Raciocínio Baseado em Modelo Aplicado ao Diagnóstico de Falha e Performance em Redes Locais

Sandro T. Fontanini

Raciocínio Baseado em Modelo Aplicado ao Diagnóstico de Falha e Performance em Redes Locais[1]

Sandro Tozzo Fontanini[2]

Departamento de Ciência da Computação

IC – UNICAMP

Banca Examinadora:

- Edmundo R. M. Madeira (Suplente)[3]
- Jacques Wainer (Orientador)[4]
- Eleri Cardozo[5]
- Paulo Lício de Geus[4]

[1]Dissertação apresentada ao Instituto de Computação da UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

[2]O autor é Bacharel em Ciência da Computação pela Universidade Estadual de Maringá

[3]Professor do Departamento de Ciência da Computação, IC – UNICAMP.

[4]Professor do Departamento de Ciência da Computação, IC – UNICAMP.

[5]Professor do Departamento de Engenharia de Computação e Automação Industrial, FEEC – UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Fontanini, Sandro Tozzo

F735r Raciocínio baseado em modelo aplicado ao diagnóstico de falha e performance em redes locais / Sandro Tozzo Fontanini -- Campinas, [S.P. :s.n.], 2001.

Orientador : Jacques Wainer

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Inteligência artificial. 2. Redes locais de computadores. 3. Diagnóstico. 4. Redes de computadores. I. Wainer, Jacques. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Raciocínio Baseado em Modelo Aplicado ao Diagnóstico de Falha e Performance em Redes Locais

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por Sandro Tozzo Fontanini e aprovada pela Comissão Julgadora.

Campinas, Novembro de 2001.

Jacques Wainer
Orientador

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Prefácio

A Gerência de Redes tem sido foco de muitos estudos com o crescimento das redes de computadores e com a importância que se tem atribuído à qualidade de serviço, disponibilidade e confiabilidade das mesmas. Esta disciplina abrange desde a monitoração da rede até a pró-atividade na previsão e contenção de falhas e conta com algumas aplicações de técnicas de Inteligência Artificial como o Raciocínio Baseado em Casos e os Sistemas Baseados em Regras na localização e principalmente correlação de alarmes.

Este trabalho é uma aplicação da técnica conhecida por Raciocínio Baseado em Modelo no diagnóstico de falha e performance em redes locais de computadores, como provisão de uma ferramenta simples porém necessária na gerência de uma rede de pequeno/médio porte, abrangendo desde o processo de descoberta da rede, sua monitoração, até o processo de diagnóstico.

A ferramenta aqui desenvolvida constrói e opera sobre um modelo lógico-funcional da rede, baseando-se nas informações do modo de funcionamento correto da rede que pode ser inferido através deste modelo para realizar o diagnóstico. Este sistema foi testado no Laboratório de Sistemas Integráveis da Universidade de São Paulo (LSI-USP) e mostra ser uma abordagem eficiente na gerência de redes locais de computadores.

Abstract

As higher the demand for quality of service, reliability and availability in computer networks, the Network Management is being the focus of many researches. The Network Management covers many subjects from the monitoring of the net to the fault location and avoidance. For this tasks some Artificial Intelligence techniques such as Case Based Reasoning and Rule Based Systems were used, mainly in what concerns fault location and correlation.

This work is a practical application of the Model Based Reasoning in the diagnostic of fault and performance for local area networks through a system that performs from the network discovery, the monitoring process, to the diagnostic.

The system developed here builds and operates over a logical model of the network, reasoning over the way that the network should work if in normal state. This system was tested at Laboratório de Sistemas Integráveis of the Universidade de São Paulo (LSI-USP) and is an efficient approach to the management of fault and performance in local area networks.

Agradecimentos

Agradeço em caráter especial ao meu orientador Jacques Wainer pela atenção e paciência dedicadas durante este trabalho, questionando, incentivando, ensinando.

À Volnys Bernal e Silvio Luis Marangon, do Laboratório de Sistemas Integráveis da Universidade de São Paulo, por dedicarem tempo e competência no auxílio às práticas aqui apresentadas.

Aos amigos, companheiros de morada e de luta, cuja união foi fundamental para superar obstáculos.

Com muito carinho, à minha família e noiva, sempre presentes e fiéis incentivadores desta jornada.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Com a rápida diversificação das tecnologias de redes de computadores e as mais novas funcionalidades agregadas às tecnologias já existentes, a Gerência de Redes toma uma perspectiva cada vez mais especializada e fundamental neste contexto.

Frente ao crescimento acelerado das redes, desde as locais até a Internet, frente ao aumento de tráfego de informações cada vez mais vitais para as corporações e ainda sob a demanda de uma alta qualidade de serviço, a Gerência de Redes tem sido foco na implementação de sistemas inteligentes, protocolos eficazes e padrões internacionais de interoperabilidade.

A Gerência de Redes é uma disciplina bastante vasta, ramificada em diversas tarefas como planejamento, operação, manutenção e administração, que ainda se aprofundam em outras tarefas como a gerência de falhas e de performance.

Grandes plataformas de gerenciamento foram desenvolvidas com o intuito de agregar o maior número de funcionalidades possíveis. No entanto, muitos administradores de redes locais não têm acesso a uma plataforma de tamanha abrangência, o que gera demanda por ferramentas menores e mais personalizadas [49].

Uma das tarefas da gerência de redes é a localização e o diagnóstico de falhas. Como em uma rede local alguns dispositivos não gerenciáveis podem se tornar transparentes para a rede (como hubs), esta tarefa pode se tornar complexa. Além disso, a quantidade de alarmes gerados por uma única falha na rede é normalmente grande e a filtragem destes alarmes até a obtenção de uma causa comum não é uma tarefa simples.

Algumas técnicas de Inteligência Artificial foram aplicadas neste contexto como Sistemas Especialistas Baseados em Regras [13, 22], Raciocínio Baseado em Casos [36, 2], Agentes Inteligentes [31] e Raciocínio Baseado em Modelos para correlação de alarmes [28].

Com o caráter mutável das redes atuais, o sistema de gerenciamento de falhas deve estar apto a se ajustar às constantes mudanças que ocorrem nos elementos e na topologia da rede. Alguns algoritmos para correlação de alarmes [28, 24] se ajustam a estas mudanças. No entanto, para identificar a localização exata da falha, é necessário um procedimento que envolva uma série de decisões baseadas no conhecimento humano e na análise do estado da rede. Paradigmas como o Raciocínio Baseado em Modelos ou o Raciocínio Baseado em Casos são adequados para este tipo de tarefa [36, 6].

Este trabalho é uma aplicação do paradigma de Raciocínio Baseado em Modelo no diagnóstico de falhas em redes locais baseadas no protocolo TCP/IP através de um modelo de resolução de problemas (PSM) sugerido em [6, 5, 7].

Este trabalho, iniciado por pesquisadores do Laboratório de Sistemas Integráveis da Universidade de São Paulo (LSI-USP) teve como planta de testes a rede local deste laboratório com a colaboração do seu administrador e do administrador da rede.

O capítulo seguinte apresenta a teoria sobre diagnósticos e o que é o Raciocínio Baseado em Modelo, apresentando ainda, um pequeno exemplo de sua aplicação no contexto de redes locais. O Capítulo 3 mostra os detalhes da teoria sobre redes de computadores, os dispositivos que a compõem, o protocolo TCP/IP, aborda os aspectos mais importantes da gerência de redes e do protocolo SNMP, posiciona este trabalho dentro da gerência de redes e faz uma breve apresentação das plataformas existentes.

O Capítulo 4 é uma descrição minuciosa do sistema desenvolvido. O Capítulo 5, por fim mostra as características da rede local LSI-USP e os testes realizados na mesma. A conclusão apresenta uma discussão sobre os principais aspectos e impactos da aplicação deste sistema sugerindo trabalhos futuros.

Capítulo 2

Diagnóstico e Raciocínio Baseado em Modelo

O diagnóstico é uma disciplina bastante conhecida na Inteligência Artificial sendo uma das primeiras abordagens na tentativa de se construir mecanismos com conhecimento embutido capazes de inferir de forma autônoma.

As primeiras e mais conhecidas aplicações de técnicas de IA em diagnóstico foram justamente na medicina, através de sistemas como o MYCIN, GLAUCOMA, INTERNIST-I [40, 22]. Outras aplicações se estenderam para o diagnóstico automatizado de componentes eletrônicos/mecânicos [14, 37]. Também o diagnóstico de falhas em redes de computadores recebeu atenção de técnicas de IA, um campo bem explorado [31, 29, 48, 30], principalmente devido a demanda por ferramentas que automatizassem este processo, acelerando a resolução de um problema que um bom especialista levaria horas para fazê-lo. Algumas das ferramentas desenvolvidas como o DRUM-II [20] e IMPACT [28] são exemplos de aplicação de técnicas de Inteligência Artificial na correlação de alarmes , uma das etapas no processo de diagnóstico em redes de computadores.

O Raciocínio Baseado em Modelos é uma destas técnicas que já vem sendo explorada por alguns pesquisadores tanto no diagnóstico de sistemas estáticos, como é o caso de componentes eletrônicos, bem como em sistemas mutáveis como as redes de computadores, que mudam com certa freqüência sua configuração [48, 35, 20].

Outras abordagens podem ser utilizadas no mesmo problema, um bom exemplo é a técnica de Raciocínio Baseado em Casos [2, 36], Agentes Inteligentes [31] ou os Sistemas Baseados em Regras [10].

Este capítulo dará uma visão geral do que se entende por diagnóstico na Inteligência Artificial e fará uma introdução necessária ao Raciocínio Baseado em Modelo, apresentando um breve modelo de diagnóstico utilizando esta técnica.

2.1 Diagnóstico, sintomas e causas

O diagnóstico é o processo de se encontrar uma ou mais explicações (causas) plausíveis para os sintomas encontrados em algum domínio. Esta tarefa, seja feita de maneira natural ou através de recursos artificiais é bastante complexa e envolve, em muitos casos, a aquisição de um grande montante de conhecimento e um longo período de aprendizagem para que se torne uma tarefa confiável e precisa.

O raciocínio utilizado em diagnósticos pode ser classificado como um tipo de inferência conhecida por raciocínio abdutivo ou abdução, que é o processo de gerar a melhor explicação, segundo um critério, para um dado conjunto de observações e fatos.

Sintomas ou efeitos são as alterações do estado normal de um sistema. Para o domínio de redes de computadores, um sintoma é uma falha , é um fato observado como uma perda de sinal (LOS). Uma causa pode ser comparada a uma doença, é o que gera o sintoma, por exemplo, o rompimento de um cabo.

Um problema de diagnóstico é o conjunto de doenças, sintomas e a relação de causas envolvidos num sistema [40]. As diversas possibilidades da relação de causa e efeito constituem uma rede causal, como representado na figura 2.1.

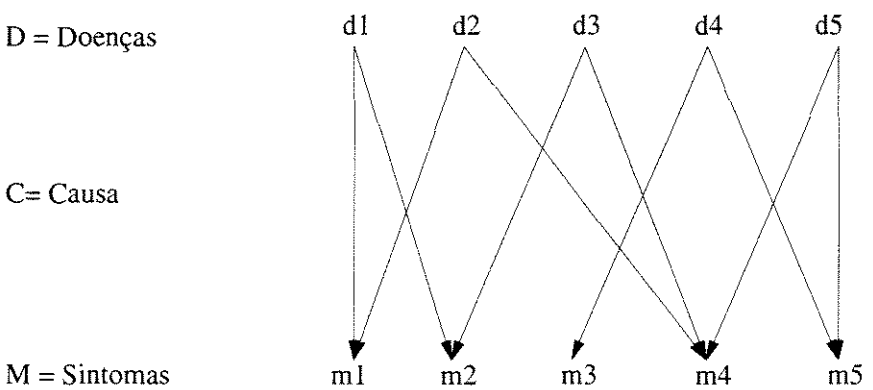


Figura 2.1: Rede Causal

A solução de um problema de diagnóstico é o conjunto de todas as explicações (relações

de causa e efeito) dos sintomas. No entanto, este conjunto pode ser muito grande. Para dar à solução maior precisão, é adotado um critério chamado parcimônia.

A parcimônia restringe todas as possibilidades de explicação de um problema de diagnóstico somente àquelas que possam constituir uma explicação plausível. Se a parcimônia para uma solução for a menor cardinalidade, então, é considerado plausível o conjunto que contenha o menor número de doenças, chamado de *Cobertura Mínima*. Outro critério conhecido é a redundância. Uma explicação é irredundante se nenhum subconjunto das doenças continua explicando os sintomas.

A parcimônia adotada neste trabalho é a de cobertura mínima, ou seja, o menor conjunto de dispositivos de rede falhos é considerado como uma explicação plausível para os sintomas.

As diversas técnicas da Inteligência Artificial já aplicadas no diagnóstico podem ser divididas em dois principais domínios: os sistemas baseados em experiência e os sistemas baseados em modelo. O primeiro é representado por Sistemas Baseados em Regras e Raciocínio Baseado em Casos (CBR)[16, 30]. O segundo é representado pelo Raciocínio Baseado em Modelo (MBR) e pode ser dividido em Modelo de Funcionamento Correto e Modelo de Falhas.

2.1.1 Raciocínio Baseado em Modelo

O Raciocínio Baseado em Modelo é um paradigma da Inteligência Artificial que consiste em representar um sistema através de um modelo lógico e funcional [35]. Para o diagnóstico de redes de computadores, este modelo contém informações dos equipamentos contidos na rede, sua topologia e informações de roteamento.

O processo básico do MBR trata da interação entre o comportamento previsto do sistema que está sendo diagnosticado e as observações sobre o que realmente está acontecendo.

No diagnóstico de *Funcionamento Correto*, uma descrição especifica a estrutura e o funcionamento *correto* do sistema. Cada componente está associado a um comportamento que representa seu estado normal. Assumindo que todos os dispositivos estejam trabalhando corretamente, o funcionamento de todo o sistema pode ser previsto. As observações coletadas durante o processo de monitoração do sistema tentam deduzir o estado atual do mesmo. As discrepâncias encontradas entre o modelo e as observações geram o diagnóstico.

A solução de um problema de diagnóstico baseado em funcionamento correto é um conjunto de causas que, considerando que os outros componentes estejam funcionando corretamente, são consistentes com a descrição do sistema e com seu estado atual. Ainda é necessário que para se chegar a esse conjunto, alguma parcimônia seja adotada. Neste trabalho, o melhor diagnóstico

é o conjunto com menor número de causas.

O *Modelo de Falhas* é outra vertente dos estudos de diagnósticos. A solução de um problema de diagnóstico baseado no Modelo de Falhas é também um conjunto parcimonioso de comportamentos (causas) que, juntamente com a descrição lógica do sistema, implicam as observações detectadas. No entanto, o foco deste paradigma está em representar o comportamento do sistema quando em estado falho. Os componentes do sistema geram uma série de observações quando em mal funcionamento. Se essas observações condizem com a representação falha do sistema, então o diagnóstico pode ser inferido.

Para alcançar esta precisão, o diagnóstico baseado no Modelo de Falhas precisa conhecer a priori todo o domínio de falhas e causas para confeccionar uma estrutura denominada rede causal (ver figura 2.1), ou seja, um grafo exaustivo de relações entre causa e sintoma [21]. Em domínios como a medicina em que a relação sintoma-causa é bastante conhecida o diagnóstico baseado em falha é mais utilizado (conhecido por diagnóstico abdutivo). Já em sistemas em que as falhas e suas causas não são bem conhecidas o diagnóstico baseado em funcionamento correto é mais eficiente e mais simples.

A escolha de uma destas formas de diagnóstico depende de quão detalhado é o conhecimento sobre o sistema e sobre seu comportamento. Alguns autores mostraram que, sob as mesmas condições, o diagnóstico baseado em funcionamento correto e o baseado em falhas geram o mesmo resultado [21].

2.1.2 Raciocínio Baseado em Funcionamento Correto

O raciocínio baseado no modelo de funcionamento correto foi adotado neste trabalho para localizar e diagnosticar falhas em uma rede de computadores local (LAN).

Neste paradigma o modelo construído serve como referência para compreender o funcionamento correto do sistema e compará-lo ao estado corrente do mesmo, coletado através de testes e observações. As discrepâncias entre o comportamento previsto e o observado indicam o mal funcionamento do sistema e identificam os pontos onde as falhas ocorreram. O esquema de funcionamento deste processo é representado na figura 2.2.

Durante o processo de diagnóstico, as discrepâncias são exploradas através de novos testes. A identificação do ponto falho é a solução do problema de diagnóstico, sua explicação.

Não é necessário admitir que o modelo esteja absolutamente fiel ao sistema real. Em geral, se a aproximação é suficientemente boa, o MBR mostra ser uma ferramenta adequada para diagnósticos [21].

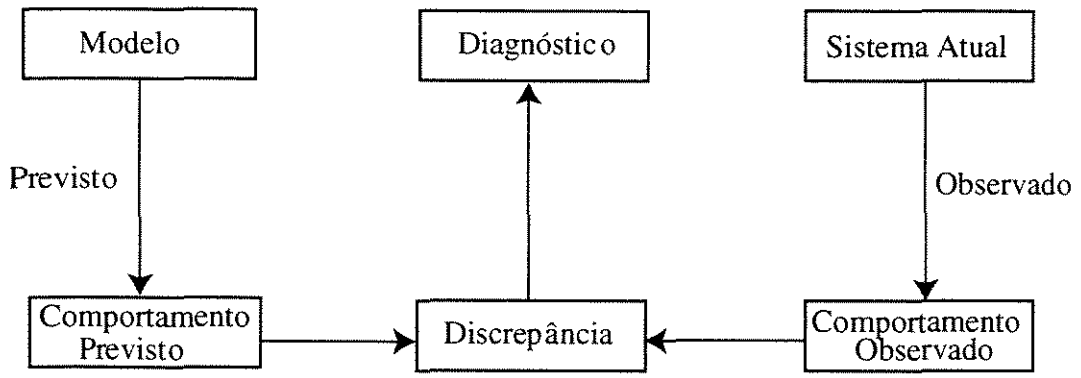


Figura 2.2: Funcionamento do Raciocínio Baseado em Modelos [21].

