tabela como Valor Final 1 e a_n como Valor Final n. O Valor Inicial da grandeza física perturbada (temperatura e/ou corrente) em cada perturbação é sempre o mesmo. Cada termo da PA indica o valor que a grandeza assumiu ao final de cada perturbação e o número de termos indica a quantidade de simulações feitas. Nas simulações de caráter aleatório o termo "Num. Repetições" indica o número de simulações feitas e o termo "Variação Max." indica máxima variação que a grandeza pode sofrer em cada instante de perturbação. Na categoria Aleatória Linear, a variação da grandeza perturbada é estritamente crescente ou decrescente conforme o caso. A Figura 7.3 mostra um exemplo de como uma simulação de perturbação aleatória ocorre. Cada simulação do GDP teve duração de 3s, sendo 2,5s o período de perturbação de cada grandeza. O início da perturbação de cada simulação ocorria no tempo 0,25s e finalizava no tempo 2,75s. O intervalo de amostragem de valores se deu a cada 0,001 s da simulação.

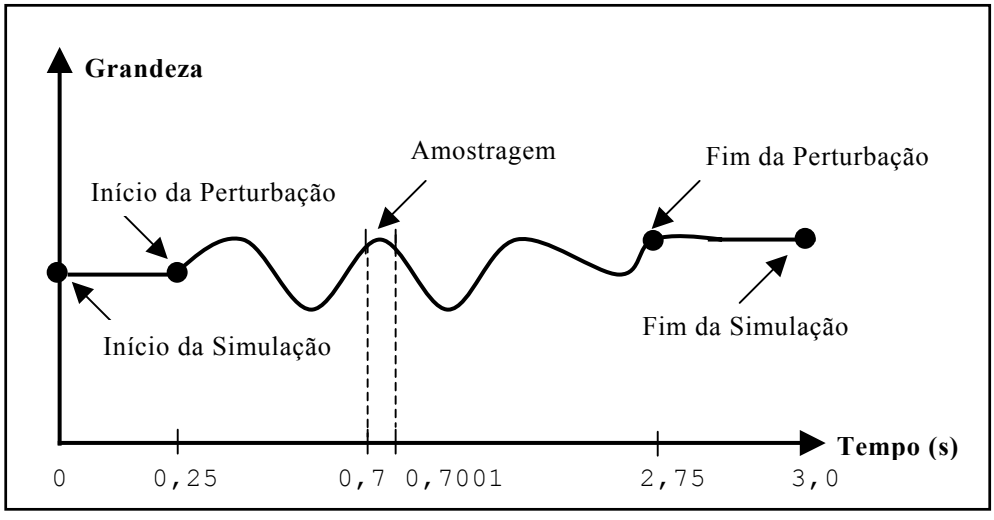

Figura 7.3 – Exemplo de andamento de uma simulação do GDP.

I. Temperatura		II. Corrente		II. CORREnte e TEMPeratura		
I.1 Linear Decrescente		II.1 Linear Decrescente		III.1 L. Decrescente	Corre.	Temp.
Decremento	-0,50	Decremento	-0,25	Decremento	-0,25	-0,50
Valor Inicial	25,00	Valor Inicial	20,85	Valor Inicial	20,85	25,00
Valor Final 1	24,50	Valor Final 1	20,60	Valor Final 1	20,60	24,50
Valor Final n	18,00	Valor Final n	18,85	Valor Final n	18,85	18,00
Total	14,00	Total	8,00	Total	8,00	14,00
I.2 Linear Crescente		II.2 Linear Crescente		III.2 L. Crescente	Corre.	Tmp.
Incremento	0,50	Incremento	0,25	Incremento	0,25	0,50
Valor Inicial	25,00	Valor Inicial	20,85	Valor Inicial	20,85	25,00
Valor Final 1	25,50	Valor Final 1	21,10	Valor Final 1	21,10	25,50
Valor Final n	31,00	Valor Final n	22,85	Valor Final n	22,85	31,00
Total	12,00	Total	8,00	Total	8,00	12,00
I.3 A. L. Crescente		II.3 A. L. Crescente		III.3 A. L. Crescente	Corre.	Temp.
Valor Inicial	25,00	Valor Inicial	20,85	Valor Inicial	20,85	25,00
Variação Max.	0,005	Variação Max.	0,003	Variação Max.	0,003	0,005
Num. Repetições	12,00	Num. Repetições	12,00	N. Repetições	12,00	12,00
Total	12,00	Total	12,00	Total	12,00	12,00
I.4 A. L. Decrescente		II.4 A. L. Decrescente		III.4 A.L. Decrescente	Corre.	Temp.
Valor Inicial	25,00	Valor Inicial	20,85	Valor Inicial	20,85	25,00
Variação Máx.	-0,005	Variação Max.	-0,003	Variação Máx.	-0,003	-0,005
Num. Repetições	12,00	Num. Repetições	12,00	N. Repetições	12,00	12,00
Total	12,00	Total	12,00	Total	12,00	12,00
I.5 Aleatória		II.5 Aleatória		III.5 Aleatória	Corre.	Temp,
Valor Inicial	25,00	Valor Inicial	20,85	Valor Inicial	20,85	25,00
Variação Max.	0,50	Variação Max.	0,30	Variação Máx.	0,30	0,50
Num. Repetições	12,00	Num. Repetições	12,00	N. Repetições	12,00	12,00
Total	12,00	Total	12,00	Total	12,00	12,00
Total I	**62,00**	**Total II**	**52,00**	**Total III**		**62,00**

Tabela 7.2 - Detalhamento das simulações de perturbações feitas no GDP.

7.2.1 Processo de Montagem de Simulações do GDP – Matriz de Simulações

A forma de comunicação entre o GDP e o *ns* se dá através de *tags*, então o usuário do RENATA 2 deve inserir a *tag #GDP* no arquivo de simulação do *ns* (*script* *.tcl) para que o GDP possa adicionar informações relativas às simulações da camada física. Assim, o *ns* pode assumir um comportamento mais próximo do mundo real no que se refere a falhas em *links* ópticos. Ao final de cada simulação do GDP, o usuário pode informar o arquivo de *script* do *ns* (com a *tag* #GDP devidamente posicionada) e o GDP fornecerá um novo *script* do *ns* já com as informações sobre as perturbações realizadas na simulação corrente. Desta forma, para cada simulação do GDP haverá uma simulação do *ns*. Ao conjuto de simulações do *ns*, as quais têm um mapeamento direto com as simulações do GDP, dá-se o nome de **Matriz de Simulações**. Esta matriz, por sua vez, serve de base para gerar as simulações finais usadas para construção do **Banco de Conhecimento** das redes neurais. Logicamente, a Matriz de Simulações foi formada por 176 simulações do *ns*, todas idênticas quanto a características do tráfego gerado e diferentes em relação ao comportamento do enlace óptico simulado.

7.3 Simulações do *ns*

As simulações do *ns* são as responsáveis em gerar os *logs* finais, os quais, de maneira não formatada, formam a base de conhecimento para o aprendizado das redes neurais. As simulações em questão têm a preocupação de simular as camadas de enlace e de rede, bem como tratam das especificidades do MPLS. Basicamente, cada simulação do *ns* gera um *log* que contém valores para o quinteto ordenado (*status do link*, quantidade de banda passante ocupada, pacotes recebidos, pacotes perdidos e BER – *Bit Error Rate* da camada física) citado anteriormente na Seção 6.2.

Cada simulação teve a duração de 3s (três segundos) e o intervalo de amostragem de cada quinteto foi feito a cada 0,1s (um décimo de segundo). A amostragem poderia ser feita até em 0,001s (um milésimo de segundo), uma vez que

as simulações do GDP foram projetadas para fornecer informações até este nível de detalhe e, assim, capturarem informações mais precisas (em termos temporais) sobre o sistema simulado. Entretanto, quando a amostragem das simulações do *ns* foi feita em milésimos de segundo, este se comportou de forma muito lenta, cerca de 1,5[23] minuto para cada simulação, o que resultaria em aproximadamente 30,8 horas para executar todos os *scripts* de simulações finais. Com a configuração de décimo de segundo, consegui-se reduzir o tempo de simulação para apenas cerca de 2 horas e 45 minutos, obviamente, com prejuízos para apontar o instante exato da ocorrência de um evento no sistema simulado. De qualquer forma, a coleta de dados dos *logs* do *ns* foi feita em décimos de segundo para permitir que a rede de computadores tenha tempo suficiente para realizar o re-roteamento em caso futuras falhas.

7.3.1 Processo de montagem de simulações do *ns* - Simulações Finais

Como já exposto, a Matriz de Simulações guarda o conhecimento relacionado com o comportamento da camada física. Ela se comporta de maneira idêntica quando se considera o tráfego gerado. Assim para cada simulação da matriz em questão, foram geradas 7 (sete) simulações nas quais a vazão de tráfego gerado assumiu os seguintes valores: 0, 2, 4, 6, 8, 10, 12 Gbps. Como a matriz conta com 176 simulações e cada uma destas simulações assumiu 7 configurações de vazão de tráfego diferente, então o número final de simulações a serem executadas no *ns* foram 1232, denominadas de **Simulações Finais**. Vale ressaltar, mais uma vez, que tal disposição de tráfego é necessária para tentar aproximar, o quanto possível, as simulações do mundo real e assim melhorar a qualidade do banco de conhecimento gerado. Desta forma, as redes neurais poderão assumir um papel mais confiável em relação a entradas inéditas da rede, ou seja, pretende-se fornecer uma melhor qualidade de subsídios para que as redes neurais possam generalizar com mais precisão.

---

[23] Tempo necessário para execução do *script TCL* da simulação (ns) inicial em um computador Pentium III com 196 MB de RAM e usando o sistema operacional Linux distribuição Conectiva 8.

Em cada um dos arquivos da Matriz de Simulações foram colocadas mais duas t*ags*: `#MSPDflg` (MSPD *Flag*) e `#MSPDbnd` (MSPD *Band*). Estas marcavam o local do *script ns* onde estavam o nome do arquivo de *log* e a vazão gerada pelo agente produtor de tráfego. Desta forma, o MSPD através da classe `CorrecLog` substitui as *tags* pelos valores apropriados, então desta maneira as simulações finais foram geradas.

Cada simulação final gerou um arquivo de *log* que continham os parâmetros de entrada e de saída necessários ao treinamento das redes neurais. Tais informações encontravam-se dispersas em 1232 arquivos de *logs* e os valores dos parâmetros capturados pelos mesmos não se encontravam devidamente escalonados, de forma que pudessem servir de entrada para redes neurais. Para resolver este problema, foi escrita a classe `GeraBanco` do MSPD, que fez o escalonamento dos parâmetros colhidos e os agrupou em três arquivos tipo texto ASCII chamados de `Treina.bnc`, `Valida.bnc` e `Testa.bnc` Estes três arquivos formavam o **Banco de Conhecimento**.

O arquivo `Treina.bnc` ficou com aproximadamente 78% das informações fornecidas pelas simulações finais e foi considerado como a base do aprendizado das redes neurais. O arquivo `Valida.bnc` e `Testa.bnc` serviram, como suas denominações já dizem, de base para validação e testes das redes neurais já treinadas e contavam, cada um, com 11% das informações geradas pelas simulações finais. O banco de conhecimento ainda não está em formato de **Padrões de Treinamento**, os quais o simulador de redes neurais JNNS possa receber, como também não é este o seu intuito. O banco de conhecimento serve para que o MSPD, através das classes codificadas especialmente para este caso, possa gerar padrões de entrada para o JNNS de forma a gerar redes neurais de diversas topologias.