O MBR foi amplamente utilizado em testes sobre circuitos, placas e dispositivos eletrônicos através da exploração do conhecimento sobre a estrutura e o funcionamento correto destes sistemas [14, 20].

Em sua aplicação no diagnóstico de redes a "máquina de inferência" é a concretização do raciocínio de um administrador que conhece o modo como sua rede deve funcionar e a partir do status falho da mesma é capaz de inferir sobre quais dispositivos apresentam problemas. Portanto, tendo disponível um modelo conciso do elemento a ser analisado, o raciocínio baseado em funcionamento correto faz uma intersecção do seu estado atual com este modelo, e a partir das diferenças encontradas é capaz de levantar uma hipótese inicial, a ser refinada para comprovar o problema [7, 6].

A construção do modelo é na maioria das aplicações orientada a componentes. De forma similar, o modelo de uma LAN, como o construído para este trabalho, tem como base o conjunto de componentes, adicionado às diferentes funcionalidades de cada um nas camadas do modelo TCP/IP. Estes podem ser redes, nós, uma camada na pilha de protocolos, um processo de software, um link ou um hardware, dependendo do detalhamento do modelo.

Os switches, cabos, hubs, roteadores e terminais formam um conjunto de componentes chamado *DISPOSITIVOS*. Somado à informação de como estes componentes estão conectados entre si, chega-se a topologia, um primeiro modelo estrutural e funcional do sistema. Cada componente tem um comportamento próprio. Sua função na rede contribui para a melhoria do modelo de funcionamento da rede. Por exemplo, sabendo-se que os hubs são repetidores de múltiplas portas, é possível inferir que um pacote enviado por qualquer dispositivo conectado ao mesmo passará por todos os cabos e será recebido por todos os outros dispositivos também

conectados.

O esquema de endereçamento físico e lógico e as tabelas de roteamento contribuem com a mais importante informação funcional, o caminho pelo qual um pacote irá passar. Finalmente, o modelo é complementado com informações de performance, como os limites de taxa de utilização, limites de latência e perda de pacotes.

Com informações suficientes o comportamento do sistema pode ser identificado através de inferências simples, desde que o modo de funcionamento dos componentes seja também compreendido. O exemplo abaixo ilustra uma das formas de localização de falhas utilizada.

Consideremos o switch S que conecta os computadores $C1$ e $C2$ como mostrado na figura abaixo (2.3). Supondo que o computador $C1$, tenta transmitir algum pacote para $C2$, e recebe o alarme de que $C2$ está inatingível.

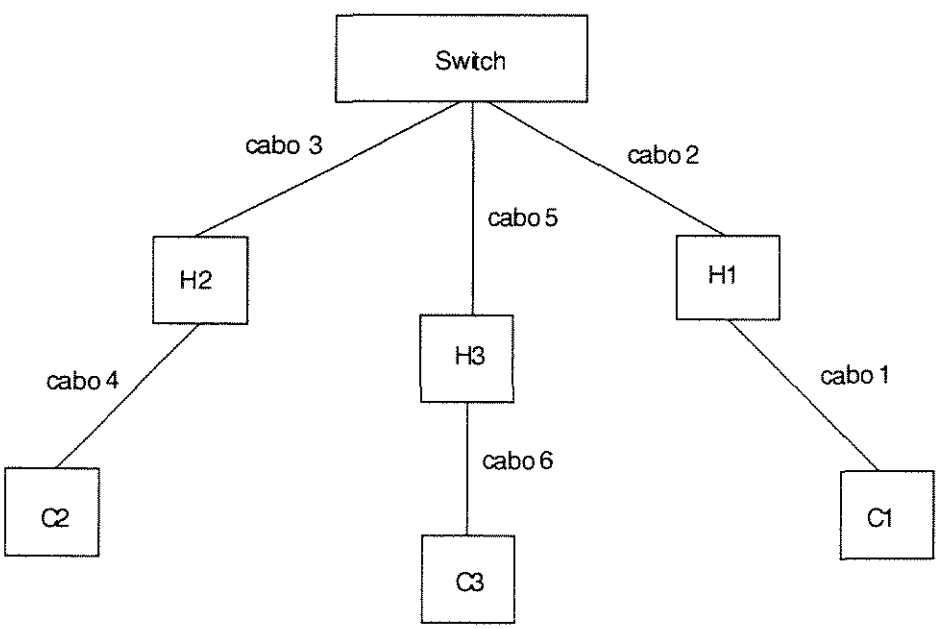


Figura 2.3: Exemplo de um caminho local onde o diagnóstico será realizado.

Através da análise do modelo construído e da comparação dos endereços IP destas máquinas, sabe-se que ambos os computadores encontram-se na mesma subrede e portanto um caminho local deve ser traçado para conhecer o caminho real por onde este pacote passaria. A hipótese inicial para o diagnóstico é formada por todos os componentes no caminho. O funcionamento correto é ilustrado pela seguinte inferência:

Seja $\phi(Conhecimento)$ a representação do conhecimento do caminho de um pa-

cote, então:

$\phi(Conhecimento) \wedge Comp(C1, OK) \wedge Cabo(cabo1, OK) \wedge Hub(H1, OK) \wedge$
$Cabo(cabo2, OK) \wedge Swi(S, OK) \wedge Cabo(cabo3, OK) \wedge Hub(H2, OK) \wedge Cabo(cabo4, OK) \wedge$
$Comp(C2, OK) \rightarrow Alcancavel(C2)$

Onde $Comp(X, OK)$ indica que o computador X encontra-se em estado normal, de forma semelhante para os outros componentes e $Alcancavel(C2)$ indica que um ping para este dispositivo receberá resposta normalmente.

Um dos principais motivos pelo qual a função de representação do conhecimento do caminho de um pacote é usada, é para eliminar da hipótese componentes como por exemplo o hub $H3$ e o computador $C3$ da figura 2.3, que não precisam ser verificados.

Qualquer desvio nesta norma resulta numa falha para o envio de um pacote. Portanto, qualquer dispositivo no caminho poderá ser o responsável pelo problema.

O passo posterior do diagnóstico é fazer um refinamento que localize a falha com precisão, através de testes. Supondo que o switch é testado através de um ping temos o seguinte refinamento:

$Hipotese = (C1, cabo1, H1, cabo2, S, cabo3, H2, cabo4, C2)$

se $Alcancavel(S)$ então

$Hipotese = Hipotese - (Hipotese \cap caminho(C1, S))$

Portanto, $Hipotese = (cabo3, H2, cabo4, C2)$

Onde $caminho(C1, S)$ é o cômputo do caminho entre estes dois dispositivos.

Este tipo de inferência elimina os testes redundantes em outros elementos da hipótese. Como o modelo fornece uma referência do funcionamento deste caminho, é possível eliminar com segurança os componentes que estão antes do switch S, uma vez que para que o mesmo respondesse ao ping, todos os outros deveriam estar funcionando corretamente.

Com mais uma iteração, os componentes restantes na hipótese poderiam ser testados, até se alcançar a localização exata da falha, ou seja, um único dispositivo.

Por se tratar de uma representação lógica em uma linguagem formal, os modelos de um sistema serão sempre diferentes do sistema real [21]. Uma representação possui simplificações e premissas que a tornam por natureza diferente. Por isso o sucesso na tarefa de modelagem está atrelada ao grau de abstração que se deseja obter. Da mesma forma, a eficiência do diagnóstico

depende desta abstração. Um modelo minucioso demais pode dificultar a tarefa de diagnóstico a ponto de perder a simplicidade e a eficácia.

Esta abstração pode ser tanto estrutural quanto funcional. A abstração estrutural agrega diversos componentes em um único dispositivo. Por exemplo, as diversas interfaces de um switch podem ser agregadas em um único componente *Switch*. A abstração funcional está mais associada à simplificação da nomenclatura. Pode-se dizer simplesmente que um dispositivo está inalcançável (Loss of Signal), porém, funcionalmente sabe-se que o mesmo não responde a um comando dentro de um tempo predeterminado (ping timeout).

O princípio para a construção do modelo é atingir uma representação tão simples quanto possível e tão sofisticada quanto necessário [17].

2.2 Sumário

A escolha de uma técnica de IA no problema de diagnóstico depende do tipo de sistema que se deseja diagnosticar, seu comportamento, funcionalidades e estrutura.

O Raciocínio Baseado em Modelos mostra ser uma ferramenta eficiente que faz o diagnóstico através das informações que têm sobre o funcionamento do sistema representadas em um modelo estrutural e funcional.

Fazendo-se a abstração correta do sistema, é possível obter um modelo simples e suficientemente completo para a representação do mesmo e para a obtenção de inferências precisas.

O sistema desenvolvido neste trabalho adota como ferramenta o raciocínio baseado em funcionamento correto, uma das vertentes do Raciocínio Baseado em Modelo. O funcionamento dos dispositivos de uma rede, com o grau de abstração coerente, é simples de ser representado. O caminho por onde um pacote IP trafega pode ser calculado, sendo uma fonte de conhecimento vital sobre a rede, possibilitando as diversas inferências até o diagnóstico. A parcimônia adotada para a obtenção da solução de um problema é a cardinalidade mínima.

Capítulo 3

Redes Locais e Gerência de Redes

Este capítulo apresenta brevemente os principais conceitos sobre Redes Locais de Computadores (LANs) bem como sobre o protocolo de gerenciamento de redes SNMP (Simple Network Management Protocol). A primeira seção é dedicada a uma introdução dos conceitos de redes locais, seus principais componentes, mostrando as camadas de referência ISO/OSI em que este trabalho foi aplicado, situando conceitos como meio físico, interfaces, endereçamento IP e MAC, roteamento e domínios de colisão.

A seção seguinte apresenta o protocolo SNMP de gerência de redes, mostrando como funciona sua arquitetura, alguns comandos utilizados e as Management Base Information (MIBs) detalhando as informações nela contidas e listando as bases utilizadas neste trabalho.

Finalmente é feita uma breve apresentação de algumas plataformas de gerenciamento existentes e uma descrição mais detalhada sobre a gerência de falhas e a gerência de performance, items escolhidos na aplicação do presente trabalho.

3.1 Redes Locais de Computadores (Local Area Networks - LANs)

Formalmente, as Redes Locais de Computadores (Local Area Network - LAN) são definidas como um conjunto de computadores conectados fisicamente dentro de uma área geográfica restrita. Sua distribuição espacial está normalmente associada à tecnologia de transmissão e arquitetura utilizadas, porém, na prática a fronteira das redes locais encontra-se dentro de uma mesma corporação, modo de gerenciamento e velocidade [47, 50]. As LANs possuem diversas

configurações de taxas de transmissão que podem estar entre 1,10 e 100 Mbps para a arquitetura Ethernet/Fast Ethernet e 1 Gbps para a tecnologia Gigabit Ethernet, no entanto a sua definição está mais atrelada a sua distribuição espacial.

A maioria das redes de comunicação de dados e dos protocolos desenvolvidos para as mesmas estão baseados no modelo de referência ISO/OSI (International Organization for Standardization / Open Systems Interconnection) [19], um conjunto de padrões formalizados com a finalidade de prover interoperabilidade entre redes de diferentes corporações e fabricantes. Este modelo de referência possui sete camadas que por definição fornecem serviços umas às outras, cobrindo todos os aspectos de comunicação em redes (ver figura 3.1).

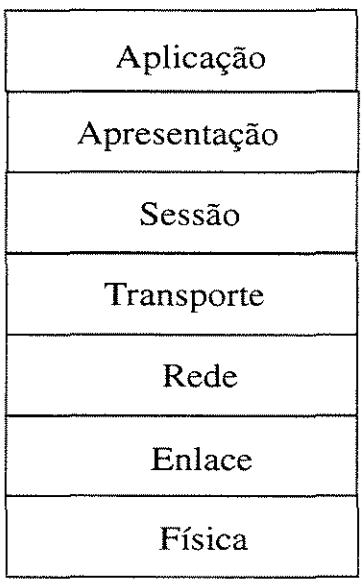


Figura 3.1: As camadas de referência do modelo OSI

Onde as quatro inferiores podem ser agrupadas segundo sua função de transporte de dados, e as demais agrupadas na função de aplicação. Cada camada é brevemente descrita a seguir:

- *Camada Física*: estabelece a conexão física entre equipamentos na rede, garantindo a transmissão de bits entre dois sistemas distintos;
- *Camada de Enlace de Dados*: oferece como serviços à camada superior a transmissão de pacotes e as funções de detecção e correção de erros, dando confiabilidade à transmissão de dados;
- *Camada de Rede*: responsável pela função de endereçamento e roteamento de pacotes pela rede. Também controla a taxa de transmissão, evitando congestionamentos;

- *Camada de Transporte*: garante o fluxo de dados entre origem e destino, assegurando que os dados cheguem corretamente no receptor. Também responsabiliza-se pela negociação das taxas de transmissão entre o dispositivo de origem e de destino. No destino, a camada de transporte remonta os pacotes e os entrega para a camada superior;
- *Camada de Sessão*: faz a negociação do início e do encerramento de um canal de comunicação, autentica o dispositivo de origem e tem direitos de estabelecer uma conexão, tratando também do sincronismo entre os dois sistemas;
- *Camada de Apresentação*: faz a conversão de padrões de comunicação tanto para pacotes que saem da máquina com destino a um dispositivo com outros protocolos como para os pacotes recebidos de diferentes protocolos;
- *Camada de Aplicação*: fornece os serviços de rede como transferência de arquivos, acesso e execução remotos, e-mail dentre outros. Garante a interface entre usuários e a rede.

Operando respectivamente nas camadas de Rede e de Transporte encontram-se os protocolos IP (Internet Protocol) e TCP (Transmission Control Protocol), padrões amplamente difundidos nas redes atuais. O IP foi projetado para permitir a interconexão de redes de computadores que utilizam a tecnologia de comutação de pacotes, o que forma hoje a Internet [12]. O TCP é um protocolo orientado a conexão que fornece confiabilidade ao serviço de transferência de dados fim a fim, projetado para funcionar sobre um serviço de rede sem conexão e sem confirmação. Maiores detalhes sobre estes protocolos serão descritos após uma introdução sobre os componentes físicos de uma LAN, devido sua importância no domínio deste trabalho.

3.1.1 Dispositivos de Rede

Várias tecnologias de redes são envolvidas na implementação de uma rede local, tecnologias essas que provêem diferentes graus de performance, confiabilidade e escalabilidade. Elas diferem em velocidade de transmissão, comprimento dos cabos e custo. As principais arquiteturas usadas em redes locais são a Ethernet, Token Ring e Fiber Distributed Data Interface (FDDI). Esta seção terá como foco o padrão Ethernet, devido ao tipo de rede utilizada para os testes deste trabalho.

O Ethernet é atualmente a tecnologia LAN predominante. É capaz de suportar vários tipos de protocolos em uso (por exemplo o TCP/IP, protocolo utilizado neste trabalho) e trabalhar com os diversos tipos de dispositivos que aderem ao padrão IEEE (Institute of Electrical and

Electronic Engineers). Sua principal característica é o uso de contenção através do algoritmo CSMA/CD (Carrier Sense Multiple Access/ Collision Detection) como método de controle de acesso ao meio de transmissão [1].

Os dispositivos básicos para comunicação que compõem uma rede são os repetidores, hubs, bridges, roteadores e switches, que operam em camadas diferentes no modelo de referência OSI, trabalhando com diferentes protocolos e em funções distintas.

Os repetidores, que operam na Camada Física, são equipamentos usados para conectar dois segmentos de rede formando um segmento maior. Sua função é a de simplesmente amplificar o sinal trafegado pelo meio.

Também operando na Camada física encontram-se os hubs. Estes são simplesmente repetidores com múltiplas portas, capazes de conectar vários equipamentos num único segmento de rede. Hubs modernos já implementam gerenciamento através do protocolo SNMP, o que será detalhado adiante.

Na Camada de Enlace de Dados, encontram-se os switches e bridges. A função do switch é a de chaveamento. Ele é capaz de encaminhar os dados de uma origem para seu respectivo destino através da verificação do endereço MAC (Media Access Control) de cada pacote, um endereço identificador único fornecido pelos fabricantes de componentes de rede. Devido a possibilidade de desempacotar dados neste nível, os switches limitam o tráfego, ao contrário dos hubs que simplesmente repetem o sinal para todas as suas portas. No entanto os switches limitam-se a esta função, sem compreender protocolos de nível superior. Atualmente alguns switches agregam as capacidades de um roteador, e são conhecidos por switch-routers. As bridges são dispositivos que separam duas redes na camada de enlace, atualmente sua função é atribuída aos switches, no entanto, são responsáveis pela conversão de sinal entre redes de características físicas distintas porém com o mesmo esquema de endereçamento sobre o enlace de dados.

Os roteadores operam na Camada de Rede e são usados para conectar várias redes distintas, oferecendo um modo inteligente de transferência de pacotes de uma rede a outra, são capazes de interpretar os endereços IP, em uma rede baseada neste protocolo.

Finalmente, os gateways operam na última camada, a de Aplicação, comunicando-se com todas as camadas inferiores do modelo OSI. Trata-se de um sistema de comunicação que pode trabalhar com quaisquer protocolos, completando a comunicação entre dois fins até a aplicação desejada por um usuário.

Estes componentes são comuns em redes locais de computadores. Sua quantidade varia de

acordo com o tamanho da rede em questão. Suas operações estão diretamente ligadas ao tipo de tecnologia utilizada (transmissão e protocolos) garantindo diferentes características para cada rede. Cada equipamento possui sua própria característica de gerenciamento, sendo portanto pró-ativo no fornecimento de informações sobre seu estado, o melhor caso, simplesmente armazenado suas informações de configuração, ou não implementando nenhum mecanismo de gerenciamento, o que faz destes dispositivos elementos problemáticos no diagnóstico de falhas da rede. A presença de informações de gerência nos dispositivos depende de sua tecnologia. Muitos hubs ainda operantes em redes locais são um bom exemplo de dispositivos sem gerenciamento. As vantagens e dificuldades da gerência destes equipamentos será detalhada posteriormente.

3.2 O protocolo TCP/IP

O *Transmission Control Protocol/Internet Protocol* é um conjunto de protocolos de comunicação que permite que computadores de arquiteturas e sistemas operacionais diferentes sejam interconectados. São os protocolos em uso na Internet, os mais difundidos atualmente.

O TCP é um protocolo da camada de Transporte orientado a conexão que divide os dados a serem transmitidos em segmentos, estes por sua vez são sequenciados pelo TCP transmissor e confirmados um a um pelo TCP do dispositivo receptor. O TCP é responsável pela negociação dos formatos de transmissão (janelas com tamanho predeterminado), pela sinalização de urgência dos pacotes, ordenação dos mesmos e pelos algoritmos de checagem de erros. Um segmento TCP consiste de um cabeçalho contendo as informações acima mencionadas somadas ao campo de dados [12].

O TCP, juntamente com o User Datagram Protocol (UDP)[42] realiza funções similares às da camada de transporte do modelo OSI.

O protocolo IP, trabalha uma camada abaixo, a camada de Rede.

A responsabilidade do protocolo IP é a organização dos endereços para os quais os pacotes se destinam, da origem dos pacotes recebidos bem como a adição de opções de entrega para pacotes vindos do TCP. O IP organiza suas informações em um cabeçalho próprio, adicionando-o ao pacote vindo da camada superior, formando um datagrama. Trata-se de um protocolo não orientado a conexão e sem confirmação, o que significa que pacotes perdidos na rede não são de responsabilidade do mesmo.

Uma importante característica deste protocolo, é a adoção de um padrão de endereçamento,

capaz de identificar redes, sub-redes e máquinas em qualquer parte do mundo. Este endereço consiste de uma campo de 32 bits (IP-Versão 4)[23, 12], separados em quatro octetos e representados de forma decimal pontuada.

Os endereços IP são divididos em classes que separam domínios de grandes corporações até pequenas redes locais de modo que haja compatibilidade de endereços, organizando-os de forma a não sobrepor ou sub-utilizar uma fração dos mesmos, uma vez que sua utilização é global.

A gama de endereços que cada corporação ou administrador de rede pode utilizar em sua rede particular é atribuída pelo IANA (Internet Assigned Numbers Authority) [43], uma administração responsável pela organização destes endereços pelo mundo.

Para uma determinada LAN, o administrador de rede pode atribuir os endereços IP para os dispositivos da forma que lhe convier, dentro da gama de endereços a ele atribuído , o que torna cada dispositivo conhecido por toda a Internet. Esses endereços podem ser configurados manualmente ou através de protocolos específicos (DHCP por exemplo [18]), dividindo a rede segundo critérios de roteamento, carga/utilização e topologia.

Deste modo introduz-se o conceito de sub-redes, uma faixa específica de endereços IP que identificam um conjunto de dispositivos dentro de uma mesma rede, podendo estar fisicamente independente. Para este fim, uma máscara de rede é utilizada. Trata-se também de um número de quarto octetos que identifica quais bits do endereço IP serão delegados para representar a rede, e quantos bits indicarão o equipamento (*host*).

Uma rede é então dividida em sub-redes, otimizando-a de acordo com as aplicações de cada usuário, balanceando sua carga de utilização e restringindo domínios de roteamento segundo critérios de segurança. Esta divisão possibilita a implementação de algoritmos de roteamento, restringindo o fluxo de pacotes somente às áreas previamente especificadas, evitando sobrecarga sobre outros domínios.

Os algoritmos de roteamento são implementados nos roteadores, que operam na Camada de Rede como visto anteriormente. Os roteadores são capazes de manusear e atualizar uma tabela interna que indica qual é o próximo endereço para o qual um determinado pacote com endereço IP de destino conhecido deve ser enviado. Para tanto eles são capazes de desempacotar os segmentos até reconhecer o IP destino do mesmo e então redirecioná-lo. Em redes de pequeno porte as tabelas de roteamento são inseridas manualmente.

Alguns tipos de endereço reservado são programados como segue:

- Loopback Address: o endereço 127.0.0.0 é reservado em todos os equipamentos destina-

se a pacotes na mesma máquina para testes ou conversação inter-processos;

- Default Route: o endereço do roteador para o qual um dispositivo envia pacotes. Esse roteador é predefinido pelo administrador, e configurado em todos os equipamentos;
- Broadcast Address: a porção HOST do endereço IP configurado para uma determinada rede pode ser preenchido com bits 1, a fim de indicar um endereço onde o pacote é destinado a todos os hosts pertencentes àquela rede;
- Multicast Address: um endereço reservado que destina pacotes a um grupo pré-definido de máquinas;

Os algoritmos de roteamento podem ser genericamente classificados em estáticos ou dinâmicos. No primeiro, uma tabela de rotas mantida pelo roteador é criada manualmente e a transmissão de pacotes segue fielmente suas especificações. No roteamento dinâmico, outras variáveis como estatísticas de performance e falhas na rede fazem com que o roteador opte imediatamente por rotas alternativas. Desta forma, no sistema de roteamento dinâmico, uma rota para os pacotes não pode ser prevista [50, 47, 12].

A propagação dos pacotes segundo seu modo de transmissão divide as redes também em conjuntos chamados domínios de colisão. Os domínios de colisão são porções da rede que compartilham quaisquer pacotes do meio físico. Isto acontece devido a pequena capacidade de seleção dos hubs, por exemplo, que reproduzem para todas as suas portas qualquer sinal capturado no meio, independente do seu destino. Possuem este nome por serem o ponto crítico de colisões de pacotes (CSMA/CD)[1]. Os switches delimitam as fronteiras dos domínios de colisão uma vez que são capazes de resolver endereçamento na Camada de Enlace de dados, através dos endereços MAC, impedindo que pacotes com endereço restrito àquele domínio sejam transmitidos para outro destino.

Os domínios de colisão são grandes responsáveis por problemas de excesso de tráfego (taxa de utilização). Uma má configuração de equipamentos, encadeamento de hubs, ou excesso de terminais compartilhando o mesmo meio físico pode ser fonte de erros de performance em uma rede.

A soma das informações do endereço IP (camada de rede), sua máscara, o endereço MAC (camada de enlace) e a forma como os protocolos TCP/IP transportam seus dados, permite que o caminho entre origem e destino de um pacote seja previamente calculado. Este cálculo (feito através de cômputos lógicos do endereço IP e sua máscara)[12], permite identificar se origem e

destino pertencem a mesma subrede, mesma rede, ou locais totalmente distintos. A informação de como os pacotes IP trafegam pela rede é primordial no sistema aqui desenvolvido.

3.3 Gerência de Redes

Com a criação e crescimento acelerado das redes de computadores a disciplina de Gerência de Redes surgiu, a fim de garantir confiabilidade, qualidade de serviço, alta performance e segurança.

A Gerência de Redes envolve desde a monitoração através de um analisador de protocolos até a manutenção de bases de dados distribuídos, polling dos dispositivos e ambientes gráficos de controle de topologia e tráfego [49, 46].

Grande parte das arquiteturas de gerência de redes usam a mesma estrutura, estações clientes que possuem sistemas de geração de alarmes e as entidades de gerência, que capturam e interpretam estes alarmes de forma a diagnosticar um problema da rede, criar os arquivos de *Log* e reiniciar o sistema quando necessário.

Segundo especificações da ISO (International Standards Organization) a gerência de redes pode ser dividida em cinco áreas conceituais:

- Gerência de falha: detectar, armazenar, notificar e corrigir erros para manter a rede em funcionamento. Uma falha na rede pode causar perdas irreparáveis, por isso é considerado um dos pontos críticos do gerenciamento;
- Gerência de performance: mede características como taxa de utilização do meio e latência de resposta a fim de manter a performance em níveis aceitáveis;
- Gerência de contas/quotas: mede os parâmetros de utilização da rede de forma a atribuir prioridades e quotas para usuários de grupo ou individuais;
- Gerência de configuração: monitora informações de configuração da rede como sistemas operacionais, versão de interfaces e versão de protocolos;
- Gerência de segurança: faz controle de acesso aos recursos a fim de evitar que usuários mal intencionados impactem de alguma forma o funcionamento da rede;

Este trabalho trata do gerenciamento dos dois primeiros items, falha e performance.

Assim como os protocolos de comunicação entre dispositivos na rede, alguns protocolos de gerência também foram desenvolvidos de modo que informações, diagnósticos e comandos de gerenciamento pudessem trafegar rápida e eficientemente. Dentre os protocolos desenvolvidos com esse fim, o SNMP (Simple Network Management Protocol) se tornou popular pela sua simplicidade e eficácia, e por se tratar do protocolo de gerência para redes TCP/IP [49, 46, 11].

Várias plataformas de gerenciamento foram desenvolvidas sobre este protocolo, por exemplo a HPOpenview [39] e Netcool [25, 26], plataformas completas capazes de controlar redes nos diversos níveis da camada OSI.

O SNMP é baseado no paradigma gerente/agente, onde cada dispositivo de rede é capaz de responder a um elemento central de gerenciamento sobre questões de confiabilidade e performance dentre outras informações (ver figura 3.2).

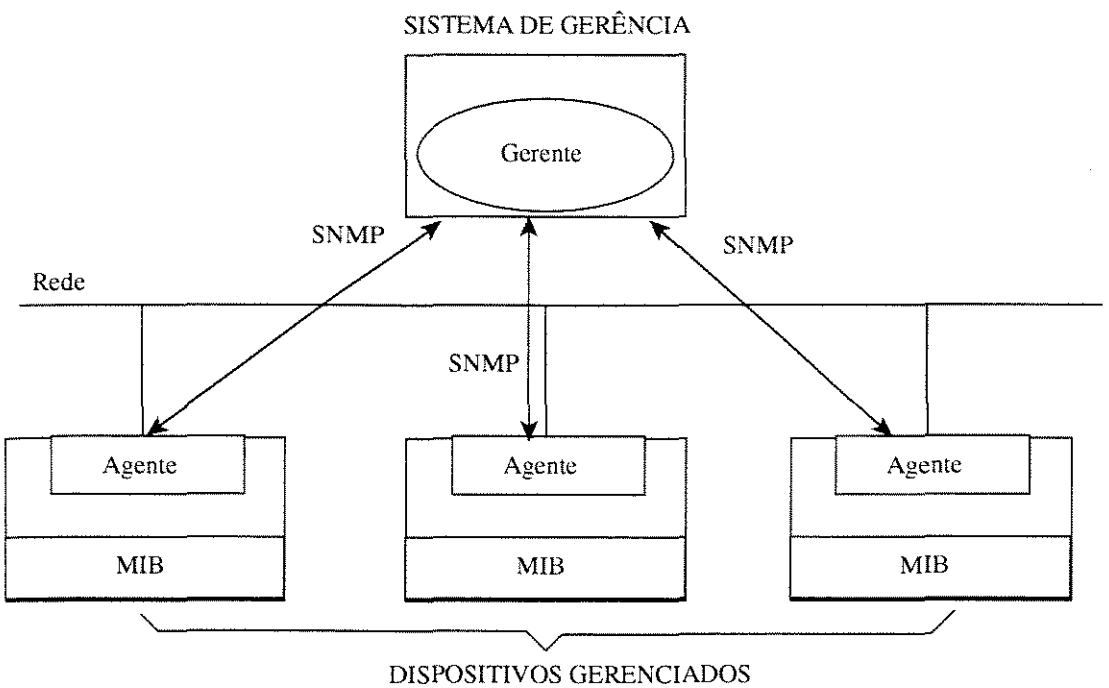


Figura 3.2: Arquitetura Gerente-agente do SNMP

Este protocolo foi projetado para ser aplicado em quaisquer tipos de dispositivos, independente da sua complexidade ou importância. Desta forma, roteadores, switches, hubs e computadores podem manter as informações necessárias para o controle da rede. No entanto, mesmo com algumas facilidades, alguns equipamentos podem não suportar o protocolo, como por exemplo alguns hubs antigos.

O SNMP foi especificado pelo órgão IETF (The Internet Engineering Task Force) como

protocolo de gerência para redes TCP/IP, delegado pelo IAB (Internet Activities Board) [11]. Devido a sua simplicidade e implementações intensivas, o SNMP se tornou o padrão de facto. Uma versão posterior, denominada SNMPv2 foi desenvolvida com o propósito de desvincular o protocolo do padrão OSI, adicionando ainda algumas melhorias. Finalmente, o SNMPv3 veio desenvolver o maior avanço no protocolo, as características de segurança [49].

Neste modelo, um equipamento central, o gerente é responsável pelo controle de todas as informações de gerenciamento em tráfego na rede. O gerente é normalmente uma máquina de grande capacidade de computação e dedicada a este fim. Na periferia do modelo encontram-se os agentes SNMP, um para cada dispositivo.

As informações trocadas entre gerente e agente se encontram nas MIBs (Management Information Base)[45, 32], repositórios de dados que mantém informações atualizadas sobre o status de cada componente, segundo uma função preestabelecida pela ISO. Desta forma, hubs por exemplo, possuem tabelas em sua MIB com entradas de informações para seu controle, esta tabela em particular é conhecida como *Repeater MIB*, descrita pela RFC 1516 (Request For Comments) assim como os switches e roteadores possuem suas próprias tabelas.

Atualmente encontram-se disponíveis duas versões de MIBs. A MIB I [32], mais antiga, foi o primeiro projeto de repositório de informações de monitoramento para redes TCP/IP, seguindo o protocolo SNMP. A MIB II [44] agrega mais algumas informações consideradas indispensáveis, bem como estende seu uso para vários meios de transmissão.

O gerente troca informações constantemente com seus agentes através do protocolo seguindo basicamente os seguintes comandos:

- *GetRequest*: solicita informações contidas na MIB de um agente;
- *GetNextRequest*: solicita a próxima entrada da tabela tendo como referência o parâmetro solicitado no comando GetRequest;
- *GetResponse*: utilizado pelo agente para enviar as informações solicitadas;
- *SetRequest*: opera sobre a MIB alterando informações específicas;
- *Trap*: informações sobre status enviadas pelo agente na ocorrência de algum evento.

3.3.1 MIBs Utilizadas e suas funções

Como citado anteriormente, as MIBs são bases de dados contendo informações relevantes para a gerência de redes. Essas informações foram separadas por grupos segundo sua função. Cada

grupo por sua vez é dividido em objetos que contém propriamente a informação desejada.

Os grupos utilizados neste trabalho foram [32]:

- Grupo de Interfaces;
- Grupo Address Translation;
- Grupo IP;
- Grupo System;

além das Repeater MIB [34], Bridge MIB [15], RMON MIB [51].

Grupo de Interfaces

As interfaces de rede de um equipamento são os dispositivos de acesso ao meio físico. Elas definem basicamente as especificações de transmissão indicando, por exemplo, se o padrão é Ethernet, Token Ring, Token Bus, além de controlarem diversas variáveis como velocidade de transmissão, o endereço físico da interface (MAC do fabricante), controle de estatísticas como quantidade de octetos recebidos ou transmitidos, erros e comprimento da fila de pacotes. Além do controle estatístico, este grupo pode fornecer endereços MAC que posteriormente identificarão conexões físicas entre equipamentos.

O grupo IP

Este grupo contém todas as informações relativas ao endereçamento IP dos componentes de uma rede. É um grupo mandatório para qualquer sistema de gerência. Informações como tabelas de roteamento, tabelas de endereços IP, e todas os dados do fluxo de pacotes pela rede são configurados neste grupo. Ele identifica praticamente toda informação da *Camada de Rede* de uma rede TCP/IP.

O Grupo Address Translation

Este grupo é constituído de uma tabela de resolução de endereços, que faz a referência entre os endereços IP de um dispositivo (e suas interfaces) com seu endereço físico apropriado (MAC).

O grupo System

Mantém descrições mais informativas inseridas pelos administradores como o nome dado ao dispositivo, sua localização física e registro de tempo em que o sistema não é reiniciado.

Cada grupo e suas variáveis são listados na tabela abaixo:

Grupo de Interfaces	Grupo IP	Grupo Address Translation	Grupo System
ifIndex	ipAddrEntry	atEntry	sysUpTime
ifPhysAddress	ipAdEntAddr	atPhysAdress	sysName
	ipAdEntIfIndex	atNetAddress	
	IpRoutingTable		
	ipRouteDest		
	ipRouteIfIndex		
	ipRouteIfMetric1		
	ipRouteNextHop		
	ipRouteMask		

Tabela 3.1: Objetos de cada grupo das Mibs utilizadas

As outras MIBS utilizadas apresentadas na tabela 3.2 são otimizadas para cada dispositivo. A Repeater MIB, para hubs, Bridge MIB para switches e a Rmon MIB, uma base apropriada para monitoração remota que mantém estatísticas sobre cada equipamento.

Bridge MIB	Repeater MIB	RMON MIB
dot1dTpFdbAddress	rptrAddTrackPortIndex	etherHistoryIndex
dot1dTpFdbPort	rptrTrackLastSourceAddress	etherHistoryIntervalStart
dot1dTpFdbStatus	rptrAddrTrackSourceAddChanges	etherHistoryUtilization
	rptrTrackNewLastSrcAddress	

Tabela 3.2: Demais MIBS Utilizadas

Todos os campos coletados contribuem com informações para a descoberta dos dispositivos e topologia bem como para monitoração da performance na rede. O uso de cada item é descrito no capitulo sobre o sistema de diagnóstico.

3.3.2 Plataformas Comerciais

Em sua grande maioria, as plataformas de gerência de redes fazem uso das informações SNMP de gerenciamento tanto para a descoberta da rede como para a monitoração e diagnóstico.

Existem atualmente diversas plataformas de gerência de redes bastante abrangentes como por exemplo a HPOpenview e Netcool [39, 25, 26].

As plataformas são compostas por diversos sistemas independentes, cada qual responsável por uma tarefa na gerência da rede. Estas tarefas vão desde o gerenciamento de falhas até os mais altos níveis de controle de segurança e aplicações.

Um dos principais componentes da HPOpenview é o Network Node Manager (NNM). Além de servir como base de interface para vários outros sistemas o NNM é a ferramenta de gerência de falha e performance em redes de computadores.

Assim como em outras plataformas, o NNM segue uma arquitetura de funcionamento que vai desde a descoberta da rede, passa pela coleta de informações críticas de status dos dispositivos (poll), e faz o diagnóstico das falhas e problemas de performance.