7.4 Montagem dos Padrões de Treinamento

Conforme detalhado no capítulo 6, foram gerados três tipos de rede neurais, diferindo entre si nos parâmetros de entrada, nos resultados fornecidos e no seu propósito. Uma breve explicação de cada um dos tipos de redes é necessária para que

se possa compreender melhor a atuação do MSPD, que é o responsável em formatar o banco de conhecimento tal que cada tipo de rede neural possa usá-lo. Abaixo é feita tal descrição:

- ✓ **Rede Neural Tipo 1** – Tem o propósito de predizer um estado de erro futuro (BER). Como entrada possui um histórico da BER dos últimos vinte décimos de segundo da simulação e aponta o valor futuro da BER;
- ✓ **Rede Neural Tipo 2** – A função desta rede é ler o *status* do *link*, a quantidade de banda passante ocupada, os pacotes recebidos e os pacotes perdidos, e inferir a taxa de erro de bits (BER) do tempo presente. Note que os três últimos parâmetros de entrada estão relacionados com o tráfego. Em termos de histórico, são necessários os cinco últimos décimos de segundo da simulação em relação ao instante a ser inferido.
- ✓ **Rede Neural Tipo3** – Trata-se da fusão das redes tipo 1 com as redes tipo 2 e tem como objetivo inferir a BER futura em função dos mesmos parâmetros de entrada da rede tipo 2. Para isso, são lidos os cinco últimos décimos de segundo da simulação e tenta prever o próximo valor da BER.

A classe do MSPD `GeraInRNAAtual` é a responsável por gerar entradas para a Rede Neural Tipo 2. Por exemplo, caso o usuário deseje gerar uma rede deste tipo com 20 nós de entrada (últimos cinco décimos de segundo de histórico), um padrão de treinamento de número *n* gerado para JNNS pela classe `GeraInRNAAtual` usará os parâmetros de entrada presentes nos quintetos *n-4, n-3, n-2, n-1, n* e o parâmetro de saída do quinteto *n* do banco de conhecimento. A classe `GeraInRNAProx` se comporta de forma semelhante, diferindo apenas no uso do parâmetro de saída, que no caso do exemplo anterior usaria o quinteto *n+1* para este fim e se relaciona com a Rede Neural Tipo3. Este tipo se propõe a predizer situações de falhas baseadas no tráfego e apresentou bom desempenho em relação ao erro médio de validação. Entretanto, apresentou fraco desempenho em relação ao erro médio calculado apenas nos pontos de pico, conforme exposto no Capítulo 6. Quanto à Rede Neural Tipo 1, a classe GeraInRNABER também do MSPD é responsável por gerar padrões de treinamento para este tipo de rede, na qual só é levada em consideração BER da camada física. A Tabela 7.3 traz uma descrição de cada uma das classes do

MSPD, as quais são bastantes simples e podem ser implementadas em qualquer linguagem de programação.

Classe MSPD	Função
`CorrecLog`	Substitui as *tags* `#MSPDflg`,`#MSPDbnd` pelos valores apropriados para gerar as Simulações Finais.
`GeraBanco`	A partir dos *logs* das simulações finais gera o Banco de Conhecimento.
`GeraInRNABER`	Gera os padrões de treinamento das redes Tipo 1.
`GeraInRNAAtual`	Gera os padrões de treinamento das redes Tipo 2.
`GeraInRNAProx`	Gera os padrões de treinamento das redes Tipo 3.
`CalcErro`	Calcula o erro médio nos pontos de pico usando as saídas de RNA.

Tabela 7.3 - Descrição das classes do MSPD.

7.5 Cálculo do MSE nos Pontos de Pico Suaves

As construções dos três tipos de redes neurais usaram o mesmo banco de conhecimento para treinar, validar e testar as respectivas redes. Para isso, o MSPD também foi projetado para adequar o formato do banco de conhecimento às restrições impostas pela topologia da rede neural a ser treinada. Para todas as redes neurais, os padrões de treinamento, validação e teste (arquivos `.pat`) foram formados, respectivamente, pelos arquivos `treina.bnc`, `valida.bnc` e `teste.bnc`. Ao final do processo de treinamento, o JNNS já fornece o MSE para os arquivos `treina.bnc` e `valida.bnc`, bem como o gráfico de andamento deste processo onde são relacionados os ciclos de treinamento com o MSE. Depois do treinamento pode-se gerar os arquivos `treina.res`, `valiada.res` e `testa.res` os quais contêm os resultados inferidos pela rede neural em questão. Assim, a classe `CalcErro` do MSPD pode calcular o MSE nos Pontos de Picos Suaves, comparando cada um dos arquivos `.pat` com seu respectivo `.res`. A Figura

7.4 mostra o diagrama de seqüência do sistema RENATA 2 completo, que servirá também para guiar o usuário do RENATA 2 no processo de geração das redes neurais finais. Em resumo, para gerar e validar uma rede neural no ambiente RENATA 2 deve-se seguir os seguintes passos:

1. Construir a simulação inicial da rede a ser modelada através de um *script* do *ns* com as respectivas *tags* de comunicação com o GDP, destacando o *link* a ser simulado (Anexo B comenta um exemplo desta simulação);
2. Fazer as simulações dos equipamentos ópticos do *link* simulado no GDP e gerar as respectivas simulações do *ns* tomando como base a simulação desenvolvida no Passo 1;
3. Usar o MSPD para atualizar as simulações do Passo 2 quanto a vazão de tráfego e nomenclatura dos arquivos de *log;*
4. Executar no *ns* as simulações atualizadas pelo Passo 3 com o intuito de gerar os arquivos de *log* necessários à construção do banco de conhecimento.
5. Usar o MSPD para transformar os arquivos de *log* no banco de conhecimento;
6. A partir do banco de conhecimento usar o MSPD para gerar os padrões de treinamento das redes neurais conforme a topologia desejada;
7. Construir a rede neural desejada no JNNS e realizar o processo de aprendizagem da mesma segundo os padrões de treinamento gerados no Passo 6;
8. Além de verificar o comportamento do MSE fornecido pelo JNNS verificar o MSE nos pontos de pico também com o uso do MSPD.

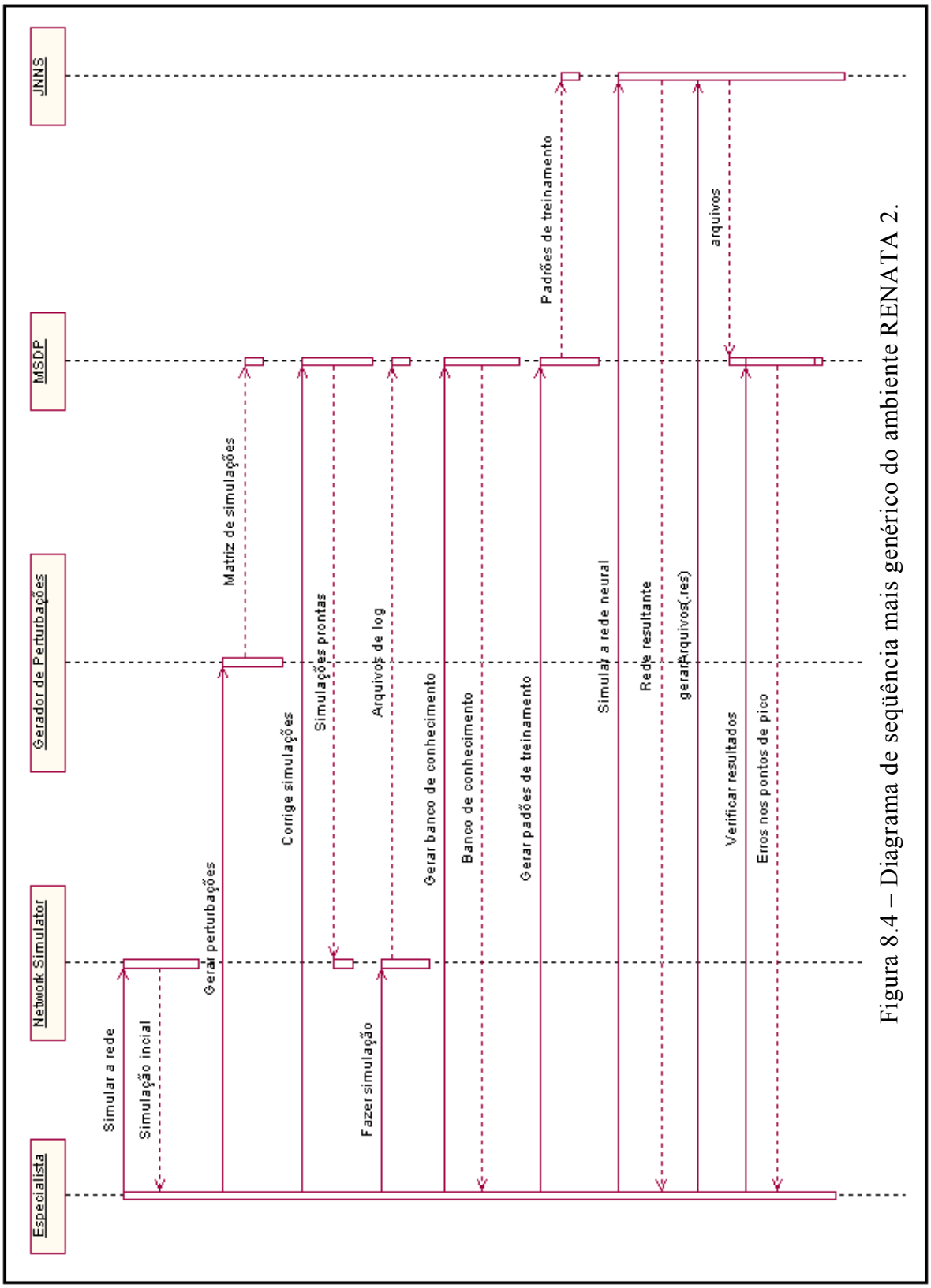

Figura 8.4 – Diagrama de seqüência mais genérico do ambiente RENATA 2.

7.6 Conclusões

As ferramentas implementadas GDP e MSPD procuram ligar as várias etapas do desenvolvimento das redes neurais aplicadas ao gerenciamento de falhas proposto por este trabalho. Quanto mais evoluídas forem estas ferramentas, mais simples o processo supracitado tornará-se-á e usuários (gerentes e administradores de redes) não especializados em rede neurais poderão desenvolver este processo normalmente. Este capítulo teve como objetivo documentar o desenvolvimento das ferramentas GDP e MSPD, para facilitar futuras atualizações e assim contribuir futuramente para um uso mais amplo do ambiente RENATA 2. Outro propósito foi deixar claro a seqüência de passos a ser seguida para construção dos agentes RENATA 2.

Capítulo 8

Conclusões e Trabalhos Futuros

Esta dissertação consolida o ambiente RENATA, que se propõe a gerência pró-ativa de redes de computadores, foi desenvolvido no LAR (Laboratório de Redes do CEFET Ceará), sendo assunto de duas dissertações do Mestrado em Ciência da Computação da UFC [8][16]. Sua arquitetura original era direcionada para a tecnologia ATM, fazendo uso de agentes inteligentes baseados em RNAs.

Com a evolução natural das redes de computadores, o ATM perdeu espaço para tecnologias emergentes, tais como, MPLS, GMPLS e DiffServ. Estas tecnologias simplificam o modelo de redes em quatro camadas e tentam aproveitar infra-estruturas legadas. A tendência é, cada vez mais, o fortalecimento das tecnologias que prometem inserir formas de garantir qualidade de serviço no protocolo IP. O desenvolvimento do RENATA 2 teve como objetivo levar em consideração a evolução proporcionada pelas tecnologias emergentes de redes de computadores. Uma das aplicações do ambiente RENATA 2 é atuar no processo de re-roteamento rápido do GMPLS.

Se melhorias forem aplicadas ao modo de re-roteamento do GMPLS, este poderá atuar com maior rapidez sobre o plano de controle de redes IP+DWDM. Agentes Inteligentes baseados em RNAs podem agilizar o processo de re-roteamento rápido de redes IP+GMPLS sobre DWDM. RNAs podem prever falhas em um *link* de uma rede óptica, permitindo assim, maior rapidez no processo de re-roteamento. O mecanismo inteligente destes agentes (RNAs) foi a principal contribuição desta dissertação. Tal mecanismo pode ser implementado por dois dos três tipos de RNAs demonstrado no Capítulo 6. As RNAs Tipo 1 utilizam variáveis relacionadas com a

camada física e são adequadas para a predição do próximo estado da BER desta camada. As RNAs Tipo 2 usam variáveis relacionadas com o tráfego da camada de rede, sendo apropriadas também para inferência do *status* instantâneo da BER da camada física. As RNAs Tipo 3 não apresentaram resultados satisfatórios, logo não devem ser consideradas para a construção dos agentes. Para aproximar do mundo real o banco de conhecimento das RNAs deste trabalho, foi desenvolvida a ferramenta GDP para melhorar o modo de tratamento de falhas do *ns*. Outra ferramenta desenvolvida foi o MSPD que ajusta os *logs* do *ns* ao JNNS, além de fazer a crítica dos resultados apresentados pelas RNAs em relação aos pontos de pico. Todo este processo foi construído dentro do escopo de atuação do ambiente RENATA 2.

A ferramenta GDP apresenta uma proposta simples, de rápida implementação e eficaz para agregar funcionalidades ao simulador *ns*. Gerar código de entrada para o *ns* não requer o entendimento do código fonte do mesmo. O *ns* foi implementado com a combinação das linguagens de programação Otcl e C++, aumentado assim a complexidade de entendimento do seu código fonte (muitas vezes pouco documentado). Desta forma, o GDP proporcionou, de maneira simples, que o tratamento imperativo de erros das simulações do *ns* ficasse mais próximo do mundo real. Esta mesma abordagem pode ser empregada para expandir ferramentas similares de código fonte proprietário.

Em discussão com John C. Kelliher, pesquisador especialista em redes ópticas do King's College, University of London, foi apontado um problema quanto ao uso da equação 3.7 (Seção 3.3) para modelar o comportamento de um receptor óptico real. De fato, não foi dado o mesmo tratamento de abstração para o transmissor (real) e para o receptor (ideal) implementados pelo GDP.

> *"The question of being able to detect link failure (in fact, link outage) is a central one. However, as the probability of error of an optical link can be calculated from the erfc integral using the expression PE = erfc (K/2), and as its form follows this expression I am not sure that measuring BER as prescribed will result in detection of link outage."* (John C. Kelliher).

O modelo matemático de um receptor óptico do mundo real faz uso de uma integral (*erfc,* encontrada na página 172 de [2]) que não apresenta solução geral, sendo resolvida apenas para valores literais. As variáveis que alimentam esta integral estão relacionadas com o ruído térmico e o *shot-noise.* A dificuldade em se obter valores para as variáveis relacionadas com o ruído, aliada à busca por uma solução de cálculo numérico para a referida integral, impossibilitariam o desenvolvimento do GDP em tempo hábil para a conclusão desta dissertação. Desta forma, preferiu-se trabalhar com as equações que modelam um receptor mais próximo de um receptor óptico ideal[24] e deixou-se a implementação do receptor real como trabalho futuro. Entretanto, tal fato não altera os resultados deste trabalho apesar de deixá-los mais longe do mundo real.

Kelliher também afirmou que a detecção preemptiva de falhas em *links* ópticos fornece consideráveis benefícios a transações de re-roteamento, classificando este tipo de trabalho como inédito.

> *"Interestingly, I am writing a transaction paper in a very similar area in which I am describing a command and control methodology which would work well with your agent based approach, although I am invoking the discipline of IKBS (intelligent knowledge based systems).*
>
> *I am sure that pre-emptive detection of link faults would provide considerable benefits to the re-routing transitions.*
>
> *I hope this helps, and more importantly is what you were looking for. It is the first time that I have reviewed a paper of this type. I hope you find my comments useful."* (John C. Kelliher).

Vale ressaltar que o uso de simulações viabilizou o desenvolvimento deste trabalho. Entretanto, as RNAs serão tão adequadas ao mundo real quão fielmente as

[24] Um receptor óptico ideal não sofre a interferência de ruídos, nem de corrente espontânea gerada sem excitação luminosa (*dark current*), dependendo apenas do limite quântico.

simulações refletirem o mesmo. No caso de se produzir RNAs para detecção de falhas em um único *link* de uma rede de computadores e pensando nos benefícios que as RNAs podem trazer, pode-se admitir a construção do banco de conhecimento a partir do próprio *link* real. Para isso, o investimento se restringe a um único ponto da rede e os resultados poderão ser aplicados a *links* semelhantes ao simulado. Entretanto, os custos são elevados mesmo para um único *link* e dificuldades, tais como, impossibilidade de uso do *link*, construção de aplicações geradoras de tráfego e coleta de dados terão que ser tratadas. O mesmo procedimento é quase inviável para simular uma rede completa. Neste caso, os simuladores são imprescindíveis.

A evolução natural do ambiente RENATA 2 deve ser a implementação da ferramenta Geradora de Agentes. Isso possibilitará soluções práticas de vários problemas do mundo real, dentre eles, o re-roteamento rápido em redes IP+GMPLS sobre DWDM e a alocação de caminhos em uma rede MPLS segundo o histórico dos enlaces e as necessidades de qualidade de serviço desejada para tal caminho. Além disso, o gerador de agentes deve ter uma interface de comunicação com ferramentas de gerência de domínio público e/ou comerciais. Assim, possibilitará que as mesmas se comuniquem com os agentes RENATA 2, aproveitando a infra-estrutura de *software* já existente e delimitando o escopo desse trabalho futuro.

Caso o simulador de RNAs de uso geral (JNNS) seja substituído por uma ferramenta específica, o uso do ambiente RENATA 2 não mais requererá que seus usuários (gerentes e/ou administradores de rede) sejam especialistas em redes neurais. Para isso, tal ferramenta em conjunto com o Gerador de Agentes deverá funcionar como uma caixa-preta, deixando totalmente transparente o desenvolvimento do agente. A responsabilidade do usuário final do RENATA será apenas a construção do banco de conhecimento segundo os dados disponíveis e relacionados com o problema em questão.

Enquanto motivador de novos trabalhos, o ambiente RENATA 2 merece ainda destaque especial por todas os trabalhos futuros já propostos e pelos que estão por vir. Com certeza, a UFC e o CEFET-CE, em especial o LAR, já têm questões científicas importantes para serem investigadas. Atualmente, o projeto RENATA já está sendo continuado por dois alunos do Curso de Tecnólogo em Telemática do CEFET-CE: um

bolsista de iniciação cientifica do CNPQ e uma aluna motivada por sua monografia de fim de curso. Enquanto se encontrou em andamento, esta dissertação gerou um trabalho de fim de curso [44] e o artigo [1], publicado na Revista Boletim Bimestral sobre Tecnologia de Redes da RNP. A empresa de desenvolvimento e pesquisa Instituto Atlântico, situada na cidade de Fortaleza, fechou uma parceria com a UFC e o CEFET-CE para submeter o RENATA 2 ao projeto GIGA, que se encontra sobre a coordenação do CPQD de Campinas, São Paulo.

Referências Bibliográficas

[1] GONÇALVES, C.H.R.; OLIVEIRA, A.M.; ANDRADE R.M.C; CASTRO, M.F.; Utilizando Redes Neurais para Predição de Falhas em Links de Redes Ópticas. In: Workshop RNP2, 4., 2003, Natal. **Revista Boletim Bimestral sobre Tecnologia de Redes,** Rio de Janeiro, v.7, n.3, 2003. 4p. Disponível em: <http://www.rnp.br/_arquivo /wrnp2/2003/rnpf01a.pdf>. Acesso em: 1 de ago de 2003.

[2] AGRAWAL, G.P. **Fiber-Optic Communication Systems**, 2.ed. New York: John Willey & Sons INC, 1997. 541p.

[3] BANERJEE, A.; DRAKE, J. et al. Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements. **IEEE Communications Magazine**, v.39, n.1, p.144-150, 2001.

[4] BANERJEE, A.; DRAKE, J. et al. Generalized Multiprotocol Label Switching: An Overview of Signaling Enhancements and Recovery Techniques. **IEEE Communications Magazine**, v.39, n.7, p.144-151, 2001.

[5] BIGUS, J.P. **Data Mining with Neural Networks**. New York: McGraw Hill Text, 1996. 220p.

[6] BIGUS, J.P.; BIGUS, J. **Constructing Intelligent Agents with Java – A Programmer´s Guide to Smarter Applications**. New York:Wiley Computer Publishing, 1998. 379p.

[7] DOVERSPIKE, R.; YATES, J. Challenges for MPLS in Optical Network Restoration. **IEEE Communications Magazine***,* v.39, n.2, p.89-96, 2001.

[8] CASTRO, M.F. **Redes Neurais na Estimativa da Capacidade Requerida em Comutadores ATM**. 1999. 138f. Dissertação (Mestrado em Ciência da Computação) - Departamento de Computação, Universidade Federal do Ceará, Fortaleza.

[9] HAYKIN, S. **Redes Neurais – Princípios e Prática**, 2.ed. Porto Alegre: Bookman, 2001. 900p.

[10] KARTALOPOULOS, S.V. **Fault Detectability in DWDM – Toward Higher Signal Quality & System Reliability**. New York: IEEE Press, 2001. 165p.

[11] LAWRENCE, Jeremy. Designing Multiprotocol Label Switching Networks. IEEE Communications Magazine, v.39, n.7, p.134-142, 2001.

[12] MARQUES, A.L.; SILVA, H.J.A. **Circuito de comando para laser semicondutor – modulação FSK directa,** Coimbra: Instituto de Telecomunicações. Disponível em <http://www.it.uc.pt/oc/ocpub/am97cp02.pdf>. Acesso em: 11 jan. 2003.

[13] MAS, Carmen, THIRAN, Patrick. An Efficient Algorithm for Locating Soft and Hard Failures in WDM Networks. **IEEE Journal**, v.18, n.10, p.1900-1911, 2000.

[14] MENDES, L.M.M.; SILVA, H.J.A. **Transmissor laser e receptor óptico sintonizado para um sistema SCM em 2GHz,** Coimbra: Instituto de Telecomunicações. Disponível em <http://www.it.uc.pt/oc/ocpub/lm97cp01.pdf>. Acesso em: 11 de jan. de 2003.

[15] N. GOLMIE, F.; MOUVEAUX, L.; HESTER, Y; et al. **The NIST ATM/HFC Network Simulator: Operation and Programming Guide – Version 4.0**. **NIST** – National Institute of Standards and Technology – U.S. Departament of Comerce, 1998.