Sob a mesma função do NNM, o sistema Precision do Netcool opera de forma semelhante. No entanto, ele incrementa o diagnóstico de falha e performance fazendo uma análise minuciosa do caminho em que a informação trafega, um paradigma muito próximo do proposto neste trabalho através do Raciocínio Baseado em Modelo.

3.3.3 Gerência de Falhas

A falha em uma rede de computadores é normalmente associada com mal funcionamento em um componente e conseqüente perda de conexão. A gerência de falhas envolve cinco passos [49]: (1) detecção de falha, (2) localização, (3) recomposição do serviço, (4) diagnóstico e (5) resolução do problema.

A detecção de falhas acontece via plataformas de gerenciamento ou controles como o polling. A localização da falha envolve o processo de identificar onde o problema ocorreu. Uma vez localizado o problema, é primordial restaurar o serviço para o usuário. O passo do diagnóstico é bastante delicado, envolve várias disciplinas e foi abordado por vários pesquisadores como mencionado no capítulo anterior. Finalmente é feita a resolução do problema, seguida da geração de um relatório (*Trouble Ticket*).

O sistema desenvolvido neste trabalho dá ênfase ao processo de detecção de falha e sua localização. No primeiro é usado um esquema de polling (consultas periódicas a elementos da rede) ou a geração de *Traps* (os agentes de gerenciamento enviam valores sobre seu status). Uma aplicação envia comandos *ping* periodicamente. Uma conexão é considerada falha após um limite de tempo sem resposta. Na localização, a fonte do problema pode ser procurada caminhando-se pela topologia. A solução ideal para a localização e isolamento de uma falha é

a aplicação de uma técnica de inteligência artificial [49]. Através da observação dos sintomas é possível identificar a origem do problema.

3.3.4 Gerência de Performance

O propósito da gerência de performance é monitorar a rede para obter estatísticas que indiquem o bom funcionamento da mesma, caso contrário diagnosticá-la e repará-la. O bom funcionamento é avaliado pela taxa de transmissão, disponibilidade da rede, tempo de resposta e confiabilidade.

Estas estatísticas podem ser obtidas através da análise do meio físico com analisadores de protocolo ou ainda através de variáveis mantidas em cada dispositivo. As medidas de tráfego numa LAN Ethernet são a base de uma gerência de performance. Parâmetros como a taxa de utilização do meio físico, o número de colisões, quantidade de pacotes saindo ou entrado em uma interface, a quantidade de pacotes perdidos e a latência de um segmento de rede são exemplos destas estatísticas. Todas elas podem ser encontradas nas MIB dos dispositivos [51, 34]. A MIB RMON é um exemplo de base de informações sobre performance. Esta MIB faz a monitoração remota ou local de um segmento de rede mantendo atualizas as informações de utilização da rede, número de colisões, perda de pacotes dentre outros e mantendo um histórico em intervalos de tempo predefinidos realizando uma monitoração perene.

A análise das variáveis de performance em uma rede depende de vários fatores como a quantidade de terminais em um segmento, o horário de uso durante o dia, os recursos de rede utilizados até o tipo de aplicações usadas [49, 9]. Cada rede possui sua característica, o que torna empírica a descoberta destas variáveis. São denominados limites ("threshold"), os valores que delimitam variáveis como a taxa de utilização do meio, a quantidade de perda de pacotes, a latência de um segmento e ainda muitos outros encontrados nas MIB RMON.

3.4 Sumário

As redes locais de computadores (LANs) são extremamente difundidas pelo mundo atualmente. Baseiam-se no modelo OSI de referência como forma de padronização e utilizam em sua grande maioria os protocolos TCP/IP.

O TCP/IP, protocolo em uso na Internet fornece meios de se identificar dispositivos pelo mundo todo e transportar dados entre os mesmos. Através de seu sistema global de endereçamento é possível computar o caminho pelo qual um pacote irá trafegar. A complexidade deste protocolo vai bem além disso, resume-se aqui sua principal função neste trabalho.

As redes locais são compostas por dispositivos com diferentes funcionalidades. Em sua maioria estes dispositivos podem ser monitorados através de algum protocolo de gerência de redes.

O SNMP é um destes protocolos, que realiza esta função através da manutenção de uma MIB em cada dispositivo. A função e gerenciar uma rede envolve a monitoração contínua dos seus elementos, afim de oferecer uma alta qualidade de serviço, confiabilidade e disponibilidade da rede. É portanto uma tarefa primordial.

Segundo estrutura desenvolvida pela ISO a gerência de redes se divide em cinco áreas principais, donde destacam-se para este trabalho a gerência de falha e a gerência de performance.

O sistema aqui desenvolvido situa-se no escopo descrito neste capítulo, os detalhes de implementação e cada item aprofundado encontram-se no capítulo sobre o sistema.

Capítulo 4

O Sistema de Diagnóstico de Redes Locais Baseado em Modelo

O Sistema de Diagnóstico de Redes Locais Baseado em Modelo trata da aplicação da técnica de Raciocínio Baseado em Modelo de Funcionamento Correto na localização e diagnóstico de falhas e performance em Redes Locais de Computadores baseadas no protocolo TCP/IP.

Este sistema, desenvolvido nas linguagens JAVA e PROLOG, abrange de forma geral vários aspectos da gerência de redes como a sua descoberta, monitoração e o diagnóstico de falhas na mesma.

Dentre as cinco áreas de gerência propostas pela ISO [49], o sistema aborda as gerências de falha e performance, através de procedimentos de detecção, localização e diagnóstico de problemas. Estas funções tem como referência um modelo lógico e funcional construído especialmente para representar o funcionamento normal da rede, de forma a auxiliar na tomada de decisão.

Este capítulo apresenta em detalhes toda a implementação deste sistema. Suas seções dividem-se de acordo com cada subsistema. Uma seção introdutória apresenta os aspectos gerais do sistema, as seções posteriores mostram o mecanismo de descoberta implementado, a configuração e diagnóstico do modelo, a monitoração da rede e finalmente o sistema central de diagnóstico.

4.1 Visão Geral do Sistema

O *Sistema de Diagnóstico Baseado em Modelo* (SDBM) é composto por quatro blocos responsáveis por: (1) descoberta, (2) diagnóstico de configuração do modelo, (3) monitoração (detecção), (4) localização e diagnóstico, abrangendo de forma geral vários aspectos da gerência de redes.

Estes blocos também chamados de subsistemas, possuem uma grande sinergia tendo em comum o modelo lógico e funcional utilizado para representar a rede. Aos quatro subsistemas foram atribuídos os seguintes nomes:

1. Subsistema de Descoberta da Rede: SDR;
2. Subsistema de Configuração do Modelo: SCM;
3. Subsistema de Coleta de Status: SCS e
4. Sistema Central de Diagnóstico: SCD.

O modo de funcionamento destes subsistemas permite dividi-los em duas partes, a primeira toda voltada para a construção do modelo lógico-funcional baseada em informações de topologia e configuração da rede e a segunda que infere sobre o modelo comparando-o com as observações coletadas da rede. O SDR e o SCM pertencem a primeira divisão, enquanto os demais subsistemas à segunda.

A forma de interação destas partes com a rede real fora descrita anteriormente [6, 5] e nomeada como modo off-line para a primeira, onde a interação com a rede se resume à captura de informações sobre topologia e configuração; e online para a segunda, onde a interação é constante e relacionada a coleta de informações de status da rede.

A divisão do SDBM e seus módulos é ilustrada na figura 4.1.

De modo geral o funcionamento do SDBM segue a representação da figura 4.1. Todas as informações estruturais e funcionais sobre a rede são capturadas pelo SDR e depuradas para a construção do modelo lógico que posteriormente passa pelo crivo de um subsistema de diagnóstico e configuração do modelo SCM. Após a obtenção de um modelo suficientemente completo e com nível de abstração aceitável [17, 6], entra em ação o subsistema de coleta de status (SCS) que monitora ciclicamente a rede notificando as observações encontradas para o sistema central

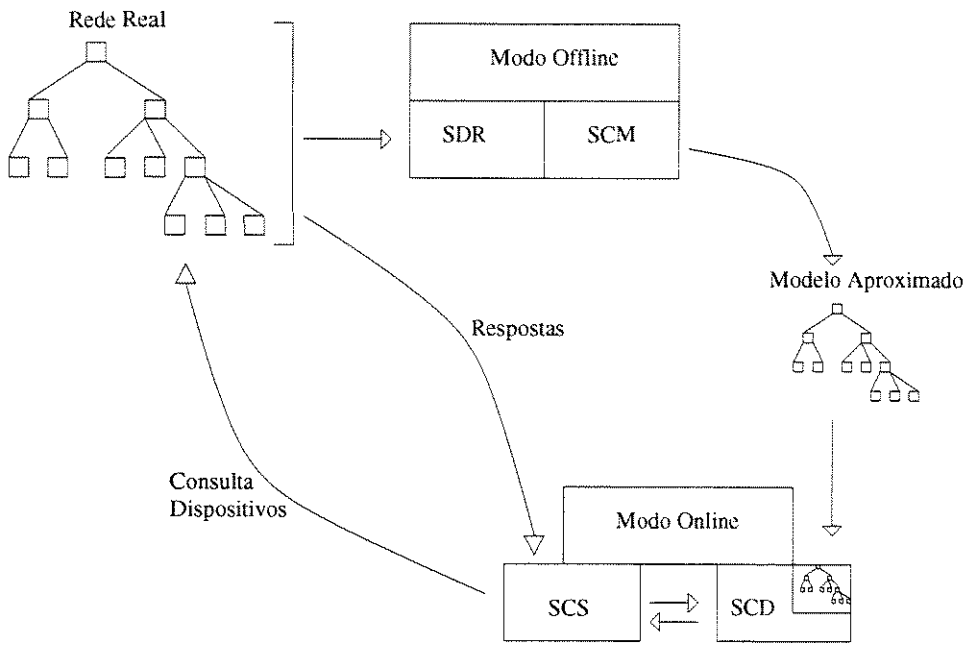


Figura 4.1: Arquitetura do Sistema de Diagnóstico Baseado em Modelo:SDBM

de diagnóstico (SCD), que no caso de uma falha [1] utiliza o raciocínio baseado em modelo para chegar a um diagnóstico plausível.

O sistema passa uma vez pelo modo off-line e itera indefinidamente no modo online. Cada subsistema é descrito em detalhes a seguir.

4.2 Subsistema de Descoberta da Rede - SDR

O processo de descoberta de uma rede é um mecanismo semi-automatizado que busca coletar todos os tipos de informação como a identidade dos equipamentos que a compõem, sua disposição física e lógica, seu esquema de endereçamento, os tipos de conexão e roteamento dentre outras.

A finalidade deste procedimento é a de elaborar uma representação lógica de toda a rede, mapeando sua topologia. Isto é realizado integrando-se informações coletadas através de pings nos equipamentos e dos protocolos de gerenciamento como o SNMP com informações presentes nas MIBs.

[1] Assume-se neste capítulo o termo "falha" de forma genérica para abstrair tanto uma falha de comunicação quanto um problema de performance.

Este procedimento é padrão em várias plataformas de gerenciamento, tendo sido nomeado "network discovery" e possuindo atenção especial de pesquisadores devido a sua importância [33]. Em geral estas plataformas tomam por base o modelo de referência OSI e fazem a descoberta por camadas.

Esta tarefa, porém, é complexa. A descoberta da rede depende da boa configuração dos equipamentos. Dependendo do quanto se deseja aprofundar na descoberta da rede, as informações de gerenciamento são fundamentais. Em sua forma mais simples, por exemplo, o procedimento lista os dispositivos presentes na rede, mas não suas conexões - para isso é necessário aprofundar-se até a camada de enlace (OSI).

De fato, muitas informações relevantes podem não estar disponíveis na rede, ou até mesmo dispositivos em modo de operação normal podem não responder a alguns comandos. A rede descoberta portanto, é passível de falhas, falta de equipamentos ou conexões, o que caracteriza este procedimento como semi-automatizado. Um exemplo bastante típico deste problema são os hubs, simples repetidores muitos dos quais não possuem informações de gerenciamento, sequer um endereço IP. Isto os torna invisíveis para um mecanismo de descoberta.

Sistemas como o HPOpenview [39] e Netcool [25, 26], possuem módulos próprios para a descoberta da rede e a fazem de forma bastante detalhada. No segundo, por exemplo, o módulo leva aproximadamente seis horas para descobrir uma rede até a Camada de Rede (OSI), com aproximadamente 80 dispositivos. Em geral, essas plataformas também enfrentam problemas na coleta de informações importantes para um sistema de diagnóstico como conexões e interfaces físicas, o que torna necessária a inserção manual de algumas delas.

Considerando-se o tamanho da rede a ser modelada, a tarefa de coleta de informações torna-se longa e exaustiva, além de gerar na rede uma descarga de informações de gerenciamento, elevando consideravelmente a utilização da mesma. Por isto, em geral é sugerido que o procedimento seja realizado em períodos noturnos, evitando assim distúrbios na rede.

O *Subsistema de Descoberta da Rede* (SDR) é o principal responsável pela construção do modelo lógico-funcional da rede TCP/IP a ser gerenciada. Em um primeiro passo o SDR procura pelos dispositivos presentes na rede e posteriormente começa uma busca exaustiva por informações das MIBs daqueles que respondem por comandos do protocolo SNMP e as organiza em arquivos distintos.

Os passos subseqüentes, descritos em detalhe a seguir, resumem-se a uma depuração minuciosa destas informações que vai construindo passo a passo o modelo, desde a camada física até a camada de rede tendo como focos a representação simples porém fiel da estrutura, da lógica e

do funcionamento da rede real.

Como ferramentas de auxílio no processo de descoberta o SDR utiliza os comandos *ICMP-echo* [41], mais conhecido por ping, e o *Traceroute*. O primeiro utiliza um endereço IP para consultar o status de uma estação. Este comando permite identificar um elemento da rede como ativo ou não. O Traceroute encontra os roteadores entre um IP de origem e um destino.

Dentre as informações coletadas estão aquelas encontradas nas MIBs MIB-I [32], MIB-II [44], Bridge MIB [15], RMON MIB [51] e Repeater MIB [34] como descrito nas tabelas do capítulo anterior.

Como nem todos os equipamentos poderão ter suas MIBs configuradas ou ainda informações atualizadas, e devido a dependência do processo de descoberta do protocolo SNMP, assume-se neste trabalho um modelo aproximado da rede, aceitando-se a inserção manual de informações imprescindíveis.

Todo o processo de interação com a rede real do SDR foi desenvolvido em JAVA por um administrador da rede LSI-USP. A depuração das informações e a construção do modelo da rede foi desenvolvido em PROLOG, assim como o restante do sistema.

Em detalhe, os passos deste módulo são três:

1. Ping broadcast e pings específicos, a fim de formar uma lista com todos os dispositivos presentes na rede;

2. Busca pelas informações disponíveis nos agentes SNMP dos dispositivos encontrados;

3. Organização das informações e construção do modelo.

O primeiro passo se baseia no fato de que cada elemento ativo na rede que possua um endereço IP configurado é capaz de responder ao comando ping através de um pacote *echo-reply*, informando sobre seu status e confirmando seu endereço IP. Através de um *ping* para o endereço broadcast da rede, uma estação com direitos de administrador consegue receber resposta de todos os elementos ativos na rede. Como os roteadores em geral impedem a transmissão de broadcast, os pings específicos são utilizados para alcançar elementos através dos roteadores. Este primeiro passo agrupa todas as respostas em uma lista de IPs ativos, é um procedimento necessário para levantar quais são os equipamentos presentes na rede e a partir disto o modelo da rede já conhece os dispositivos que a irão compor.

Através desta lista de IPs e de forma exaustiva, o SDR passa a se comunicar com cada dispositivo através dos comandos do protocolo SNMP [11] para a obtenção das informações contidas

nas MIBs. Estas informações são separadas por rede e por tipo de equipamento (hub, switch, roteador) em arquivos específicos. As MIBs dos roteadores e switches permitem a construção da topologia aproximada da rede através das conexões especificadas em suas interfaces e as informações de roteamento dão uma primeira noção do funcionamento da rede. Faltam ainda as informações de conexões relacionadas aos hubs, que dificilmente são obtidas automaticamente quando os hubs não são gerenciáveis (o que é comum em muitas LANs) ou quando outros dispositivos não implementam propriamente suas MIBs. Nesta rede incompleta, não é possível distinguir automaticamente a topologia de um domínio de colisão, mas apenas os equipamentos que pertencem ao mesmo. Informações fundamentais ausentes são adicionadas manualmente para completar o modelo.

No terceiro e último passo, o sistema faz a depuração (parsing) das informações para cláusulas na linguagem PROLOG como descrito nas subseções a seguir, fazendo uma analogia às camadas do modelo de referência OSI. Estas cláusulas são o modelo lógico-funcional que passa a ser utilizado em todo o restante do sistema como referência do funcionamento correto da rede. A topologia, o esquema de endereçamento, as tabelas de roteamento e identificação dos dispositivos estão descritas por estas cláusulas.

Construção da camada de rede

O primeiro passo disposto acima já oferece de modo genérico o endereço IP de todos os dispositivos ativos na rede. Através do protocolo SNMP as tabelas *IpRouteTable* são capturadas, possibilitando a construção lógica das tabelas de roteamento, e dando ao sistema a idéia de como os pacotes IP deverão trafegar. Os pares IP/Máscara encontrados permitem a montagem do esquema de endereçamento da rede. A depuração (parsing) da tabela leva à construção de cláusulas PROLOG como exemplificado abaixo:

A entrada do Grupo IP da MIB-I [32]

<ipdevice> 10.0.10.157
<ENTRY>
<ipRouteDest> 0.0.0.0
<ipRouteIfIndex> 3
<ipRouteMetric1> 1
<ipRouteMetric2> -1

<ipRouteMetric3> -1
<ipRouteMetric4> -1
<ipRouteNextHop> 10.0.10.1
<ipRouteType> indirect (4)
<ipRouteProto> local (2)
<ipRouteAge> 0
<ipRouteMask> 0.0.0.0
<ipRouteMetric5> -1
<ipRouteInfo> 3
</ENTRY>

gera a cláusula:

rotax(ip(10, 0, 10, 157), ip(0, 0, 0, 0), mask(0, 0, 0, 0), ip(10, 0,10, 1), 1, 3).

onde os campos deste predicado indicam:

rotax(IP Origem, IP Destino, Máscara de rede, Próximo Hop, Métrica, Interface)

A cláusula *dev_ip(Dev1, ip(10,0,10,157), net1)* indica que o dispositivo Dev1 cujo endereço IP é 10.0.10.157 está presente e encontra-se na rede 1.