[16] NASCIMENTO, A.S. **Desenvolvendo Agentes Inteligentes para a Gerência Pró-Ativa de Redes ATM**. 1999. 166f. Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação, Universidade Federal do Ceará, Fortaleza.

[17] NAUGHTON, S.; SOMERS, F.; Asynchronous Transfer Mode (ATM) Source Frame Prediction Using Neural Networks**.** In: INNC - IRISH NEURAL NETWORKS CONFERENCE. 1995. 13p.

[18] SOTO, C.P. **Redes Neurais Temporais para o tratamento de Sistemas Variantes no Tempo.** 1999. 112f. Dissertação (Mestrado em Engenharia Elétrica) – PUC/RJ, Rio de Janeiro.

[19] STEVENS, W.R**. TCP/IP Ilustrated - The Protocols.** Berkeley: *Addison Wesley*, 1997. 575p.

[20] TAFNER, M.A.; RODIGUES I.W. **Redes Neurais Artificiais, Introdução e Princípios de Neurocomputação.** Blumenau: EKO, 1996. 199p.

[21] The MPLS Resource Center, Disponível em <www.mplsrc.com/mplsfac>. Acesso em: 12 de nov.2001.

[22] University of Stuttgart – Institute for Parallel and Distributed High Performance Systems (IPVR). **SNNS (The Stuttgart Neural Network Simulator)**. V.4.0, 1995.

[23] BRISA (Sociedade Brasileira de Sistemas Abertos). **Gerenciamento de Redes – Uma Abordagem de Sistemas Abertos.** São Paulo: Makron Books, 1993. 363p.

[24] STALLINGS, W. **High-speed Networks – TCP/IP and ATM design principles.** New Jersey: Prentice-Hall, 1998. 576p.

[25] FLESCH, F.H.; NACAMURA, L.J. AGEDP: Uma Arquitetura de Gerenciamento Dinâmico de Redes Baseado em Políticas. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 2002, Itajaí. Anais... http://www.cbcomp.univali.br/anais/2002/rcomputadores.htm. Acesso em 02 de mar. de 2003.

[26] THOMPSON, J.P., Web-Based Enterprise Management Architecture. **IEE Communication Magazine,** v.36, p.80-86, mar. de 1998.

[27] BÉNECH, D.; MONROZIER, F.J.; RIVIÈRE, A.I. *Supervision of the CORBA Environment with SUMO: a WBEM/CIM-Based Management Framework.* In: DISTRIBUTED OBJECTS AND APPLICATIONS, 2000, Antwerp, Belgium. **Proceedings…** Antwerp, Belgium: IEEE, 2000. 359p. p. 241-250.

[28] OLIVEIRA, M.; FRANKLIN, M.; NASCIMENTO, A.; VIDAL, M. **Introdução à Gerência de Redes ATM.** Fortaleza: Publicações CEFET-CE, 1998. 168p.

[29] JMX (Java Management Extensions)**. Disponível em <http://java.sun.com/products/JavaManagement/>.** Acesso em: 07 de mar. de 2003

[30] **MNS (MPLS Network Simulator)**. **Disponível em** <http://flower.ce.cnu.ac.kr/~fog1/mns/>. Acesso em: 14 de mar de 2003.

[31] PARC (Palo Alto Research Center), LBNL (Lawrence Berkeley National Laboratory) e UCB (University of California). ***ns* (*Network Simulator*)**.

[32] ***Nam*: Network Animator. Disponível em** < http://www.isi.edu/nsnam/nam/>. Acesso em: 14 de mar de 2003.

[33] **VINT - Virtual InterNetwork Testbed**. **Disponível em** <http://www.isi.edu/nsnam/vint/index.html>. Acesso em: 14 de mar de 2003.

[34] **The ns Manual (formerly ns Notes and Documentation)** – Uc Berkeley, LBLm USC/ISI, and Xerox PARC Disponível em <http://www.isi.edu/nsnam/ns/ns-documentation.html>. Acesso em: 1 nov. de 2002.

[35] **Use of Altamar Optical Amplifiers with DWDM - APPLICATION NOTE**. Disponível em: <http://www.altamar.com/products/appnotes/DWDM.pdf>. Acesso em 16 dez. de 2002.

[36] **Scientific Atlanta Muliplexer/Demultiplexer for DWDM Systems - USER MANUAL.** Disponível em <http://tvccatalog.tvcinc.com/Scientific_Atlanta/ IO106069dwdm.pdf>. Acesso em 16 dez. de 2002.

[37] **1640 OADM Long-Haul and Ultra Long-Haul DWDM Optical Add/Drop Multiplexer.** Disponível em <http://www.alcatel.com/products/productsummary. jhtml?_DARGS=%2Fcommon%2Fsolutionselector%2Finclude%2Fprodsinsoloutput.j html.1_A&_DAV=%2Fx%2Fopgproduct%2FAlcatel_1640_OADM.jhtml>. Acesso em 16 dez. de 2002.

[38] CHEIKHROUHOU, M.; CONTI, P.; LABEROULLE, J. Intelligent Agents in Network Management, a State-of-Art. **Networking and Information Systems Journal.** v.1, n.1, p.9-38, 1998. In: NASCIMENTO, A.S. **Desenvolvendo Agentes Inteligentes para a Gerência Pró-Ativa de Redes ATM**. 1999. 166f. Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação, Universidade Federal do Ceará, Fortaleza.

[39] SOARES, L.F.; LEMOS, G.; COLCHER, S. **Redes de Computadores das LANS, MANS e WANS as Redes ATM.** Rio de Janeiro: Campus, 1995. 728p.

[40] SILVEIRA M. S.; KOVACH S.; CARVALHO, T. C. M. B.; RUGGIERO, W. Arquitetura, Topologia e Roteamento em Redes Ópticas. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 21., 2003, Natal. **Minicursos**… Natal: Soc. Brasileira de Computação, 2003. 363p. p.279-318.

[41] WONG, P.K.C., RYAN, H.M., TINDLE, J.; Power System Fault Prediction Using Artificial Neural Networks, In: PROC. OF INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING. 1996, Hong Kong. 10p.

[42] ABRAHAM, A.; A Soft Computing Approach for Fault Prediction of Electronic Systems, In: INTERNATIONAL CONFERENCE ON COMPUTERS IN INDUSTRY (ICCI), 2., 2000, Bahrain. **Anais…** The Bahrain Society of Engineers, 2000, p.83-91.

[43] **Internet 2**. Disponível em < http://www.pucsp.br/internet2/>. Acesso em 15 ago. de 2002.

[44] BARBOSA, C.H.F. **MPLS – Uma Análise de Recursos Através de Simulações**. 2002. 57f. Monografia de Fim de Curso (Tecnólogo em Telemática) – Departamento de Telemática, Centro Federal de Educação Tecnológica do Ceará, Fortaleza.

[45] BOX, G.; JENKINS, G. **Time series analysis, forecasting and control**. San Francisco: Holden-Day, 1970.

[46] KANTZ, H.; SCHREIBER, T. **Nonlinear Time Series Analysis**. Cambridge: University Press, 1999.

[47] AGUIRRE, L. A. **Introdução à Identificação de Sistemas**. Belo Horizonte: UFMG, 2000.

[48] BARRETO, G. A.; ANDRADE, M.G. Robust Bayesian approach for AR(p) models applied to streamflow forecasting. **Journal of Applied Statistical Science**, 2003.

[49] BAZARAA, M. S.; SHERALI, H.D.; SHETTY, C. M. **Nonlinear Programming Theory and Algorithms**. Wiley, 2ed. 1993.

[50] HAYKIN, S. **Neural Networks: A Comprehensive Foudation**. Englewood Cliffs, NJ, Macmillan Publishing Compay, 1994.

[51] STERRITT, R.; BUSTARD, D. W. Fusing Hard and Soft Computing for Fault Management in Telecommunications Systems. *IEE Transactions on Systems*, v.32, n.2, p.92-98, 2002

Bibliografia

DEITEL, H.M.; DEITEL P.J. **Java Como Programar.** 4.ed. Porto Alegre: Bookman, 2003. 1386p.

DZIONG, Z. **ATM Network Resource Management**. New York: McGraw-Hill, 1997. 315p.

FRANCESCHI, A.S.M.; BARRETO, J.M. Desenvolvendo Agentes de Software para Gerência de Redes Utilizando Técnicas de Inteligência Artificial. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 2., 2002, Itajaí. Disponível em <http://www.cbcomp.univali.br/anais/2002/agentes.htm> Acesso em 20 de jan. de 2003.

RUSSEL, S.; NORTVING, P. **Artificial Intelligence - A Modern Approach.** New Jersey: Prentice Hall, 1995. 932p.

GILBERT, S.; M$_C$CARTY, B.; **Object-Oriented Design in Java.** Corte Madera: Waite Group Press, 1998. 731p.

Anexo A

Redes Neurais Artificiais

Embora Redes Neurais Artificiais não estejam ligadas diretamente com o gerenciamento das redes de computadores, estas estão intimamente relacionadas aos agentes inteligentes dos Ambientes RENATA 1 e 2. Desta forma, este anexo é muito importante para o entendimento desta área da Inteligência Artificial e fornece subsídios para o aprofundamento de conhecimentos sobre este tipo de máquina inteligente que dá suporte a tecnologia de agentes.

Uma rede neural artificial é uma estrutura composta de unidades processadoras simples, distribuídas e paralelas, que tem o propósito de armazenar conhecimento por meio de mecanismos empíricos e torná-lo disponível para o uso. Ela se espelha no cérebro humano em dois aspectos [9] [20] :

1. O conhecimento é adquirido pela rede a partir do meio ambiente através de mecanismos de aprendizagem;
2. A "*força*" das conexões entre as unidade processadoras são minimizadas ou maximizadas de forma a armazenar melhor o conhecimento adquirido.

Apesar do caráter matemático dominante, as redes neurais artificiais recebem este nome por terem sido inspiradas nas redes neurais biológicas, sendo assim, analogamente correspondentes. Com o intuito de facilitar a leitura, será usado simplesmente o termo redes neurais para designar a estrutura matemática em questão.

O termo neurônio refere-se a cada unidade processadora da rede neural, estes são implementados através de uma função matemática. O procedimento utilizado para realizar a aprendizagem da rede neural é feito por meio de um algoritmo de aprendizagem; cuja função é modificar a magnitude da "*força*" de ligação entre os neurônios, ou seja, modificar os pesos sinápticos das conexões de uma forma ordenada para alcançar um objetivo desejado (observe a Figura A.1). As redes neurais são também mencionadas na literatura como *neurocomputadores, redes conexionistas* e *processadores paralelamente distribuídos,* entre outras. [9] **.**

A.1 Capacidades e Propriedades

O poder computacional de uma rede neural advém de sua estrutura maciçamente paralela e distribuída e de sua capacidade de aprendizagem e generalização. O conceito de generalização deve ser visto como a habilidade da rede em fornecer valores adequados para entradas não disponíveis no processo de aprendizagem. As características de aprendizagem e generalização podem ser consideradas como o diferencial fornecido pelas redes neurais em comparação com outras tecnologias, tais como, sistemas especialistas e inteligência artificial simbólica. Estas duas capacidades de processamento de informação tornam possível para as redes neurais resolverem, com grau de aproximação satisfatório, subproblemas específicos de problemas complexos (de grande escala) que são atualmente intratáveis em sua forma completa. Além de aprendizagem e generalização, as redes neurais apresentam as seguintes capacidades:

1. **Aproximação de funções:** uma forma comum de aprendizado é o paradigma supervisionado onde é fornecida a rede neural uma entrada com a respectiva saída esperada, então os pesos sinápticos são ajustados de forma a produzir a saída ou resposta esperada. Este processo deve se repetir até que a rede forneça as saídas esperadas (i.e.,convergência da rede). Uma aplicação desta propriedade é a classificação de padrões, da qual o objetivo é encontrar

arbitrariamente a fronteira de disjunção entre as várias categorias ou classes preestabelecidas de objetos ou eventos.

2. **Adaptabilidade:** uma rede neural que opere em um ambiente não-estacionário pode ser projetada para retreinamento. Assim, pequenas variações nos pesos sinápticos em tempo real podem ser realizadas com o intuito de se adequar a pequenas variações no ambiente. Apesar da adaptabilidade ser um fator de robustez, ela deve ser feita de forma criteriosa em relação ao tempo. Um sistema adaptativo com constantes de tempo pequenas pode se modificar rapidamente e responder a perturbações espúrias, causando uma drástica degradação do sistema. Para manter a robustez do sistema, as constantes de tempo responsáveis pelo período de re-treinamento da rede neural devem ser suficientemente grandes de forma que perturbações espúrias não interfiram no bom funcionamento do sistema e sejam suficientemente pequenas para capturar mudanças no ambiente o mais rápido possível.

3. **Resposta a evidências**: no contexto de classificação de padrões, uma rede neural pode ser projetada não só para responder sobre a disjunção de objetos, mas também sobre a confiança ou crença na decisão tomada. Esta característica deverá ser aplicada quando o problema em questão tratar de padrões ambíguos, dando assim maior confiabilidade ao processo de classificação.

4. **Tolerância a falhas**: Caso um neurônio venha a falhar, a própria estrutura descentralizada das redes neurais favorece a continuidade do bom funcionamento do sistema. Desta forma, o dano causado a rede deve ser relativamente amplo para que a resposta final da rede seja seriamente comprometida. Em princípio, a degradação de uma rede neural ocorre suavemente e situações anormais só são causadas por sérios danos à mesma.

Além das capacidades acima citadas, as redes neurais possuem propriedades que garantem que as mesmas possam ser aplicadas em diversos tipos de problemas. Abaixo há uma lista destas propriedades.

1. **Não-linearidade:** o grande benefício oferecido por esta propriedade é o fato das redes neurais poderem tratar entradas não-lineares como, por exemplo, um sinal de voz. É importante frisar que esta característica não impede as redes neurais de tratar grandezas lineares.

2. **Implementação em VLSI:** caso um problema exija tratamento muito rápido, há possibilidade de se implementar redes neurais diretamente em *hardware*. A tecnologia VLSI (*Very Large Scale Integration)* fornece um meio de capturar comportamentos complexos de forma hierárquica, o que vem ao encontro perfeito da tecnologia das redes neurais.

3. **Analogia Neurobiológica:** As redes neurais artificiais são motivadas pela analogia com o cérebro, o qual mostra que o processamento paralelo tolerante a falhas é não somente possível fisicamente, mas também poderoso. Os neurobiólogos olham para as redes neurais artificiais como uma ferramenta de interpretação e pesquisa para fenômenos neurobiológicos. Em contrapartida, os engenheiros e cientistas da computação procuram na neurobiologia novas idéias para resolver problemas mais complexos dos que os resolvidos pelos métodos lineares convencionais.

A.2 Neurônios - Unidades de Processamento

O neurônio é a unidade básica de processamento de informação de uma rede neural. A Figura A.1 mostra um neurônio com a identificação de seus elementos básicos.

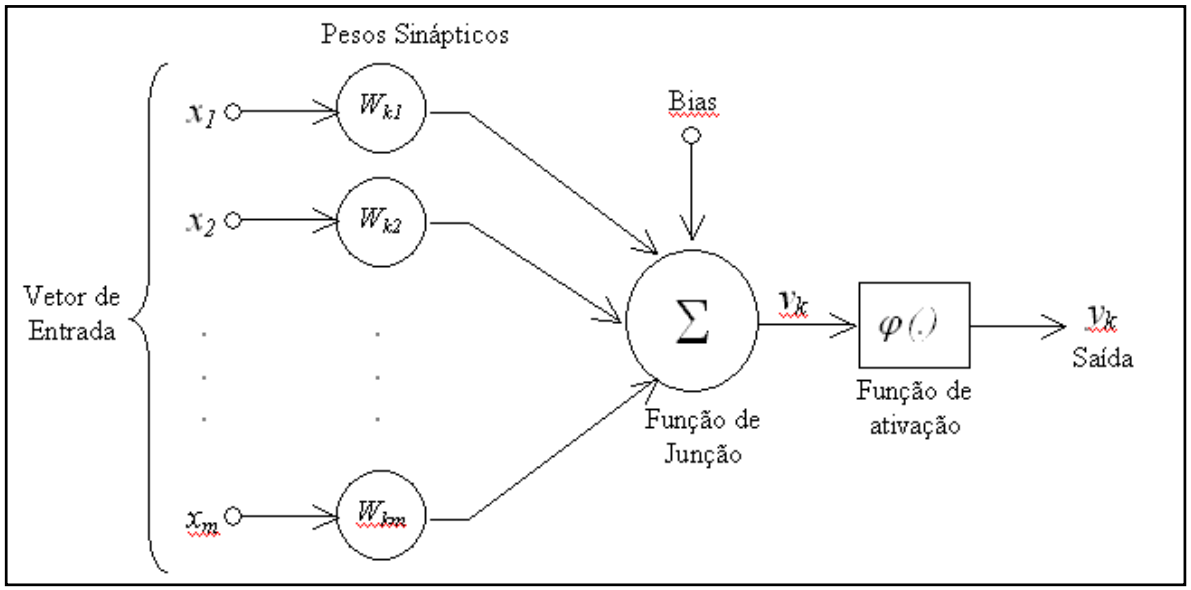

Figura A.1 – Modelo básico para um neurônio[25].

1. **Conjunto de Sinapses** (neurônios de entrada ou elos de conexão)**:** Cada um dos neurônios de entrada se caracteriza por seu peso ou força própria. Um sinal x_j na entrada j conectado a um neurônio k é multiplicado pelo peso sináptico w_{kj}, onde o índice k refere-se ao neurônio em questão e o índice j refere-se ao terminal de entrada.
2. **Função de Junção**: serve para compor de forma linear os sinais de entrada já ponderados pelos seus respectivos pesos sinápticos. A função de junção também é conhecida como combinador linear ou simplesmente somador.
3. **Função de ativação**: seu uso se faz necessário para restringir a amplitude de saída de um neurônio. Por convenção, as saídas dos neurônios devem estar

[25] Figura retirada da página 36 de [9]

restritas ao intervalo fechado [0,1] ou [-1,1], há casos ainda de saída binária 0 ou 1. Assim, a função de ativação ou função restritiva tem com objetivo adequar a saída do neurônio a um destes intervalos. Um fator que pode interferir na função de ativação é o *bias*, o qual pode ser identificado na Figura A.1, sendo b_k sua representação matemática. Este fator tem o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo se ele é positivo ou negativo, respectivamente. Desta forma, o *bias* interfere diretamente na sensibilidade do neurônio em análise às entradas.

Em termos matemáticos, um neurônio k é descrito segundo as equações A.1 e A.2.

$$u_k = \sum_{j=1}^{m} w_{kj} x_{j,} \tag{A.1}$$

$$y_k = \varphi\,(u_k + b_k) = \varphi\,(v_k), \tag{A.2}$$

tal que $x_1, x_2, x_3,\ldots, x_m$, são sinais de entrada; $w_{k1}, w_{k2}, w_{k3},\ldots, w_{km}$, são os pesos sinápticos do neurônio k; u_k é a saída da função de junção; b_k é o bias; $\varphi(\,.\,)$ é a função de ativação e por fim y_k é o sinal de saída do neurônio. A equação A.3 define o termo potencial de ativação v_k, que trata da entrada da função de ativação.

$$v_k = u_k + b_k \tag{A.3}$$

A.3 Tipos de Função de Ativação

A função de ativação definida pelo termo $\varphi(\,.\,)$ é responsável pela saída do neurônio conforme convenção já citada no item anterior. Há três tipos básicos de funções de ativação:

1. **Função Limiar:** definida pela equação A.4 e seu gráfico é exposto na Figura A.2.a.

$$y_k = \begin{cases} 1, \text{ se } v_k \geq 0 \\ \\ 0, \text{ se } v_k < 0 \ , \end{cases} \tag{A.4}$$

Onde v_k é o potencial de ativação do neurônio definido na equação A.3.

2. **Função Linear por Partes (*piecewise*):** esta função é definida pela equação A.5 e seu gráfico está exposto na Figura A.2.b.

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{A.5}$$

3. **Função Sigmóide.** Este é o modelo mais comum de função de ativação usado em projetos de redes neurais. O gráfico desta função, exposto na Figura A.2.c forma um s, o que fornece um comportamento de balanceamento adequado entre linear e não-linear, sendo ainda estritamente crescente. Um exemplo de função sigmoidal é a função logística, definida pela equação A.6.

$$\varphi(v) = \frac{1}{1+\exp(-av)} \ , \tag{A.6}$$

na qual *a* representa o parâmetro de inclinação (*slope*) da função sigmóide. Variando-se *a* se obtém diferentes inclinações desta função, com ilustrado na Figura A.2.c.

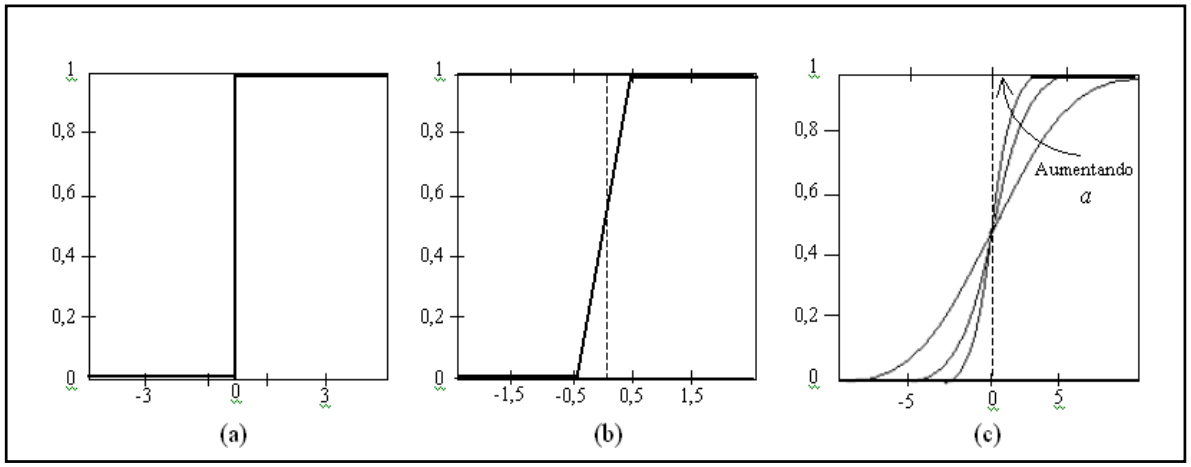

Figura A.2 – (a) Função de limiar. (b) Função linear por partes. (c) Função sigmóide.