Uma grande quantidade destas cláusulas é gerada (uma para cada entrada da tabela de roteamento de cada dispositivo presente na rede). Estas cláusulas junto das identificação dos dispositivos, suas conexões e o esquema de endereçamento da rede (subredes e máscaras) permite computar o caminho previsto para o tráfego de um pacote, a principal informação funcional do modelo.

Associação com informações de enlace

Posteriormente o grupo de interfaces e "address translation" mostram a associação dos endereços IP com seus respectivos endereços físicos, e ainda a distribuição das interfaces de um dispositivo. Novamente há uma tradução para cláusulas PROLOG:

A seguinte entrada da tabela *IpNetToMedia* da MIB-II [44]

<ipdevice> 10.0.10.157
<ENTRY>
<ipNetToMediaIfIndex> 3
<ipNetToMediaPhysAddress> 0 60 67 30 db ce
<ipNetToMediaNetAddress> 10.0.10.1
<ipNetToMediaType> dynamic (3)
</ENTRY>

É traduzida para a cláusula:

ipmac(ip(10, 0, 10, 157), [0, 60, 67, 30, 'db', ce]).

que simplesmente associa o endereço IP do dispositivo, com seu endereço físico (MAC). As tabelas do grupo de interfaces da MIB-I complementam a informação associando o endereço físico ao número da interface do equipamento ao qual está atrelada.

Dado o fato de que nem todos os dispositivos possuem suas entradas atualizadas na tabela *IpNetToMedia*, muitos deles não possuem toda especificação de suas diferentes interfaces. Informações complementares podem ser obtidas através da intersecção com outras tabelas das MIBs. A tabela *ifTable* fornece o endereço MAC de todas as interfaces de um dispositivo (incluindo o tipo de interface) que quando associadas à tabela *IpNetToMedia* e com a *Bridge MIB* podem fornecer por completo os endereços MAC encontrados em um único equipamento associados às suas interfaces. Este passo apresenta como resultado a representação dos elementos da rede de forma mais concisa, mostrando suas interfaces, seus endereços físicos e IP.

Construção da camada física

Finalmente a construção da camada física estabelece as conexões entre equipamentos e suas identidades (estações, hubs, switches ou roteadores), o que permite mostrar a topologia de forma mais consistente e dividi-la propriamente em domínios de colisão e broadcast.

A modelagem das conexões entre equipamentos é uma tarefa minuciosa, pois, as informações não estão explicitadas em tabelas das MIBs. Cabos e muitos repetidores (hubs) não são gerenciáveis e é preciso analisar outras informações (como as capturadas até então) para chegar a uma conexão física.

A Bridge MIB, através da tabela *dot1dTpFdb* permite associar um elemento da rede (através de seu endereço físico) a uma porta do switch, por exemplo. Isto não significa porém uma conexão física fim a fim, mas dá uma primeira distribuição espacial dos elementos.

É através das informações contidas nas MIBs dos repetidores [34] que as principais informações sobre conexão física podem ser obtidas e isto também é feito de forma minuciosa. Os seguintes campos desta MIB são utilizados:

- <ipdevice>, endereço IP do dispositivo;
- <rptrAddrTrackPortIndex>, a porta do dispositivo que contém as informações abaixo;
- <rptrAddrTrackLastSourceAddress>, endereço MAC de origem do último pacote que passou por esta porta e
- <rptrAddrTrackSourceAddrChanges>, quantas mudanças de endereço MAC de origem aconteceram.

Estas tabelas são exclusivas dos Hubs. De forma geral, cada entrada desta tabela indica, por porta, o endereço MAC do último pacote que passou pelo hub.

O campo *<rptrAddrTrackSourceAddrChanges>* conta quantas vezes houve mudança no endereço MAC de origem dos pacotes que passaram por esta porta. Quando este número é zero, pode-se concluir que o dispositivo com este endereço está diretamente conectado ao Hub por esta porta. Caso seja maior do que zero, pode-se pelo menos inferir que existe uma conexão, ainda que indireta que liga este dispositivo ao Hub, é o caso dos hubs cascateados. Através destes valores de MAC e número de mudanças, o SDR gera cláusulas da forma:

con(IP do dispositivo, Número de Mudanças, MAC do último pacote, Porta)

Ao cruzar esta cláusula com as obtidas através das informações de enlace (par IP/MAC, por exemplo) o SDR modela uma conexão física.

As informações das MIBs para a construção da topologia dependem de uma configuração apurada dos equipamentos e a manutenção constante das tabelas frente as modificações na rede. O estabelecimento de uma conexão física entre dispositivos de forma automática só é possível quando os hubs existentes num domínio de colisão são gerenciáveis. Os hubs não gerenciáveis são pontos críticos para esta modelagem, pois são transparentes para os protocolos da rede.

Neste ponto, se faz necessária a inserção de informações manualmente, que completem as conexões. Plataformas de gerenciamento como o HPOpenview [39] possuem ferramentas para a inserção manual destas informações.

Como resultado de todo este processo minucioso de coleta e análise de informações e com a inserção de outras manualmente, o SDR gera um modelo lógico-funcional da rede constituído de cláusulas PROLOG. Este modelo contém desde a topologia da rede, incluindo as conexões físicas entre elementos até o esquema de endereçamento e roteamento de pacotes IP. Como as redes em geral são passíveis de falhas e de muitos erros de configuração, o SCM faz em um passo posterior ao SDR, uma crítica sobre o modelo obtido, levantando os principais problemas encontrados.

4.3 Subsistema de Configuração do Modelo - SCM

Um dos items sugeridos no modelo de gerência OSI é a *Gerência de Configuração*, normalmente utilizada no contexto de descoberta de topologia, mapeamento da rede e configuração de parâmetros nos agentes e sistemas de gerenciamento sendo responsável por manter atualizadas as informações contidas na rede [49].

O *Subsistema de Configuração do Modelo* (SCM) faz uma análise da qualidade do modelo gerado pelo SDR baseada no conhecimento de que alguns erros de configuração são bastantes comuns e bem conhecidos por administradores de rede.

O SCM procura por informações contraditórias, freqüentemente encontradas em redes TCP/IP devido seu caráter mutável. As mudanças constantes rendem a rede duplicações ou falta de endereços IP, entradas inconsistentes nas tabelas de roteamento e sistema de nomes também inconsistentes dentre outros. O objetivo do SCM é gerar um modelo livre de erros, suficientemente consistente para o posterior diagnóstico.

Seis principais testes são realizados sobre o modelo da rede:

1. Verifica dispositivos sem identificação: alguns dos dispositivos da rede podem ter endereços atribuídos (IP ou MAC), mas não possuir nome (caso o sistema de nomes esteja disponível na rede) ou uma identificação própria no modelo lógico da rede;

2. Verifica ambiguidade de nomes: caso o mesmo dispositivo tenha dois nomes ou identificações distintos um deles será removido;

3. Verifica se dois elementos têm o mesmo IP: avalia a redundância de endereços IP na rede;

4. Verifica se os elementos estão e redes definidas: segundo o endereço e máscara das redes e sub-redes, este teste verifica se todos os endereços IP coletados estão num domínio válido para esta rede;

5. Verifica hubs repetidos: algumas informações do processo de descoberta podem estar duplicadas para os hubs, sendo necessária esta correção;

6. Verifica conexões repetidas: caso a mesma conexão seja obtida por modos distintos, uma delas é automaticamente eliminada.

Por ser aplicado somente ao modelo da rede, o SCM não deve ser considerado um gerente de configuração, mas, um passo agregado ao modo off-line. Com aprimoramentos adequados, este módulo poderá ser utilizado como referência para um administrador verificar os problemas de configuração de sua rede e até mesmo fazer diagnósticos de tempos em tempos na rede, levantando mais informações.

Ao executar o SDR e o SCM o sistema cumpre sua primeira etapa como descrito na figura 4.1. Como resultado desta primeira etapa, o modelo lógico-funcional da rede está construído e corrigido. O modelo passa a ser o ponto de referência para os subsistemas seguintes.

4.4 Subsistema de Coleta de Status - SCS

O *Subsistema de Coleta de Status* SCS, é o responsável pela monitoração da rede. Tendo por base o modelo coletado, este subsistema decide quais elementos serão monitorados e em que seqüência, suprindo constantemente o subsistema de diagnóstico (SCD) com as observações encontradas, além de ser a interface entre a rede real e o SCD. Estes dois subsistemas (separados na parte online do sistema) trabalham de forma conjunta em grande sinergia, diferente dos primeiros subsistemas que tinham suas execuções em tempos separados.

O SCS realiza sua tarefa através de um procedimento conhecido por *polling*, que é a monitoração cíclica dos dispositivos da rede de uma forma gerente-cliente. O ping é o principal agente do polling. A estação gerente da rede envia pings constantemente para os elementos (clientes), aguarda e analisa suas respostas. O polling testa a conectividade dos equipamentos e ainda algumas estatísticas de performance, pois o ping retorna como resultado uma latência e o percentual de perda de pacotes. As observações encontradas (como a perda de sinal, latência alta ou perda de pacotes) são reportadas imediatamente para o SCD junto com o endereço IP do dispositivo onde estas observações foram detectadas.

O polling é uma tarefa cíclica e exaustiva. Para evitar uma sobrecarga de informações de gerência na rede alguns pesquisadores sugerem (como discutido a seguir), que o polling da rede aconteça de forma inteligente. Antes de iniciar o processo de monitoração o SCS faz o cômputo da seqüencia dos equipamento que irá monitorar. Neste trabalho, uma heurística foi criada para minimizar o impacto do polling na rede.

4.4.1 Heurística para a seqüência de polling

Diversos pesquisadores na área de gerenciamento de redes [49, 52, 46], atentam para o problema da sobrecarga da taxa de utilização de um domínio de colisão devido a um excessivo monitoramento através de ferramentas de gerência.

Normalmente estas ferramentas realizam vários testes ciclicamente e em intervalos muito curtos de tempo, iterando os testes para cada dispositivo na rede (polling). Um ciclo de polling é a execução de um comando ping para cada elemento numa lista de elementos monitorados. O período de um ciclo de polling é normalmente regido pelo tempo gasto com os pings, no entanto algumas heurísticas usam as estatísticas de utilização da rede para definir um melhor intervalo, mudando este período [52]. Se o período for muito longo, as observações podem não refletir o estado real da rede, se for muito curto, pode haver uma sobrecarga de informações de poll, rendendo uma taxa de utilização elevada.

O SCS utiliza o modelo da rede para verificar quais elementos são chaves para serem monitorados. Estes elementos são normalmente os pontos finais do caminho entre a máquina de gerência e um elemento, cobrindo a monitoração de vários outros elementos. Como técnica para tornar mais inteligente o processo de polling, adotou-se neste trabalho uma heurística de embaralhamento de elementos para a obtenção da seqüência. Este algoritmo busca selecionar e ordenar os dispositivos de forma a aumentar o tempo entre pings num mesmo domínio de colisão.

A partir do modelo da rede, é formada uma árvore que têm como raíz a estação gerente como mostrado na figura 4.2. Como os pings acontecem a partir da raíz e conhecendo o caminho pelo qual o pacote irá passar, só é necessário monitorar os elementos folha desta árvore, pois caso o pacote não chegue, sabe-se que o problema está em algum elemento neste caminho e o sistema passa para o processo de diagnóstico da falha (SCD).

Como os nós folha desta árvore são normalmente computadores pessoais, há uma maior probabilidade de que eles estejam desligados em algum momento. Para isso uma poda nesta árvore é feita, mantendo somente elementos que estarão respondendo ao ping. Os computadores

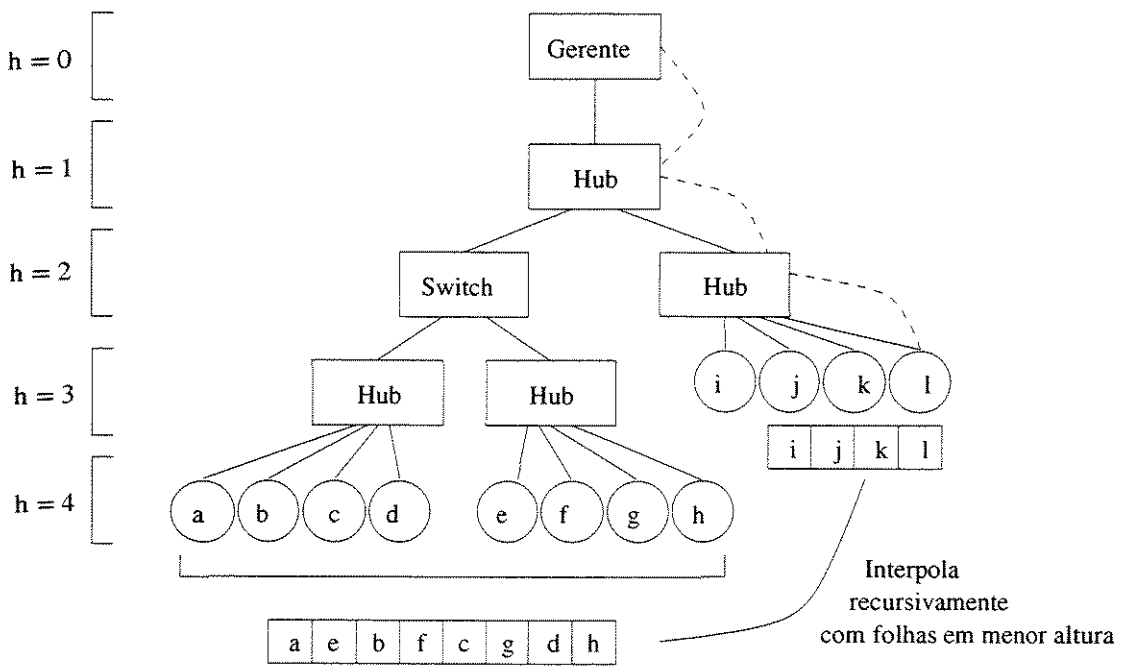


Figura 4.2: Interpolação de elementos de domínios de colisão. As linhas pontilhadas indicam o caminho que um pacote percorre para chegar em um nó folha, o que indica que os elementos internos desse caminho (os HUBs neste caso) não precisam ser consultados durante o ciclo de polling.

que poderão estar desligados são listados previamente no modelo.

Pela forma como a árvore é construída, as folhas irmãs são dispositivos de um mesmo domínio de colisão. O algoritmo proposto interpola de forma recursiva duas listas de elementos folha que estejam na maior altura da árvore. O resultado é então interpolado com outra lista de elementos da mesma altura ou menor, até que todas as folhas estejam interpoladas (ver figura 4.2).

Desta forma, como os elementos folha de um mesmo ramo da árvore pertencem ao mesmo domínio de colisão, intercalando-se diferentes ramos o tempo entre pings num mesmo domínio de colisão aumenta, diminuindo assim a sobrecarga de informações de gerência no mesmo domínio.

A interpolação de duas listas segue uma heurística de maior distância entre elementos irmãos como mostrado na figura 4.3 através do seguinte algoritmo:

Algorithm 1 Algoritmo para interpolação de duas listas

mergelists($Lista1$,$Lista2$,$Resultado$)
    N = tamanho da $Lista1$
    K = tamanho da $Lista2$
    $Mod = N$ mod K
    $Z = N$ // K
    if (Mod == 0) then
        $Resultado$ = intercala(Z,$Lista1$,$Lista2$)
    else
        if (Z = 0) then
            $X = N$ // Mod
            $Resultado$ = intercala(X,$Lista2$,$Lista1$)
        else
            $Temp$ = intercala(Z,$Lista1$,$Lista2$)
            $Modulo = Mod$ últimos elementos de $Temp$
            $Temp = Temp$ - $Modulo$
            W = tamanho de $Temp$
            $Y = W$ // Mod
            $Resultado$ = intercala(Y,$Temp$,$Mdulo$)

Onde *intercala(Z,L,M)* é a função que intercala Z elementos da lista L com a lista M.

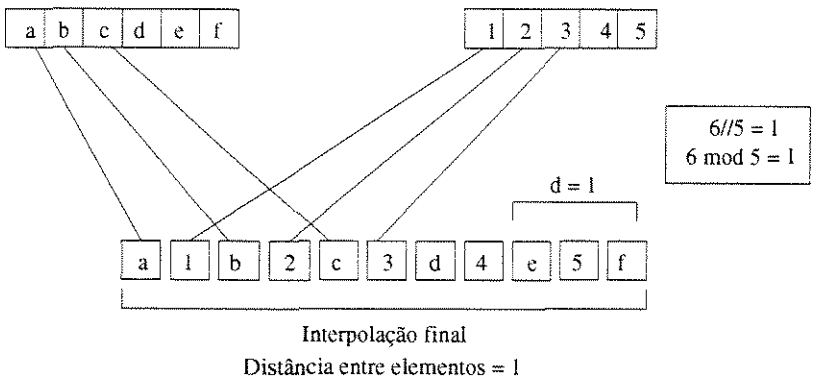


Figura 4.3: Cômputo da maior distância entre elementos de um mesmo domínio de colisão

A adoção desta heurística como forma de polling inteligente baseia-se na sua simplicidade e na sua característica estática, ou seja, uma vez gerada a seqüência ela não deverá ser alterada para os próximos ciclos, a menos que o modelo da rede seja alterado.

Tanto na construção da seqüência de polling quanto no subsistema central de diagnóstico o sistema verifica se os elementos escolhidos possuem agentes SNMP. Novamente o modelo lógico-funcional da rede auxilia no processo de escolha de um representante válido caso algum elemento não possua agente, de forma que se consultado, este representante possa ajudar no processo de diagnóstico (ver figura 4.4).

Através da seqüência de polling, o SCS fica monitorando a rede indefinidamente. Juntamente com o SCD, um controle é feito sobre a lista de polling para que, quando uma falha é diagnosticada, o elemento falho saia da lista por sessenta ciclos, para evitar que o sistema entre sempre no subsistema de diagnóstico para um mesmo problema.

As três observações seguintes ativam o subsistema de diagnóstico - (1) *Perda de Sinal*: a latência do ping excede o limite máximo preestabelecido (ping timeout); (2) *Degradação de Performance*: a latência excede um limite padrão (threshold) associado a um domínio de colisão [2]; e (3) *Alta perda de Pacotes*: uma das estatísticas do ping, que ao superar um limite preestabelecido também invoca o diagnóstico.

Até que o SCS encontre a primeira observação, ele opera de modo autônomo seguindo a seqüência de polling. Porém, ao entrar em estado de diagnóstico segundo as premissas acima, o SCS suspende o polling e o SCD assume o controle dos pings. Ao se encerrar o estado de diagnóstico o SCS reassume, a partir do ponto em que parou.

[2] O capítulo seguinte discute os aspectos de como se obter um valor padrão.

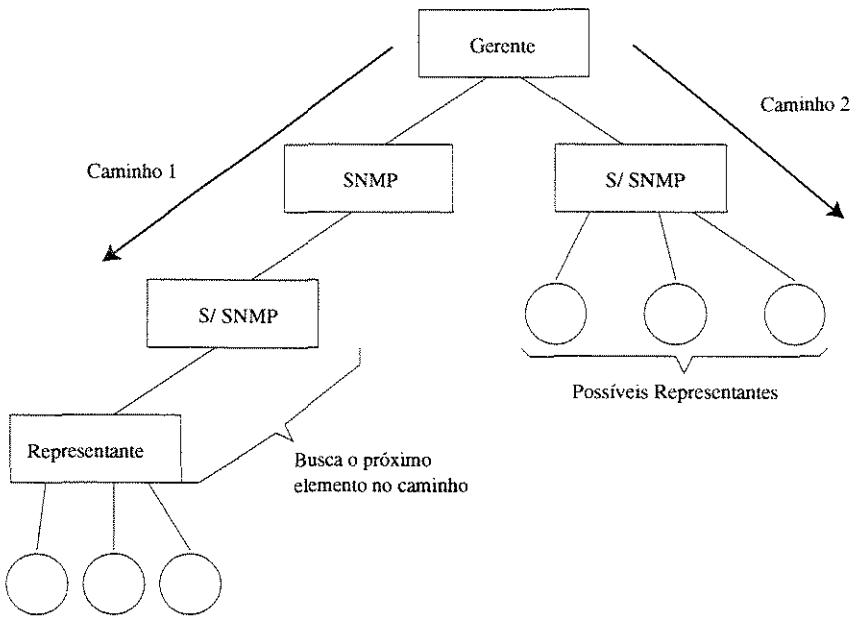


Figura 4.4: Busca por um representante válido para o polling

4.5 Sistema Central de Diagnóstico - SCD

O *Sistema Central de Diagnóstico* (SCD), é responsável por inferir sobre as observações que coleta da rede, analisando-as junto ao modelo lógico-funcional e fazendo novas consultas na rede real até chegar ao ponto exato onde a falha ocorreu gerando como solução para o diagnóstico um conjunto de causas que explicam os sintomas encontrados.

O SCD possui dois procedimentos distintos, um para o diagnóstico de falhas de comunicação e o outro para problemas de performance, porém, ambos raciocinando da mesma forma sobre o modelo, ou seja, procurando as disparidades entre o modelo de funcionamento correto e as observações encontradas.

O ponto chave do diagnóstico baseado em modelo de funcionamento correto para uma rede local é conhecer o caminho pelo qual um pacote irá trafegar. Com este conhecimento, com o restante das informações que o modelo pode prover e ainda conhecendo as observações, o SCD executa uma série de verificações na rede afim de chegar a um diagnóstico plausível que é neste caso o menor conjunto de componentes da rede que, quando em mal funcionamento, podem provocar a falha encontrada explicando assim o sintoma. Assume-se neste trabalho o critério de falha única, ou seja, apenas uma causa é responsável pelas observações encontradas, não

ocorrendo duas causas ao mesmo tempo [4, 5, 40].

Para compreender o estado da rede, o SCD depende das informações fornecidas pelo SCS, por isto operam em grande sinergia. Existem várias formas de interação entre um sistema de diagnóstico e a rede em si [6]. As formas passivas (*Passivo* e *Passivo por Transição*) basicamente aguardam e analisam todas as informações vindas da rede mesmo durante o processo de diagnóstico. Elas aguardam a descarga de alarmes ou eventos gerados e os interpreta para chegar a uma solução.

A forma de interação *Ativa*, como a adotada neste trabalho, garante ao sistema maior autonomia. Durante o processo de diagnóstico o SCD solicita as informações que lhe convierem, tendo a capacidade de tomar decisões, quais informações descartar e solicitar outras se necessário. Esta característica pode ser bastante conveniente frente à grande quantidade de alarmes gerados por uma mesma falha. Esta forma é apropriada para um sistema de diagnóstico baseado em modelo pois permite que o mesmo use de forma autônoma o modelo da rede para realizar suas consultas. Posteriormente, auxilia no processo de filtragem dos alarmes, pois, o sistema analisa o estado da rede antes de atentar para a descarga de alarmes gerados [27, 24]. Desta forma, o SCD é responsável por analisar os pontos que considera críticos para se chegar ao diagnóstico mais preciso.

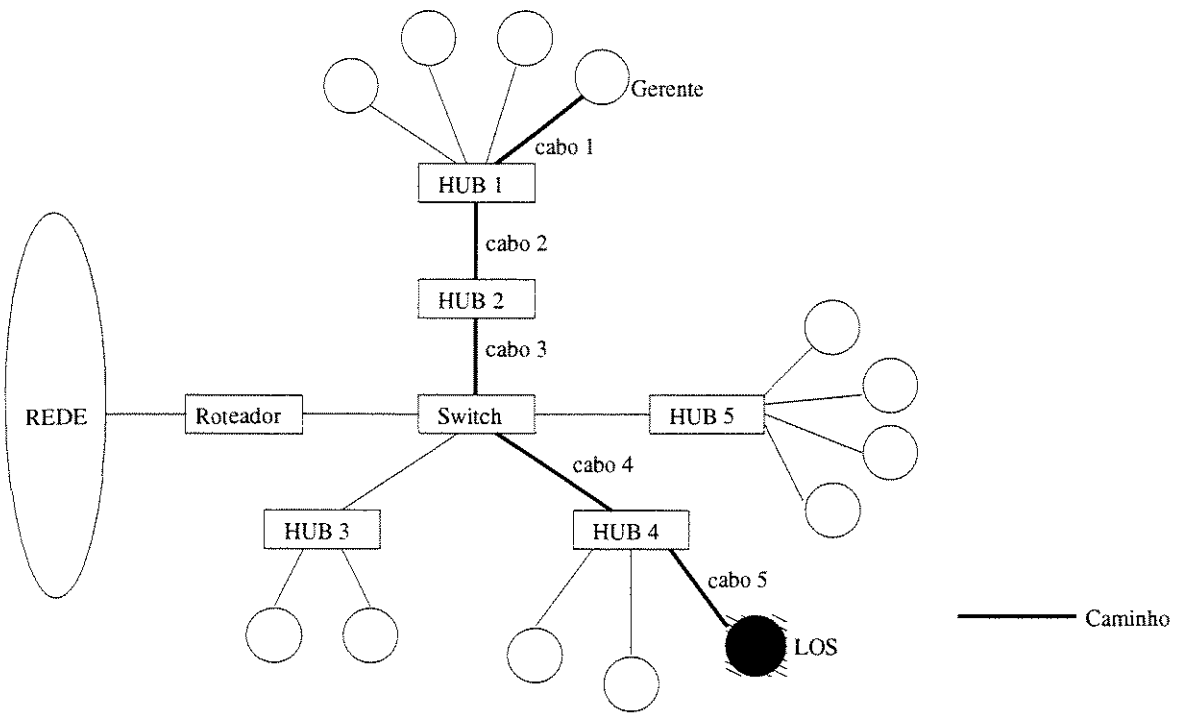
4.5.1 Diagnóstico de Falha de Comunicação

A forma de inferência do algoritmo de diagnóstico de falha do SCD é baseada no conhecimento do caminho que um pacote IP faz entre a máquina gerente e o dispositivo que gerou a primeira observação. Através da topologia, do esquema de endereçamento e de roteamento descritos no modelo da rede o sistema chega a este conhecimento. Os endereços de origem e destino do pacote e as máscaras de rede IP permitem que o gerente saiba se o destino encontra-se no mesmo domínio de colisão, subrede ou redes distintas.

Ao receber um alarme do SCS o gerente computa (através do modelo), por qual caminho este pacote deveria passar caso a rede estivesse em funcionamento normal. A primeira hipótese formada para o diagnóstico contém todos os componentes presentes no caminho, incluindo os cabos da conexão física. A partir desta hipótese inicial de possíveis elementos falhos, o algoritmo monta a estrutura de consultas que irá fazer para refiná-la.

Como mostrado na figura 4.5, o gerente consulta um dispositivo qualquer no caminho através de um ping. Se este dispositivo não responde ao comando (está inalcançável), o algoritmo elimina da hipótese todos aqueles dispositivos que se encontram após este elemento no caminho

pois certamente o problema estará ou nele, ou nos elementos anteriores e passa a verificar os outros componentes, iterando este processo até alcançar a menor hipótese que explique os sintomas. A hipótese final consiste normalmente de um único dispositivo e o cabo que o conecta no caminho.



Hipótese inicial = { Gerente, cabo1, HUB 1, cabo 2, HUB 2, cabo3, Switch, cabo 4, HUB 4, cabo 5, Dispositivo}
Ping Switch = LOS
Hipótese 2 = {Gerente, cabo1, HUB 1, cabo 2, HUB 2, cabo 3, Switch}
Ping HUB 2 = OK
Diagnóstico = {cabo3, Switch}

Figura 4.5: Refinamento da hipótese inicial, até a derivação do diagnóstico

O algoritmo tem o cuidado de verificar se o caminho de ida é diferente da volta (casos onde há ciclos na rede) e tratá-los adequadamente. Havendo diferença, o diagnóstico é feito para cada sentido.

O SCD faz as consultas aos dispositivos no caminho de forma binária, como mostrado no algoritmo a seguir:

A condição inicial deste algoritmo é a ativação do `active(Gerente,` x`,` $\emptyset$`)`, pelo SCS, quando este detecta uma perda de sinal para o dispositivo x. O sistema irá automati-

Algorithm 2 Algoritmo binário genérico

```
active(From,Dest,H)
        if (From = Dest) then return(H)
        p = path(From, Dest)
        e = select(From,Dest,p)
        if e cannot be queried then
                e = substitute(e)
        if (e = Dest) or (e = From) then
                return(H)
        else
                q = path(From,e)
                if e is reachable then
                        H = H − H ∩ q
                        active(e,Dest,H)
                else
                        H = H ∪ q
                        active(From,e,H)
```

camente atribuir o caminho entre a máquina gerente (que executa o SCS) e o elemento falho x como hipótese inicial.

A função `path(From,Dest)` computa a seqüência de dispositivos no caminho entre `From` e `Dest`, e sua volta. A função `select` é um algoritmo de busca binário que procura o próximo elemento a ser consultado na hipótese.

Uma vez encerradas todas as consultas necessárias no caminho, o SCD retorna o menor conjunto de dispositivos que explicam o problema e envia uma notificação como saída.

Um mecanismo simples de controle foi desenvolvido no sistema de coleta de status para garantir que uma falha reincidente ainda não tratada, não seja considerada como um novo problema, apenas advertindo sobre a permanência e não resolução do mesmo. Ao voltar a seu estado normal um sinal de confirmação é gerado como forma de controle.

A escolha da busca binária foi baseada nos testes realizados. Outras três formas de busca foram desenvolvidas. Duas delas, a *Busca em Retrocesso* e *Busca em Avanço* simplesmente seguem o caminho em sua ordem invertida (de trás para frente) ou natural, respectivamente. Estas formas não apresentam grandes ganhos de performance, a não ser quando da coincidência da posição da falha em relação ao caminho (mais a frente ou ao final do mesmo). Outra forma,

bem mais vantajosa, consistiu do uso de estatísticas de falhas dos componentes. Supondo-se, por exemplo, que switches sejam mais suscetíveis à falha do que hubs, estes serão consultados antes, podendo-se chegar mais rapidamente ao diagnóstico.

O intervalo de tempo do diagnóstico é regido pela latência dos pings. O tempo de processamento do cômputo do caminho e da análise dos dados é uma pequena fração do tempo total em estado de diagnóstico. Pouco se pode afirmar sobre o melhor desempenho da busca binária ou da probabilística, a não ser que elas são melhores do que as buscas em avanço ou retrocesso.

4.5.2 Diagnóstico de performance

A distinção entre os procedimentos de diagnóstico de falhas e de performance ocorreu por dois motivos. Primeiramente, para o diagnóstico de performance o SCD acessa informações da MIB RMON, uma MIB que até então não tinha sido utilizada. Posteriormente, o caminho previamente computado é reduzido aos domínios de colisão por onde o pacote passa.

O parâmetro adotado como sintoma para um problema de performance foi a *Taxa de Utilização do Meio*. O que caracteriza a degradação de performance neste trabalho é uma descarga de pacotes por algum dispositivo, que acaba gerando excesso de tráfego e conseqüentemente redução no desempenho da rede durante um período predeterminado de tempo.

A taxa de utilização é uma função que tem por variáveis o número de pacotes trafegados, octetos e o intervalo de tempo de medição. É medida através da interface de um equipamento em um segmento de rede. A MIB RMON [51] mantém uma tabela de histórico de monitoração em um segmento e através do campo *etherHistoryUtilization* mede o percentual de uso da rede em uma interface. Todos os dispositivos que possuam agente SNMP podem estar configurados para manter esta MIB, que funciona de forma bastante similar a um analisador de protocolos [49].

A Taxa de Utilização de cada rede é estritamente dependente do tipo de tecnologia utilizada, como por exemplo tamanhos de segmentos, número de nós por domínio e tamanho do pacote trafegado. Para que se possa ter uma idéia de quanto uma taxa de utilização elevada caracteriza uma degradação de performance é necessário estabelecer um parâmetro (média), um limite por tecnologia. O parâmetro pode ser medido de diferentes formas, onde o mais indicado é que seja feito empiricamente, constatando na prática os limites de utilização que um domínio de colisão e conseqüentemente uma rede suportam [49].

Para refinar os parâmetros de performance de uma rede sem dar tendência a valores muito altos ou baixos de utilização em um domínio, é conveniente distribuir as atribuições destes

valores e diagnosticar a rede por setores. Neste trabalho os parâmetros (thresholds) foram atribuídos, manualmente para cada domínio de colisão existente.

A tabela *etherHistoryTable* da MIB RMON é mantida por cada dispositivo onde os valores de monitoração são capturados ciclicamente em intervalos predeterminados de tempo e ocupando um tamanho máximo em bytes. Este tamanho varia de acordo com cada equipamento. As tabelas aqui utilizadas, mantinham até 256 entradas coletadas em intervalos de 30 segundos. Para dar fidelidade ao diagnóstico de performance foi utilizado a média das taxas de utilização destas entradas e ainda um mecanismo próprio de tolerância como mostrado na figura 4.6.

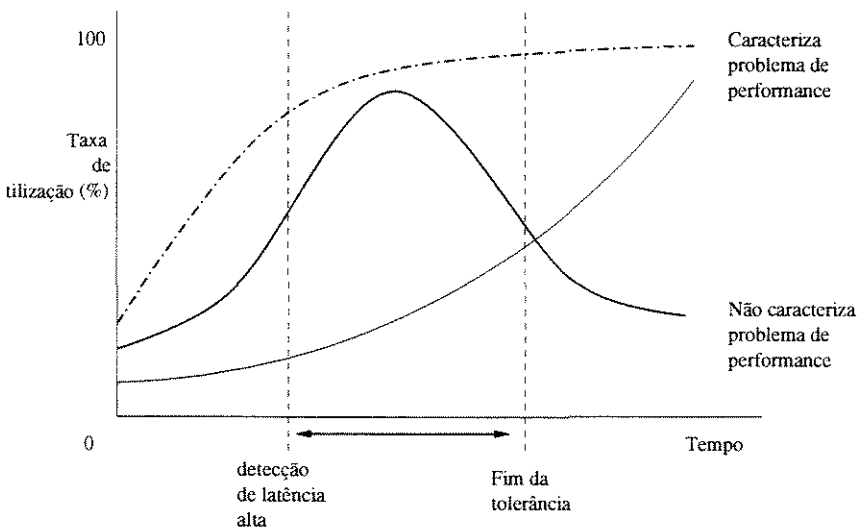


Figura 4.6: Nesta figura, três curvas mostram os possíveis comportamentos da taxa de utilização do meio. A janela de tolerância mostra o mecanismo adotado para verificar que esta taxa continua aumentando (caracterizam problemas de performance) ou diminuem (não caracterizam problemas de performance)

A tolerância é um tempo de espera adotado para verificar o comportamento da taxa de utilização. Se os valores coletados para o cálculo da média estiverem próximos do momento em que foi detectado uma utilização elevada, a média poderá ficar muito alta. O ideal é aguardar um tempo (ver figura 4.6), para garantir que o estado de degradação de performance se mantenha. Tipicamente a taxa de utilização poderá ficar muito elevada durante um intervalo curto de tempo devido a, por exemplo, uma transmissão de vídeo. Ao terminar a transmissão a utilização volta a seu estado normal não caracterizando um problema em si.

A tolerância utilizada para fins de teste foi de um minuto. Na amostragem de entradas da MIB RMON são coletados os quatro últimos valores. Esta tolerância também é um valor extre-

mamente variável e depende de uma análise minuciosa do tipo de rede, inclusive dos horários em que esta rede é mais usada. Isto implica que para se assumir efetivamente que a rede passa por uma degradação de performance, vários medidores precisam ser calibrados de forma empírica.