A.4 Arquitetura de Redes Neurais

A forma como os neurônios estão organizados e interconectados define a arquitetura de uma rede neural e está intimamente ligada com o algoritmo de aprendizagem usado para treinar a rede. Nesta seção serão consideradas apenas as formas de conexão entre os neurônios, ou seja, a estrutura ou topologia da rede. A maneira como as redes são treinadas será tratada na Seção A.5. Em geral pode-se citar três classes de arquitetura de redes neurais fundamentalmente diferentes.

A.4.1 Redes *Feedforward* de Uma Camada

Na forma mais simples de uma rede em camadas, temos uma camada de entrada de nós fontes que se comunica diretamente através de sinapse com a camada de saída em sentido único, ou seja, esta rede é sempre alimentada adiante, por isso denominada de rede *feedforward* ou acíclica. Uma rede como a ilustrada na

Figura A.3 é considerada de uma camada, uma vez que os neurônios da camada de entrada apenas fazem o repasse do sinal e não desempenham nenhum processamento. Então o termo unicamada se refere à camada de saída, a qual é a única responsável pelo tratamento do sinal de entrada.

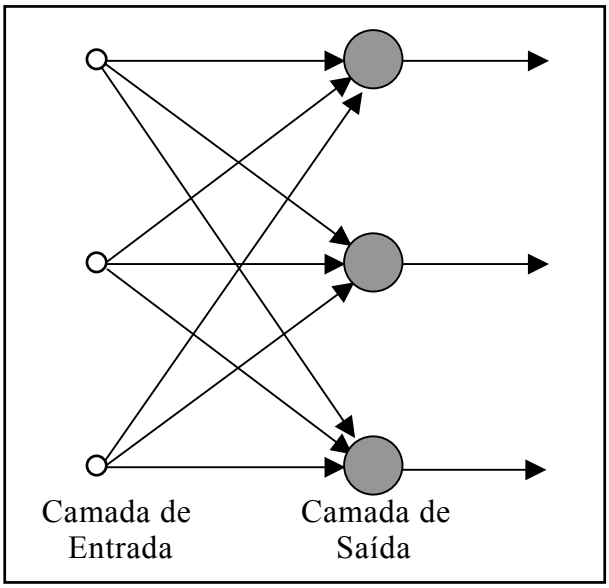

Figura A.3 – Rede neural *feedforward* unicamada.

A.4.2 Redes *Feedforward* Multicamada

Redes neurais *feedforward* multicamadas se distinguem das redes com camada única por apresentarem uma ou mais camadas intermediárias entre a entrada e a saída da rede neural. Estas camadas são denominadas camadas ocultas, cujos nós computacionais são chamados correspondentemente de neurônios ocultos. A razão para a presença de neurônios ocultos é proporcionar às redes neurais a capacidade de extrair estatísticas de ordem mais elevada sobre os dados de entrada. Este tipo de arquitetura é particularmente valioso quando se tem uma camada de entrada muito grande.

A Figura A.4 mostra uma rede *feedforward* multicamada 5-3-2, visto que possui 5 (cinco) neurônios da primeira camada (entrada), 3 (três) na segunda camada (oculta) e 2 (dois) na terceira (saída). Uma rede neural é considerada totalmente ligada ou conectada quando todo neurônio de uma camada qualquer estiver conectado a todos os neurônios da camada subseqüente, caso contrário, é denominada parcialmente ligada.

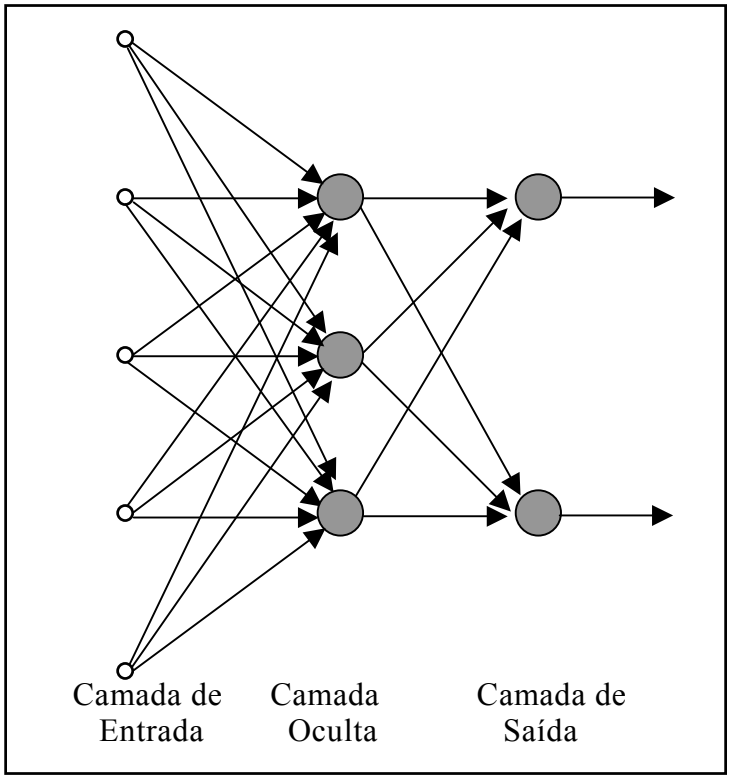

Figura A.4 – Rede neural 5-3-2, *feedforward m*ulticamada totalmente ligada.

A.4.3 Redes Recorrentes

Uma Rede Neural Recorrente difere das *feedforward* pelo fato de apresentar pelo menos um laço de retro-alimentação. Os elementos de atraso unitário, representados por z^{-1}, são responsáveis por fornecer saídas já computadas em iterações passadas à rede neural em questão, o que explica o termo retro-alimentação. O atraso no fornecimento de entradas à rede neural introduz memória na rede, proporcionando aos neurônios valores de entrada atuais e valores temporalmente anteriores a eles. As redes neurais recorrentes geralmente apresentam um comportamento dinâmico não-linear, o que as torna mecanismos úteis na predição de valores inerentemente temporais (séries temporais). A Figura A.5 traz a representação de uma rede neural recorrente com apenas uma camada, onde a saída de cada neurônio é apresentada aos demais neurônios da rede e não se verifica traços de auto-alimentação. Na prática, redes recorrentes são utilizadas principalmente para *memórias associativas* e para

aproximar *mapeamento entrada-saída* [18] . Neste tipo de rede também pode ocorrer casos de auto-alimentação e multicamadas.

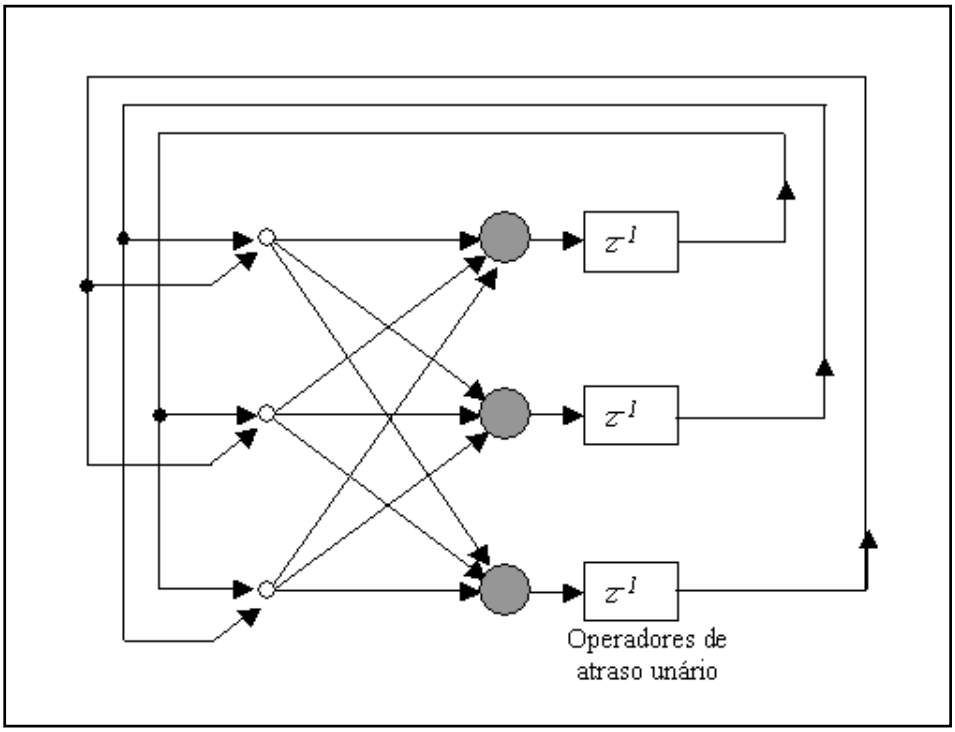

Figura A.5 – Exemplo de rede neural recorrente.

A.5 Processo de Aprendizagem

A propriedade primordial das redes neurais é aprender conforme o ambiente onde estão inseridas e assim melhorar seu desempenho. Para que uma rede neural possa aprender, se faz necessário apresentar um conjunto de exemplos a mesma de forma seqüencial e interativa. Este processo é chamado de treinamento de redes neurais. Em cada iteração, os pesos sinápticos e o *bias* são ajustados. Este processo deve se repetir até o objetivo da rede ser alcançado. Teoricamente, uma rede neural torna-se mais instruída a cada passo do processo de aprendizagem. Segundo [9], aprendizagem no contexto de redes neurais pode ser definida como um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. O tipo de aprendizagem é determinado pela forma que a modificação dos parâmetros ocorre. Então se pode

identificar três eventos principais do processo de aprendizagem que ocorrem de forma seqüencial:

1. A rede neural recebe os padrões de aprendizagem;
2. Segundo os padrões de teste, os pesos sinápticos da rede neural são modificados e
3. A rede neural responde de uma maneira nova ao problema para qual se destina.

Há dois pontos chaves que devem ser tratados durante o processo de aprendizagem de uma rede neural. O primeiro é o algoritmo de aprendizagem, o qual deve definir regras para a solução do processo de aprendizagem. O segundo é o paradigma de aprendizagem, o qual trata do modelo de ambiente no qual a rede deverá operar. Existe uma variedade considerável destes algoritmos, cada uma com suas peculiaridades, tal que a diferença básica entre eles se dá na maneira como os pesos sinápticos são tratados e no paradigma de aprendizado escolhido. A Tabela A.1 enumera um conjunto de algoritmos de aprendizagem com suas respectivas características.

Quanto aos paradigmas de aprendizagem há dois tipos básicos: o supervisionado, também conhecido como processo de aprendizado com auxílio de um professor e o paradigma não-supervisionado ou auto-supervisionado. No paradigma supervisionado um "*professor*" indica à rede neural qual o resultado esperado para uma entrada específica da mesma. O valor esperado pelo professor é comparado com o valor fornecido pela rede neural, desta forma se consegue um erro estimado. O erro deve ser usado pela rede como parâmetro de correção dos pesos sinápticos. Neste paradigma, o professor é representado por um conjunto de vetores de entrada com seus respectivos valores de saída desejados. Estas informações compõem o padrão de treinamento ou exemplos usados no processo de aprendizado. Este processo interativo deve prosseguir até que a taxa de acerto da rede em treinamento atinja um patamar satisfatório. Esta forma de aprendizado é bem conhecida e tem demonstrado excelentes resultados em aplicações reais [20] .