O SCD faz o diagnóstico de degradação de performance através do seguinte algoritmo genérico:

Algorithm 3 Algoritmo de diagnóstico de performance

performance(From,Dest,H)
    if ($From = Dest$) then return(H)
    else
        $p = path(From, Dest)$
        $DR = transform$(p)
        $H = query\_utilization$(DR)
    $query\_utilization$(DR)
    $e = select\_first$(DR)
    $utili = utili\_rate$(e)
    if $utili \leq Media \div Treshold$ then
        $DR = DR - e$
        $query\_utilization$(DR,H)
    else
        $H = e$
    $return$(H)

A condição inicial do algoritmo tem como hipótese o conjunto vazio. Diferente do diagnóstico de falha, este procedimento não faz busca binária pelo domínio de colisão, mas uma busca seqüencial até encontrar o responsável pela degradação de performance. Este modo de busca foi adotado pois a alta taxa de utilização em um dos domínios pode não interferir em outro, o que descaracteriza as intersecções feitas para o refinamento da hipótese.

Da mesma forma a função `path(From,Dest)` computa a seqüência de dispositivos no caminho entre `From` e `Dest`, e sua volta. No caso do diagnóstico de performance, porém, este caminho é transformado nos respectivos domínios de colisão pela função `transform(p)`. A variável DR é uma lista dos domínios de colisão em ordem estabelecida pelo caminho entre $From$ e $Dest$. O procedimento $query\_utilization$(DR) consulta a taxa de utilização de um elemento nos domínios de colisão da lista DR em ordem. Para cada domínio há um dispositivo

eleito para responder a essa consulta. Caso a utilização ultrapasse o limite preestabelecido o domínio falho é a solução do diagnóstico. Caso contrário o domínio consultado é retirado da lista e o procedimento é repetido para o próximo elemento. O tempo de tolerância é um atraso utilizado antes da chamada do algoritmo de performance. Como citado anteriormente, a variável *utili* é uma média de taxas de utilização capturadas na janela de tempo segundo o critério de tolerância acima descrito.

Finalmente, após apresentado a forma de funcionamento do diagnóstico de falhas e de performance, o algoritmo a seguir apresenta o modo de operação genérico do SCD:

Algorithm 4 Algoritmo genérico do SCD

```
poll(dev)
        Result = ping(dev)
        if Result = LOS then
                active(Gerente,dev,Diagnstico_Falha)
        else
                Pkt_rate = analyse_loss_packet(Result)
                Ping_latency = analyse_latency(Result)
                if Pkt_rate > Limite OR Ping_latency > threshold then
                        wait(Tolerancia)
                        performance(Gerente, dev, Diagnstico_Performance)
```

Este algoritmo analisa cada ping efetuado pelo SCS. Se uma perda de sinal para o dispositivo *dev* for observada, o SCD entra em seu diagnóstico de falha através do procedimento `active(`*Gerente*`,` *dev*`,` *Diagnstico_Falha*`)` retornando a solução do diagnóstico. Caso contrário, as estatísticas de performance do ping serão analisadas. Se as variáveis *Pkt_rate* ou *Ping_latency* excederem valores predefinidos o sistema aguarda um tempo de tolerância e inicia o diagnóstico de performance através do procedimento `performance(`*Gerente*`,` *dev*`,` *Diagnstico_Performance*`)`.

Este algoritmo é regido pelos pings executados pelo SCS. O modo online do Sistema de Diagnóstico Baseado em Modelo itera indefinidamente fazendo a monitoração e o diagnóstico da rede, reportando as falhas encontradas, sua localização e sua causa.

4.6 Sumário

O *Sistema de Diagnóstico Baseado em Modelo* SDBM apresentado neste capítulo é uma aplicação que engloba vários aspectos da gerência de redes para consolidar o uso de Raciocínio Baseado em Modelo de Funcionamento Correto no domínio de localização e diagnóstico de falha e performance em uma rede local.

Grande parte do sistema dedica-se à construção de um modelo lógico-funcional da rede, que é usado pelos outros subsistemas na monitoração e diagnóstico da rede. A construção deste modelo se faz através de um processo de descoberta da rede, depuração das informações e posterior diagnóstico de configuração para se obter um modelo completo suficiente para tornar viável o diagnóstico.

O restante do sistema dedica-se a monitoração dos dispositivos que acontece através de um mecanismo de polling inteligente e análise das observações encontradas durante este processo. Caso as observações caracterizem uma falha de comunicação ou uma degradação de performance segundo valores médios de referência e sua comparação com o modelo lógico-funcional, o sistema entra em seu estado de diagnóstico retornando como solução o menor conjunto de causas que expliquem os sintomas.

O sistema de diagnóstico raciocina de uma forma ativa sobre a rede, fazendo as consultas necessárias para se refinar a hipótese inicial. Este raciocínio tem como principal referência o conhecimento do caminho que um pacote IP percorre na rede, tendo por base seu modelo de funcionamento correto. Os pontos de discrepância entre o modelo e as observações capturadas são indícios da localização da falha.

O capítulo de conclusão discute vários aspectos da aplicação do Raciocínio Baseado em Modelo no diagnóstico de redes locais não abordados neste capítulo.

Capítulo 5

A rede LSI e os Testes Realizados

O sistema de localização e diagnóstico de falhas e degradação de performance implementado neste trabalho teve como base de testes a rede local do Laboratório de Sistemas Integráveis da Universidade de São Paulo (LSI-USP).

As características desta LAN mediana, constituída de aproximadamente 120 componentes são descritas em maiores detalhes a seguir. Esta foi palco de vários testes funcionais do sistema, comprovando a eficiência do mesmo, possibilitando diversas melhorias e ainda indicando outras possíveis funcionalidades.

Todos os testes ocorreram paralelamente ao desenvolvimento do sistema. A cada avanço no projeto, novos testes eram realizados.

Os testes giraram em torno principalmente da efetivação do sistema, a fim de colocá-lo em funcionamento. Portanto, pode-se notar que os testes acompanharam o desenvolvimento do sistema a cada novo módulo. Primeiramente durante o processo de descoberta da rede, a subseqüente geração da seqüência de polling, a monitoração da rede através dos pings, até finalmente a inserção forçada de falhas no sistema para testar o processo de diagnóstico e comparar os resultados obtidos.

Este capítulo apresenta em uma primeira seção, a descrição da rede na qual os teste foram realizados e suas principais restrições. Posteriormente mostra os passos realizados, a geração de falhas, o processo de diagnóstico e alguns dados coletados durante algumas amostras.

5.1 A Rede LSI-USP

A Rede Local do Laboratório de Sistemas Integráveis da Universidade de São Paulo (LSI-USP Brasil) foi utilizada para fins de teste do sistema de diagnóstico desenvolvido neste trabalho.

Esta é uma rede baseada nos protocolos TCP/IP que opera sobre a arquitetura Ethernet composta por aproximadamente 120 componentes dentre os quais constam:

- 3 roteadores
- 3 switches
- 12 hubs
- dezenas de terminais

O esquema de endereçamento desta LAN é composto pelas seguintes redes:

1. rede 10.0.10.0 com máscara 255.255.255.0
2. rede 10.0.160.0 com máscara 255.255.254.0
3. rede 143.107.161.128 com máscara 255.255.255.128
4. rede 143.107.160.0 com máscara 255.255.255.0
5. rede 192.168.10.0 com máscara 255.255.255.0
6. rede 192.168.210.0 com máscara 255.255.255.0

A figura a seguir 5.1 mostra a topologia desta LAN.

A LAN-LSI é gerenciada pelo protocolo SNMP, porém, sem a adoção de nenhuma plataforma de gerência. Seus componentes são então configurados para manter agentes SNMP e conseqüentemente as MIBs apropriadas, que respondem às solicitações do gerente, destacado na topologia da figura 5.1, cujo endereço IP fixo era 10.0.161.138.

Apesar da configuração dos agentes, que mantinham as informações das MIBs, muitos deles não possuíam seus mecanismos de Traps (envio autônomo de alarmes). A LAN-LSI também não possuía mecanismos de monitoração cíclica como polling, sendo que em grande parte dos casos de falha o processo de localização e diagnóstico era feito manualmente.

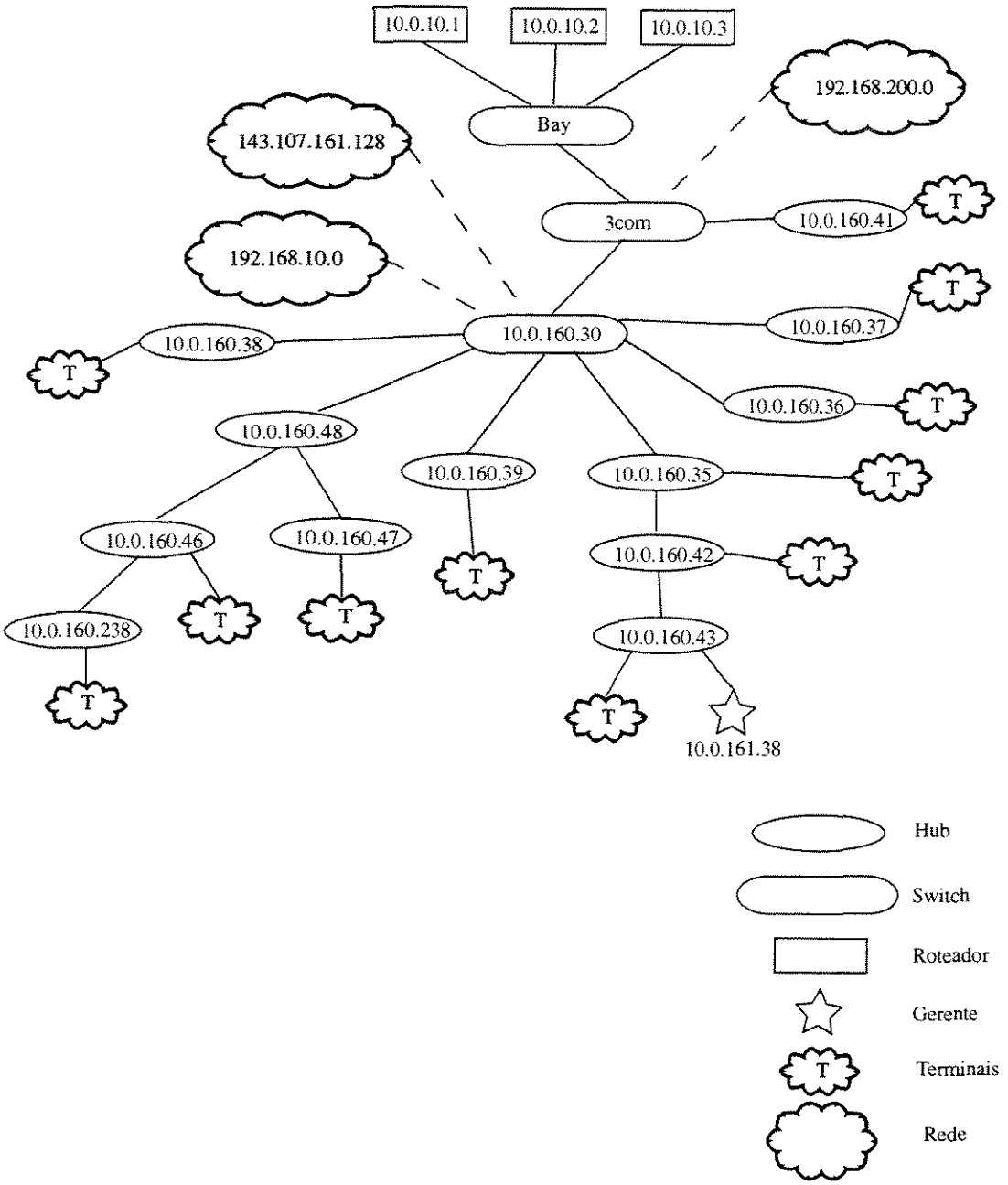


Figura 5.1: Topologia da LAN-LSI

Por se tratar de uma rede destinada para fins de pesquisa e testes, muitos dos equipamentos não possuem suas MIBs consistentes. Alguns deles sequer as têm. A finalidade acadêmica desta rede confere à mesma um dinamismo particular, seu diagnóstico de configuração (passo importante na manutenção da rede) é difícil e muitas vezes desatualizado.

Certamente um sistema de diagnóstico que abordasse toda esta rede deveria ser extremamente preciso e minucioso. Para contornar este obstáculo, no entanto, assumiu-se o diagnóstico de uma parte da rede, aquela cujos agentes estavam atualizados e corretamente configurados.

Isto restringiu a topologia vista na figura 5.1 à parte destacada na figura 5.2.

Deste modo, os aspectos importantes do processo de diagnóstico baseado em modelo puderam ser analisados. A manutenção de informações de gerenciamento em uma rede é de suma importância e mais, é vital para o funcionamento de qualquer plataforma. Por isso, a restrição dos testes a uma porção "polida" da rede foi uma decisão com foco na análise da aplicabilidade do sistema e não somente direcionada à solução de problemas como sua configuração.

Outra característica desta rede é que ela não implementa roteamento dinâmico. Suas tabelas de roteamento são configuradas manualmente.

Grande parte dos terminais são computadores pessoais. Isto significa que a qualquer momento os mesmos podem estar desligados, o que não implica necessariamente uma falha. Para superar esta restrição, como apresentado no capítulo anterior, o mecanismo de eleição de um representante irá sempre procurar por um computador "ligado" no mesmo domínio de colisão, ou elegerá um elemento diretamente conectado para realizar o diagnóstico. Na pior hipótese, quando não há representantes, o diagnóstico irá retornar a possibilidade deste elemento estar simplesmente desligado e não falho.

5.1.1 Valores estimados

As informações de quais computadores pessoais podem ser desligados não pode ser coletada automaticamente via processo de descoberta. Estas são, portanto inseridas manualmente.

Diversos valores, principalmente os utilizados no diagnóstico de performance, também precisam ser discriminados. O processo de coleta de estatísticas de performance é empírico, depende de inúmeros fatores como a distribuição de usuários pela rede, suas funções, seu horário de trabalho, enfim, um estudo minucioso à parte, para se obter dados precisos sobre a performance.

Os valores mais utilizados para diagnosticar a performance e as falhas são a latência, a taxa de utilização do meio, a tolerância (janela de tempo antes de confirmar o problema) e a perda de pacotes. A latência, na maioria dos casos utilizada no ping como referência para a perda

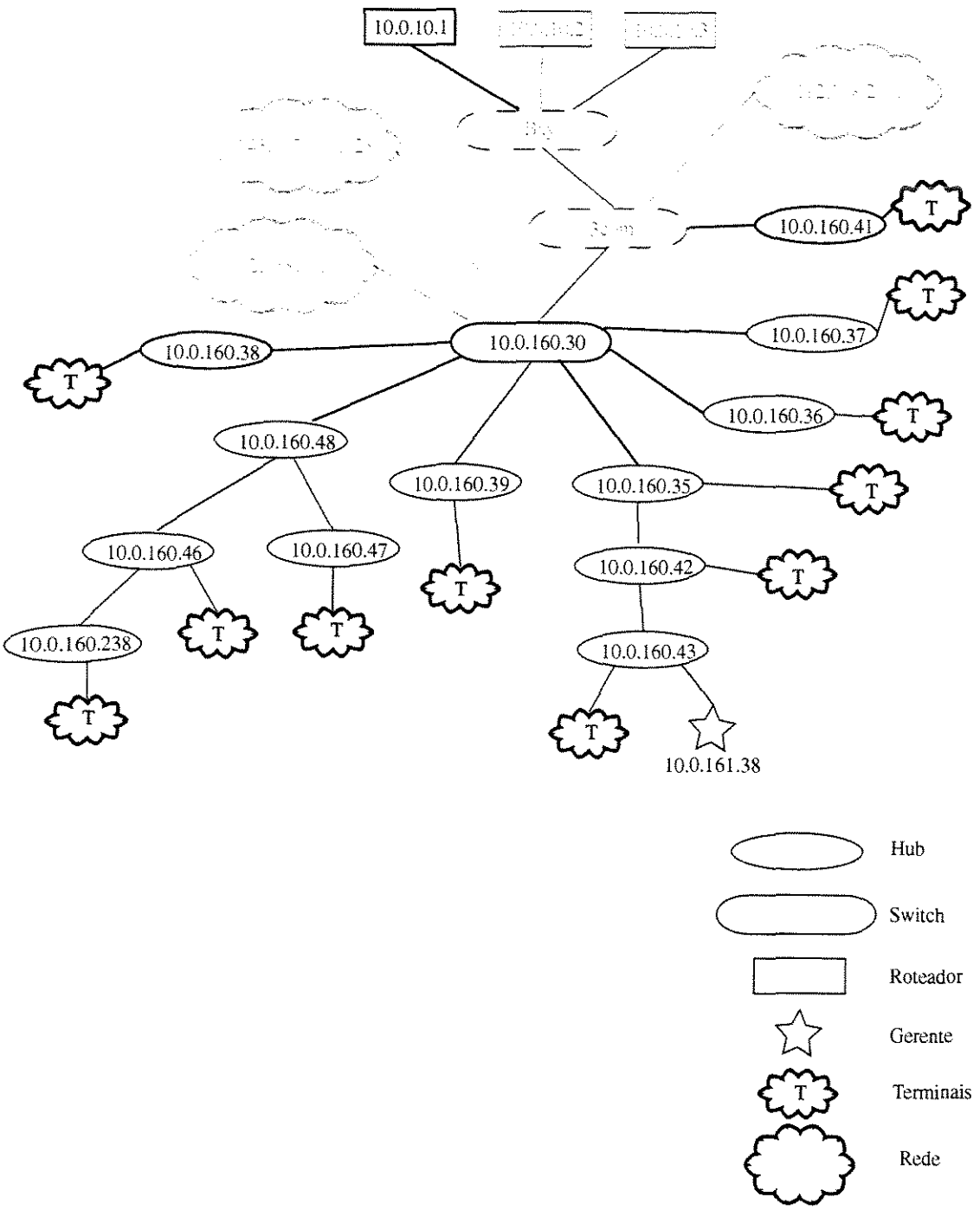


Figura 5.2: Porção efetivamente adotada para os testes. As áreas transparentes pertencem a rede porém não foram descobertas. Os switches Bay e 3Com foram inseridos manualmente para ligar fisicamente ao roteador 10.0.10.1

de sinal é um padrão de 10 milisegundos do próprio comando. Já a taxa de utilização do meio depende das características da rede. Para os fins de teste, foi adotado que 70% de utilização do meio já caracteriza algum problema de performance. O sistema prevê, na verdade, a atribuição de diferentes limites de utilização, um para cada domínio de colisão. A perda de pacotes está diretamente associada a utilização do meio , 30% foi considerado um valor limite. A janela de tolerância é um ponto bastante passível de discussões, ela indica quanto tempo aguardar antes de inferir que uma alta taxa de ocupação do meio se deve a uma falha. Depende, na verdade, do tipo de aplicação mais utilizado pelos usuários de um determinado setor, que com alta probabilidade deverão estar num memo domínio de colisão. Este valor é uma constante para fins de teste.