Algoritmo de Aprendizagem	Paradigma de Treinamento	Arquitetura	Aplicação Principal
Adaptative Resonance Theory	Não Supervisionado	Recorrente	Agrupamento
Kohonen Feature Maps	Não Supervisionado	*Feedforward*	Agrupamento
ARTMA*P*	Supervisionado	Recorrente	Classificação
Recurrent Backpropagation	Supervisionado	Recorrente	Séries Temporais
Radial Basis Function Networks	Supervisionado	*Feedforward*	Classificação, Séries Temporais
Backpropagation	Supervisionado	*Feedforward*	Classificação

Tabela A.1 - Algoritmos de aprendizagem e suas características.

No paradigma não-supervisionado não há necessidade de saídas nos padrões de treinamento da rede neural. Isso implica que não há necessidade de um professor que auxilie o processo de treinamento. A rede trabalha os valores de entrada e os organiza de forma classificatória com base em suas propriedades estatísticas. Hora através da competição, hora através da cooperação entre os neurônios, o paradigma não-supervisionado atinge seus objetivos de classificação. Muitos pesquisadores têm utilizado este tipo de rede como detector de características, dada a sua capacidade de aprender a discriminar estímulos ocorridos em partes espacialmente diferentes [20] .

A.6 Fases de um Projeto de uma Rede Neural

Para que uma rede neural possa cumprir seu propósito, fatores como disponibilidade e características dos dados, bem como topologia e algoritmo de treinamento devem ser criteriosamente analisados. Há dois momentos distintos da operação de uma rede neural: a fase de treinamento e a fase de uso. O aprendizado é um processo de ajuste dos pesos das conexões em resposta ao erro gerado pela rede.

Ou seja, a rede é modificada de acordo com a necessidade de aprender segundo os padrões sugeridos. No processo de utilização, a rede apenas fornece um valor segundo os dados de entrada e nenhum ajuste nos pesos sinápticos é efetuado. A Figura A.6 mostra o procedimento a ser adotado para o desenvolvimento de uma rede neural.

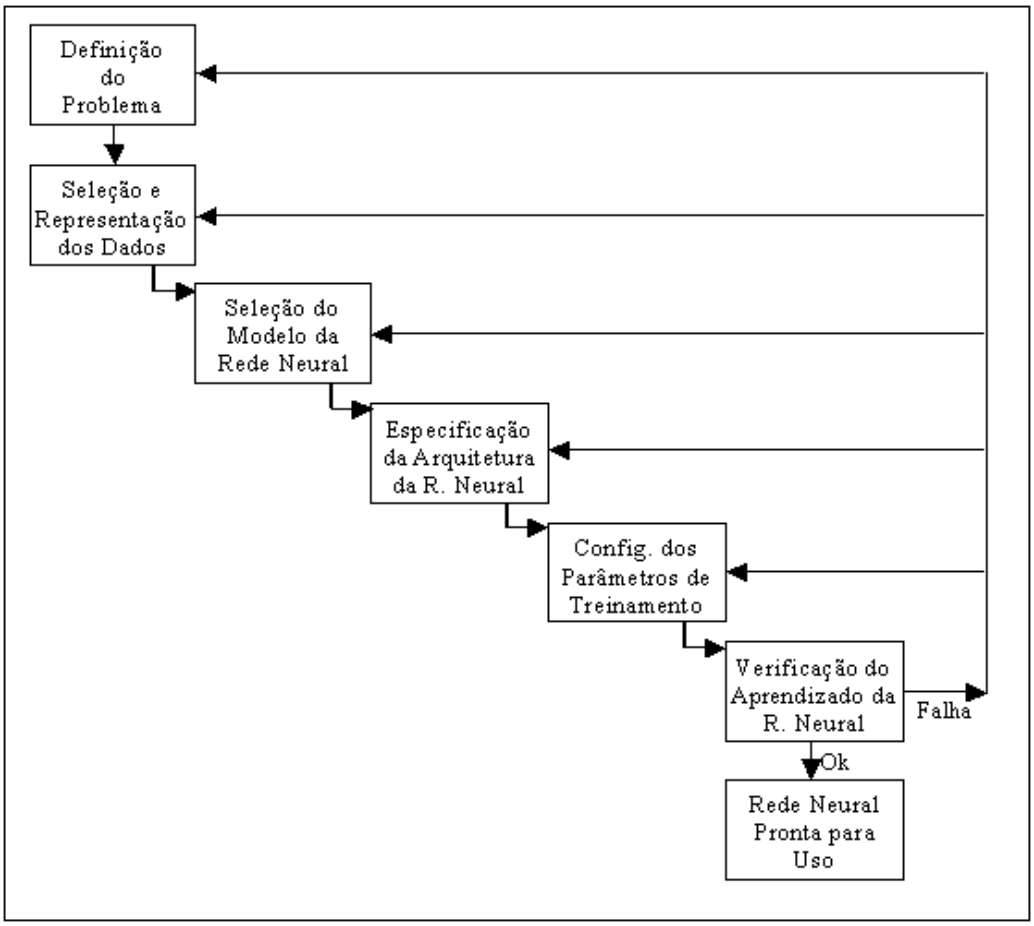

Figura A.6 – Fases do desenvolvimento de uma rede neural.

A.6.1 Definição do Problema

Antes de tentar resolver um problema através do paradigma de redes neurais é aconselhável que se faça uma análise para verificar se domínio do problema pode ser tratado por uma tecnologia mais simples. Geralmente, as redes neurais são aplicáveis quando não há um algoritmo com custo computacional (complexidade) tratável pelos computadores atuais, entretanto a solução proposta pelas redes neurais se restringe a um domínio restrito do algoritmo (subproblema local). Alguns exemplos destes problemas são: reconhecimento e classificação de padrões, predição de séries

temporais e aproximação de funções complexas. Outro aspecto importante é o cuidado na seleção das variáveis relevantes ao problema em questão. Esta seleção envolve, além da identificação das variáveis intimamente relacionadas ao problema, a exclusão das variáveis não-confiáveis ao processo, ou cujo uso seja impraticável por razões técnicas ou econômicas.

A.6.2 Seleção e Representação de Dados

Há necessidade do tratamento de dados antes que estes possam fazer parte da rede neural, alguns parâmetros identificados na definição do problema podem ser combinados em um só ou eliminados em caso de redundância. Visto que qualquer valor não pode ser apresentado a rede neural, técnicas de escalonamento, codificação e normalização dos dados podem ser requeridas nesta fase. Geralmente as redes neurais só aceitam como entrada valores reais no intervalo fechado [0,1] ou [-1,1], ou ainda valores binários 0 ou 1.

A.6.3 Seleção do Modelo da Rede Neural

A seleção do modelo da rede neural está intimamente ligada ao tipo de problema a ser resolvido. Por exemplo, se os dados têm forte relacionamento temporal, uma melhor escolha para o modelo da rede seria do tipo *Recurrent Backpropagation,* em vez do modelo *Backpropagation* padrão. A Tabela A.1 também pode ser usada para escolher o modelo da rede a ser usada conforme a aplicação requerida.

A.6.4 Especificação da Arquitetura da Rede Neural

Para o desenho da rede é preciso especificar os valores e condições inicias para o paradigma selecionado. Para os neurônios, deve-se definir o tipo de entrada e a integração dessas entradas. Em nível de rede, deve-se pensar em número de camadas, número e tipos de neurônios de entrada, de saída e intermediários, além do tipo de conectividade. O número de camadas intermediárias deve ser avaliado com cuidado, pois seu excesso pode ocasionar a degradação do poder de generalização da rede e o

aumento no tempo de treinamento, fazendo com que a rede apenas memorize os exemplos fornecidos sem extrair nenhum relacionamento dos dados.

A.6.5 Configuração dos Parâmetros de Treinamento

Nesta fase são definidos os parâmetros de aprendizado, que entre outros fatores, determinam a velocidade do treinamento e o poder de generalização da rede neural. Função de ativação e taxa de aprendizado são os parâmetros de treinamento mais gerais, ou seja, estão presentes em quase todos os tipos de redes neurais. A taxa de aprendizado é um parâmetro geral que determina o grau de atualização realizada nos pesos sinápticos, visando a saída alvo. A taxa de aprendizado é diretamente proporcional a velocidade de treinamento, mas pode acarretar na variação demasiada dos pesos sinápticos e assim causar uma maior imprecisão nos resultados gerados. Há também diversos parâmetros de treinamento específicos a cada modelo de rede neural, como é o caso do fator de momento e tolerância do erro do modelo *Backpropagation*.

A.6.6 Verificação do Aprendizado da Rede

Nesta fase, exemplos inéditos e já testados devem ser mostrados à rede neural para verificar se ela realmente obteve um aprendizado satisfatório. Se a rede atingir um valor aceitável de erro, ela está treinada e pronta para o uso. Se a rede não obtiver uma performance condizente, mostrando-se inadequada. Neste caso, as fases de desenvolvimento anteriores deverão ser reavaliadas com o intuito de identificar a origem do problema, não sendo descartada nem mesmo a fase de definição do mesmo.

A.6.7 Uso da Rede

A rede neural estará pronta para o uso, caso passe no teste de verificação do aprendizado da rede citado acima. Neste momento, os pesos sinápticos guardam o "*conhecimento"* da rede e não são mais atualizados, Assim, conforme seu

treinamento, a rede deve classificar, agrupar ou prever situações que lhe são fornecidas através de suas entradas. É possível que o ambiente mude e o problema resolvido pela rede neural sofra alterações, logo a rede deve ser re-treinada para se adequar às novas especificidades do problema. Há redes que aceitam re-treinamento *on-line* (paradigma não supervisionado), mas redes do paradigma supervisionado necessitam de um "professor".

Anexo B

Base das Simulações

O *script* do *ns* (arquivo *.tcl) com seus devidos comentários são expostos abaixo. Tal *script* é ponto de partida para que o usuário comece o processo de simulação necessário ao desenvolvimento dos agentes inteligentes RENATA 2. A linguagem Otcl é a responsável pela interface homem máquina do *ns*. Os comentários feitos no *script* em questão são elucidativos, tanto para o entendimento das simulações feitas nesta dissertação, como para o entendimento dos comandos da linguagem Otcl e por conseguinte do *ns*.

```
# ******** Definições Iniciais ****************************************
d
# Cria o objeto simulador.
set ns [new Simulator]

# Abre o arquivo de log.
# A tag abaixo #MSPDflg) marca o local do identificador do arquivo de log
# para que o MSPD possa renomeá-lo e montar os vários logs finais.
#MSPDflg
set logf [open logf.log w]

# Cada set abre um arquivo para gerar gráficos com o xgraph (opcional).
set grfBanda        [open Banda.tr w]
set grfPktPerdidos  [open PktPerdidos.tr w]
set grfPktRecebidos [open PktRecebidos.tr w]
set grfBER          [open BER.tr w]


# Abre o arquivo para armazenar a animação do nam (opcional).
set nf [open simula_Teste.nam w]
ns namtrace-all $nf
```

```
# ******** Procedimentos ******************************************

# Procedimento de finalização da simulação.
proc finish {} {

      $ns flush-trace
      close $logf
      close $nf
      close $grfBanda
      close $grfPktPerdidos
      close $grfPktRecebidos
      close $grfBER


      exec nam simula_Teste.nam &

      # Comando para executar gráficos (opcional).
      exec xgraph -m Banda.tr PktPerdidos.tr PktRecebidos.tr BER.tr -
      geometry    800x400 &
      exit 0
}


# Procedimento para geração de tráfego.
# Parâmetros: nó transmissor, receptor, tamanho do pacote,
# rajada, tempo inativo e taxa de transmissão.
proc attach-expoo-traffic {node sink size burst idle rate} {
      global ns
      set source [new Agent/CBR/UDP]

      $ns attach-agent $node $source
      set traffic [new Traffic/Expoo]

      $traffic set packet-size $size
      $traffic set burst-time $burst
      $traffic set idle-time $idle
      $traffic set rate $rate
      $source attach-traffic $traffic

      $ns connect $source $sink
      return $source
}



      set totalpkt 0
      set erro 0
      set link_ok 1

      # Procedimento que periodicamente faz as gravações no arquivo
      # de log.
      proc record {} {
            global sink0 logf  erro totalpkt link_ok
            set ns [Simulator instance]
```

```
            # Verifica se o link está ok.
            if { $erro == 1  } {
                  set link_ok 0
              } else {
                  set link_ok 1
            }

            # Indica quanto tempo depois este procedimento deve ser
            # chamado novamente.
            set time 0.1

            # Verifica quantos bytes chegaram no receptor.
            set bw0 [$sink0 set bytes_]
            set nl  [$sink0 set nlost_]
            set np  [$sink0 set npkts_]

            # Obtém o tempo corrente.
            set now [$ns now]

            # Calcula a banda ocupada em GBit/s e a escreve
            # no arquivo de log.
            puts $logf  "$link_ok   [expr $bw0*10000/$time*8/1000000000]
            $nl   $np   $erro"

            $sink0 set bytes_ 0
            $sink0 set nlost_ 0
            $sink0 set npkts_ 0

            # Re-agenda a chamada deste procedimento.
            $ ns at [expr $now+$time] "record"

            # Calcula o número de pacotes totais.
            set bw0 [expr $bw0 / 53]
            set totalpkt [expr $totalpkt + $bw0]
      }




set prvseqnb -1
set seqerrnb 0


# ******** Configurações MPLS ************************************

# Protocolo de roteamento.
$ns rtproto DV


# Nós IP e MPLS.
set node0  [$ns node]
set LSR1   [$ns mpls-node]
set LSR2   [$ns mpls-node]
set LSR3   [$ns mpls-node]
set LSR4   [$ns mpls-node]
set LSR5   [$ns mpls-node]
```

```
set LSR6   [$ns mpls-node]
set node7 [$ns node]

set em [new ErrorModel]
$em unit pkt
$em set rate_ 0


# Construções dos links.
$ns duplex-link $node0 $LSR1  1.2Mb  5.0ms DropTail
$ns duplex-link $LSR1  $LSR2  1.2Mb  5.0ms DropTail

$ns duplex-link $LSR2  $LSR6  1.2Mb  5.0ms DropTail
$ns duplex-link $LSR2  $LSR3  1.2Mb  5.0ms DropTail
$ns duplex-link $LSR2  $LSR4  1.2Mb  5.0ms DropTail
$ns duplex-link $LSR3  $LSR4  1.2Mb  5.0ms DropTail
$ns duplex-link $LSR6  $LSR3  1.2Mb  5.0ms DropTail
# link simulado
$ns duplex-link $LSR6  $LSR4  1Mb    5.0ms DropTail
$ns duplex-link $LSR4  $LSR5  1.2Mb  5.0ms DropTail

$ns duplex-link $LSR5  $node7  1.2Mb  5.0ms DropTail
$ns lossmodel $em $LSR6 $LSR4



# Configura de dos agentes LDP em todos dos agentes MPLS.
$ns configure-ldp-on-all-mpls-nodes


# Configura a cor das mensagens LDP (visualização no nam.)
$ns ldp-request-color       blue
$ns ldp-mapping-color       red
$ns ldp-withdraw-color      magenta
$ns ldp-release-color       orange
$ns ldp-notification-color  green


set LSRmpls1 [$LSR1 get-module "MPLS"]
set LSRmpls2 [$LSR2 get-module "MPLS"]
set LSRmpls3 [$LSR3 get-module "MPLS"]
set LSRmpls4 [$LSR4 get-module "MPLS"]
set LSRmpls5 [$LSR5 get-module "MPLS"]
set LSRmpls6 [$LSR6 get-module "MPLS"]


# Habilita o modo pré-negociado do re-roteamento MPLS.
$ns enable-reroute notify-prenegotiated
$LSRmpls6 set-protection-lsp 0.7 0.01 1000

# LSP default
$ns at 0.0  "$LSRmpls1  setup-erlsp  5   1_2_6_4_5     1000"

# LSP de backup
$ns at 0.1  "$LSRmpls1  setup-erlsp  5   1_2_3_4_5      2000"

# Liga o fluxo ao LSP
```

```
$ns at 0.3  "$LSRmpls1 bind-flow-erlsp   7   100   1000"

# Liga o LSP default ao LPS backup
$ns at 0.3  "$LSRmpls1 reroute-lsp-binding      1000    2000"


# Cria o agente consumidor de tráfego e o anexa ao nó 7.
set sink0 [new Agent/LossMonitor]
$ns attach-agent $node7 $sink0
$sink0 clear

# Cria o agente produtor de tráfego e o anexa ao nó 0.
# A tag abaixo (#MSPDbnd) é necessária para que o MSPD possa gerar várias
# configurações de tráfego (0.0, 0.2, 0.4, 0.6, 0.8, 1.0 1.2 Mb) e assim
# montar as simulações finais.
#MSPDbnd
set src0 [attach-expoo-traffic $node0 $sink0 53b  5kb 0 0.2Mb]
$src0 set fid_ 100
$ns color 100 magenta

# Link da rede 10Gb, representado por 1Mb (Devido a necessidade de
# se melhorar o tempo de compilação).
# Tamanho do pacote 64KB (Bytes), representado por 64b (bits),
# isto significa que cada bit transmitido equivale a 10000 bytes


# Indica ao LSP qual entidade será o LSP de backup do link simulado.
proc notify-erlsp-setup {node lspid} {
      set ns [Simulator instance]
      set module [$node get-module "MPLS"]

        if { $lspid==1001 } {
            $module secondary-lsp-binding 1000 1001
        }
}


# Re-configuração da LIB em caso de falha
proc notify-erlsp-fail {node status lspid tr} {
      set ns [Simulator instance]
      set module [$node get-module "MPLS"]

        if { [$node id] == 1 && $status=="BSNodeError" } {
            $module set-lib-error-for-lspid $lspid 1
        }
        if { [$node id] == 1 && $status=="NodeRepair" } {
            $module set-lib-error-for-lspid $lspid -1
        }
}
```

```
# ******** Agendamentos ************************************


$ns at 0.0 "record"
$ns at 0.3 "$src0 start"


# Os agendamentos feitos abaixo foram gerados pelo MSPD e foram
# substituídos pela tag #GDP.  A perturbação feita no GDP foi em
# relação a VARIAÇÃO LINEAR DA CORRENTE.
$ns rtmodel-at  0.0000 down $LSR6 $LSR4
$ns at 0.0000 "set erro 1.0 "
$ns rtmodel-at  1.2100 up $LSR6 $LSR4
$ns at 1.2100 "set erro 0.0 "
$ns at 1.2100 "$em set rate_ 0.869761412"
$ns at 1.2100 "set erro 0.869761412 "
$ns at 1.2200 "$em set rate_ 0.238606703"
$ns at 1.2200 "set erro 0.238606703 "
$ns at 1.2300 "$em set rate_ 0.065411332"
$ns at 1.2300 "set erro 0.065411332 "
$ns at 1.2400 "$em set rate_ 0.017918886"
$ns at 1.2400 "set erro 0.017918886 "
$ns at 1.2500 "$em set rate_ 0.004905198"
$ns at 1.2500 "set erro 0.004905198 "
$ns at 1.2600 "$em set rate_ 0.001341805"
$ns at 1.2600 "set erro 0.001341805 "
$ns at 1.2700 "$em set rate_ 0.000366783"
$ns at 1.2700 "set erro 0.000366783 "
$ns at 1.2800 "$em set rate_ 0.000100188"
$ns at 1.2800 "set erro 0.000100188 "
$ns at 1.2900 "$em set rate_ 0.000027347"
$ns at 1.2900 "set erro 0.000027347 "
$ns at 1.3000 "$em set rate_ 0.000007459"
$ns at 1.3000 "set erro 0.000007459 "
$ns at 1.3100 "$em set rate_ 0.000002033"
$ns at 1.3100 "set erro 0.000002033 "
$ns at 1.3200 "$em set rate_ 0.000000554"
$ns at 1.3200 "set erro 0.000000554 "
$ns at 1.3300 "$em set rate_ 0.000000151"
$ns at 1.3300 "set erro 0.000000151 "
$ns at 1.3400 "$em set rate_ 0.000000041"
$ns at 1.3400 "set erro 0.000000041 "
$ns at 1.3500 "$em set rate_ 0.000000011"
$ns at 1.3500 "set erro 0.000000011 "
$ns at 1.3600 "$em set rate_ 0.000000003"
$ns at 1.3600 "set erro 0.000000003 "
$ns at 1.3700 "$em set rate_ 0.000000000"
$ns at 1.3700 "set erro 0.000000000 "
$ns at 1.7700 "$em set rate_ 0.000000002"
$ns at 1.7700 "set erro 0.000000002 "
$ns at 1.7800 "$em set rate_ 0.000000008"
$ns at 1.7800 "set erro 0.000000008 "
$ns at 1.7900 "$em set rate_ 0.000000032"
$ns at 1.7900 "set erro 0.000000032 "
$ns at 1.8000 "$em set rate_ 0.000000121"
$ns at 1.8000 "set erro 0.000000121 "
$ns at 1.8100 "$em set rate_ 0.000000460"
```

```
$ns at 1.8100 "set erro 0.000000460 "
$ns at 1.8200 "$em set rate_ 0.000001750"
$ns at 1.8200 "set erro 0.000001750 "
$ns at 1.8300 "$em set rate_ 0.000006671"
$ns at 1.8300 "set erro 0.000006671 "
$ns at 1.8400 "$em set rate_ 0.000025445"
$ns at 1.8400 "set erro 0.000025445 "
$ns at 1.8500 "$em set rate_ 0.000097125"
$ns at 1.8500 "set erro 0.000097125 "
$ns at 1.8600 "$em set rate_ 0.000371005"
$ns at 1.8600 "set erro 0.000371005 "
$ns at 1.8700 "$em set rate_ 0.001418247"
$ns at 1.8700 "set erro 0.001418247 "
$ns at 1.8800 "$em set rate_ 0.005425601"
$ns at 1.8800 "set erro 0.005425601 "
$ns at 1.8900 "$em set rate_ 0.020771492"
$ns at 1.8900 "set erro 0.020771492 "
$ns at 1.9000 "$em set rate_ 0.079581396"
$ns at 1.9000 "set erro 0.079581396 "
$ns at 1.9100 "$em set rate_ 0.305126259"
$ns at 1.9100 "set erro 0.305126259 "
$ns rtmodel-at  1.9200 down $LSR6 $LSR4
$ns at 1.9200 "set erro 1.0 "

# Agendamentos finais (de responsabilidade do usuário).
$ns at 4.9 "$src0 stop"
$ns at 5.0 "record"
$ns at 5.0 "finish"

# Comando para iniciar o processo de simulação do ns.
$ns run

# ******** Fim da Simulação ********************************************
```