5.2 Testes Realizados

Os primeiros testes realizados tiveram caráter bastante funcional. O objetivo inicial era analisar o funcionamento da descoberta da rede. Este processo durou em média seis horas (para cada teste de descoberta de toda a rede) e aconteceu várias vezes durante a implementação do sistema. Como resultado o sistema descobriu aproximadamente 65% da rede, tanto com informações físicas quanto lógicas. Novamente deve-se considerar a ressalva de que boa parte da rede não estava adequadamente configurada com informações de gerência, e que, por ser um processo noturno (afim de minimizar impacto na rede) muitos computadores pessoais encontravam-se desligados, o que justifica este percentual um tanto baixo. Ao analisar a porção destacada da figura 5.2, a estatística quanto a descoberta melhora: grande parte dos componentes foi descoberta (cerca de 85%) de forma correta, com informações suficientes para montar o modelo estrutural e funcional no nível de abstração proposto. Isto comprova a dependência do sistema de descoberta nas informações presentes nos agentes de gerenciamento. De fato, grande parte das plataformas de gerenciamento [25, 26, 39] também exigem uma configuração mínima dos agentes. Os 15% restantes são alguns componentes não gerenciáveis como hubs e alguns computadores que estavam desligados durante a descoberta. Estes 15% são as informações que devem ser inseridas manualmente no modelo, junto com as conexões que eles implicam.

O passo seguinte, foi o teste do sistema de monitoração da rede (polling) descrito como o modo online. Em sua implementação inicial, o sistema estava desprovido do algoritmo de pings inteligentes de modo que uma lista de componentes foi construída manualmente. A operação deste módulo trata do envio de pings para os componentes desta lista e da análise das infor-

mações como perda de pacotes e a latência do próprio ping. A comunicação desta parte do sistema desenvolvida em Prolog, com a porção que executava efetivamente os comandos na rede (desenvolvido em Java) foi feita através de filas controladas pelo sistema operacional (FIFO do Linux). Durante os testes deste sistema foram ajustadas funções como a periodicidade dos pings, e um mecanismo de controle para que, ao diagnosticar uma falha, o componente falho saísse da lista de pings até sua manutenção.

A função de polling inteligente foi desenvolvida e testada posteriormente. O teste da geração da seqüência de polling não dependeu da rede real, pois, o modelo já possuía informações suficientes para se construir uma árvore à partir da máquina gerente e posteriormente executar o método de embaralhamento para se obter o resultado.

Uma vez que a monitoração coletava e analisava os dados de forma coerente, identificando as falhas e a degradação de performance, o sistema estava preparado para entrar no estado de diagnóstico.

Na análise de cada ping as seguintes tomadas de decisão eram feitas:

- Caso o ping retornasse uma perda de sinal (latência maior do que 10 milisegundos), o sistema entraria em estado de diagnóstico de falha de comunicação, executando o algoritmo de refinamento baseado no caminho do pacote.

- Caso o ping retornasse uma latência maior do que 0.7 milisegundos [1], o sistema entraria no diagnóstico de performance, iniciando os comandos para a coleta de estatísticas da MIB RMON.

Para gerar um erro real na rede, alguma interface, escolhida aleatoriamente era desconectada, causando assim uma perda de sinal.

O sistema diagnosticou corretamente todos os casos testados, desde a notificação do alarme até o processo de localização da falha. Em todos os casos, o diagnóstico obtido consistiu do dispositivo (cuja interface houvera sido desconectada) e do cabo.

Além da funcionalidade do sistema, algumas observações puderam ser colocadas. O tempo para a realização de um diagnóstico é dominado pelas latências dos pings. Mesmo com a aplicação dos diferentes algoritmos de busca no caminho, pouco se pode afirmar sobre a maior ou menor eficácia dos mesmos. A posição da falha em relação ao caminho pode dar vantagem

[1] Valor considerado razoável para os primeiros testes dentro de um mesmo domínio de broadcast, para o tamanho da rede testada.

a algum dos métodos. O binário é portanto preferível, por gerar menos pings na maioria dos casos.

Para o diagnóstico de performance o sistema monitorou sistematicamente todas as latências dos pings. A medida que uma latência ultrapassasse o limite preestabelecido de 7 milisegundos o sistema entrava em seu diagnóstico de performance, consultando a taxa de utilização da MIB RMON [51] dos elementos em todos os domínios de colisão no caminho até o elemento cuja latência alta fora detectada. Em se comprovando uma taxa de utilização superior a 70%[2] em algum dos domínios de colisão, o sistema retornava o mesmo como solução.

Da mesma forma, em algumas circunstâncias uma transmissão de vídeo fora executada, para elevar a taxa de utilização e gerar o problema de performance. O sistema detectou e diagnosticou o problema corretamente.

Em cada experimento realizado, vários ciclos de polling eram feitos. Como forma de aprimorar o sistema, um mecanismo simples de controle de falhas foi desenvolvido, evitando que em ciclos diferentes o mesmo problema fosse diagnosticado. Desta forma, a cada 60 ciclos que um dispositivo permanecesse falho, uma mensagem era enviada.

Algumas estatísticas foram coletadas a cada experimento, como o percentual de falhas de comunicação dentre os dispositivos, o percentual de problemas de performance encontrados, a quantidade de pacotes perdidos (totalizando as vezes em que a perda excedia o limite preestabelecido), o tempo total despendido em estado de diagnóstico (por ciclo de polling), a média de tempo para cada diagnóstico e o tempo total do ciclo de polling. Estas permitiram mostrar principalmente que tanto o tempo gasto com o diagnóstico quanto o tempo para realizar o ciclo completo do polling são regidos pelo tempo gasto com os pings. Ainda em alguns casos, o sistema entrou em diagnóstico de performance, porém retornou solução nula, demonstrando que a alta latência do ping não caracterizou efetivamente um problema. Vários casos aconteceram de se detectar perda de sinal em computadores que eram desligados durante o processo, esses pertenciam a uma lista preestabelecida de computadores, e também não caracterizaram uma falha na rede.

Como comentado anteriormente, um estudo empírico sobre a rede seria necessário para calibrar as variáveis utilizadas como limites em todo o processo. Assim seria possível coletar dados reais como a porcentagem de erros na rede durante um intervalo de tempo, o tempo real gasto no processo de diagnóstico, enfim, dados interessantes para um gerente de redes, além da

[2]Taxa de utilização calculada segundo os critérios de tempo explicados no capítulo anterior e segundo uma média que consideramos razoável para as características desta rede.

localização e diagnóstico da falha.

Vários problemas aconteceram durante os testes, muitos deles com a própria codificação, outros com os sistemas operacionais como a gerência das filas (FIFO). Um grande desafio no processo foi certamente alcançar a representação na linguagem Prolog mais adequada para o modelo, de forma que as operações sobre o mesmo pudessem ser feitas do modo mais simples. Foram necessários alguns procedimentos de descoberta, e muitas adaptações manuais para se obter o primeiro modelo que representasse de forma adequada a rede real.

Capítulo 6

Discussão e Conclusão

Com o objetivo inicial da aplicação do Raciocínio Baseado em Modelo no diagnóstico de redes de computadores locais, este trabalho tornou-se bastante multidisciplinar, abrangendo um pouco dos vários aspectos da Gerência de Redes.

Somando-se a proposta de um modelo de resolução de problemas [5, 7] à necessidade de uma ferramenta simplificada porém mais personalizada para auxílio na gerência de falha e performance em uma rede local, o SDBM (Sistema de Diagnóstico Baseado em Modelo) concretizou, através de desenvolvimento e testes, a idéia inicial proposta.

Os principais marcos atingidos durante o desenvolvimento deste sistema foram o processo de descoberta da rede, a construção de um modelo lógico-funcional, a monitoração através de um mecanismo de polling e finalmente o processo de diagnóstico. Cada um destes items possui uma série de discussões e perspectivas já abordados pela comunidade científica. Estes pontos são agora discutidos sob a ótica dos desenvolvedores deste trabalho.

Como apresentado em detalhes nesta dissertação, um Sistema Baseado em Modelo depende muito de quão bem o modelo representa o mundo real. Existem várias formas de se implementar um sistema de descoberta da rede [33, 8], porém, seu resultado sempre dependerá de como, ou que representação obter da rede, quais informações são dispensáveis ou indispensáveis. No sistema NNM da plataforma HPOpenview [39], por exemplo, a forma como hubs (não gerenciáveis) conectam um determinado segmento de rede é descartada, muitas estações também o são e esta decisão é bem justificada com o que se deseja efetivamente gerenciar na rede. Assim como outros processos de descoberta implementados, o processo aqui desenvolvido esteve restrito a configuração dos equipamentos, em especial quando amarrado a um protocolo específico como o SNMP. O processo de descoberta aqui implementado não difere de outros a não ser

pelo tipo de informações coletadas, restritas ao tipo de monitoração que se objetivou e a forma como o modelo deveria estar representado. A topologia de um segmento de rede neste trabalho foi em muitos casos, o diferencial entre uma solução de diagnóstico contendo vários cabos e dispositivos, ou uma mais plausível contendo poucos. Por isto também a ênfase na inserção manual de algumas informações.

O tipo de monitoração teve foco na análise de observações ocorridas na rede como a perda de sinal e na análise da performance a partir dos dados contidos na MIB RMON, ambos através de um mecanismo de polling. A forma de representação da rede teve como objetivo ser simples, porém tão apurada quanto necessário. A quantidade de informações presentes nas MIBs é enorme. O SDBM foi restrito a coleta das informações mais importantes para se atingir um modelo com nível de abstração confiável.

A abstração utilizada foi a representação de vários dispositivos em um único equipamento. Por exemplo, um hub possui várias portas, cada qual com uma característica de conexão física, no entanto todas estas portas foram consideradas um único equipamento. Da mesma forma as várias interfaces de um roteador foram representadas por um único dispositivo, no entanto contendo informações sobre estas interfaces e sua tabela de roteamento. Este tipo de representação torna o modelo mais simples e mais generalizado. Como o conhecimento sobre o caminho de um pacote IP foi uma das principais informações retiradas do modelo, as informações sobre qual número de porta conectava dois dispositivos não foi vital. Num trabalho futuro esta característica poderá indicar com um nível maior de profundidade, informações que possam levar a uma solução mais detalhada. O modelo construído neste trabalho atingiu simplicidade e consistência suficiente para ser utilizada com sucesso no diagnóstico baseado em modelo.

A manutenção do modelo levanta um aspecto importante, o caráter mutável da rede. As redes locais estão sob constante mudança, atualizam suas tabelas de rotas, nós são inseridos ou removidos, endereços diferentes são atribuídos. Se o modelo não for atualizado com certa freqüência, em pouco tempo sua representação não irá condizer com a rede real. A rede LSI-USP não implementa roteamento dinâmico e é uma rede estável, de forma que as poucas mudanças que ocorreram durante os testes puderam ser inseridas manualmente no modelo. No entanto, um processo de atualização do modelo é uma sugestão não só válida como importante. As ferramentas de gerenciamento, também através de um mecanismo de poll, verificam quais nós estão presentes ou ausentes na rede, e atualizam as informações destes nós com freqüência configurável. [1]. Um mecanismo similar pode ser implantado no processo de poll da rede (sub-

[1] É comum que, ao se olhar a representação gráfica da rede em uma destas ferramentas, muitas informações

sistema SCS), buscando de tempos em tempos novas informações de configuração. O interesse maior é que as tabelas de roteamento estejam sempre atualizadas, pois são fundamentais para o cômputo do caminho entre dois nós, para isso, um mecanismo distribuído e autônomo de envio de tabelas de rotas atualizadas poderia ser implantado. É importante que o modelo esteja sempre representando a rede real da melhor forma possível.

É exatamente neste aspecto que o Raciocínio Baseado em Modelo apresenta uma grande vantagem - as mudanças do modelo não comprometem a forma de inferência do sistema de diagnóstico. Os algoritmos de raciocínio não estão atrelados a uma disposição dos elementos no modelo e nem a forma como a rede estava representada no passado. Eles estão ligados ao seu modo de funcionamento, um caminho será sempre computado da mesma forma, um hub sempre trabalha da mesma forma. Esta característica pôde ser comprovada ainda por uma pequena simulação. Um modelo de rede diferente do aplicado a rede LSI-USP foi desenvolvido e um simulador de erros provocava a entrada em estado de diagnóstico em locais e tempos distintos: a solução do diagnóstico foi correta para todos os casos mesmo para um modelo diferente do original (LSI-USP). O paradigma de Raciocínio Baseado em Modelo dá flexibilidade ao sistema evitando a reconstrução do mecanismo de inferência.

São sugestões para um trabalho futuro, a implementação de um mecanismo de atualização do modelo, tratamento de redes com roteamento dinâmico [38], a alteração da abstração do modelo aprofundando-se em informações que possam minimizar a solução do diagnóstico e a validação da heurística de polling. É ainda válido que se analisem outras informações de performance como a taxa de colisão no meio, que indica efetivamente os problemas que vem ocorrendo no domínio de colisão, e a taxa de perda de pacotes, que somadas a informações da camada da aplicação (a qual aplicação pertencia o pacote) podem concluir um diagnóstico mais rico e propiciam a evolução do sistema. É característica comum aos sistemas baseados em técnicas de Inteligência Artificial a capacidade autônoma de evolução e aprendizado. O sistema aqui desenvolvido pode agregar esta característica no que se refere a coleta permanente de informações de performance da rede e adaptação dos "thresholds" de forma inteligente, monitorando a evolução da mesma, o comportamento de seu uso e atualizando o modelo de forma a refletir cada vez melhor a saúde da rede.

O sistema desenvolvido neste trabalho atingiu seu objetivo de ser uma ferramenta de diagnóstico baseado em modelo, porém, abrangendo muito mais aspectos da gerência de rede do que se esperava alcançar. Existem, de fato, sistemas que já tratam da monitoração de redes, do

estejam desatualizadas, informações que foram levantadas durante o processo de descoberta.

processo minucioso de descoberta e da gerência de falha e performance, no entanto, a passagem por todos estes aspectos foi um passo necessário e que agregou muito conhecimento e experiência em várias disciplinas. Permitiu (1) a aplicação de um método de resolução de problema (PSM) [5, 4, 3] na automatização do processo de diagnóstico no ambiente de gerência de redes, (2) o conhecimento do processo de descoberta da rede com ênfase em informações mais apropriadas para a construção do modelo (incluindo conexão física), (3) a identificação de formas de interação entre o sistema de diagnóstico e a rede, o sistema e o modelo e entre o modelo e a rede, (4) o conhecimento e a implantação de uma heurística para o processo de monitoração (polling), (5) a identificação da discrepância entre o observado e o esperado, caracterizando os problemas de falha e performance em uma rede e, (6) as diferentes abstrações que o modelo lógico-funcional da rede pode assumir.

Os testes na rede LSI-USP e outros feitos através de simulação, validaram o sistema como uma ferramenta simples e importante na gerência de redes locais. Em especial, o Raciocínio Baseado em Modelo mostra ser uma técnica da Inteligência Artificial que simplifica e dá flexibilidade a complexa tarefa de se gerenciar uma rede.

Bibliografia

[1] IEEE Std. 802.3. Part 3:carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications. IEEE Std 802.3, 2000 Edition, 2000.

[2] E. S. Artola and L. M. R. Tarouco. Um sistema especialista para gerência pró-ativa remota. In *14th Simpósio Brasileiro de Redes de Computadores*, pages 118–139, 1996. In Portuguese.

[3] V. R. Benjamins. *Problem Solving Methods for Diagnosis*. Phd. thesis, University of Amsterdam, Amsterdam, 1993.

[4] V. Richard Benjamins, Dieter Fensel, and Remco Straatman. Assumptions of problem-solving methods and their role in knowledge engineering. In *European Conference on Artificial Intelligence*, pages 408–412, 1996.

[5] Volnys Bernal, Leliane Nunes de Barros, Marilza Antunes de Lemos, and Jacques Wainer. Building reusable models for the communication networks domain. In *In Fourth Australian Workshop on Knowledge Acquisition: AWKA 99*, Sidney, 1999.

[6] Volnys Bernal, Leliane Nunes de Barros, Marilza Antunes de Lemos, and Jacques Wainer. Fault diagnosis for local area network environments. In *IEEE LANOMS'99 - Latin American Network Operations and Management Symposium*, 1999.

[7] Volnys Bernal, Leliane Nunes de Barros, Marilza Antunes de Lemos, and Jacques Wainer. Model based diagnosis for network comunication faults. In *In Proceedings of the Third International Workshop on Artificial Intelligence for Distributed Information Networking: AiDIN 99*, Orlando - Florida, 1999. Steven Willmott and Sue Abu-Hakima.

[8] Yuri Breitbart, Minos N. Garofalakis, Cliff Martin, Rajeev Rastogi, S. Seshadri, and Abraham Silberschatz. Topology discovery in heterogeneous ip networks. In *INFOCOM*, pages 265–274, 2000.

[9] N. Brownlee, C. Mills, and G. Ruth. *RFC 2063: Traffic Flow Measurement: Architecture*, January 1997.

[10] S. Brugnoni, G. Bruno, and et.al R. Manione. An expert system for real time fault diagnosis of the italian telecommunications network. *IFIP - International Symposium on Integrated Network Management, (ISINM'93)*, pages 617–628, 1993.

[11] J. Case, M. Fedor, M. Schoffstall, and J. Davin. *A Simple Network Management Protocol*, May 1990.

[12] D. E. Comer and D. L. Stevens. *Internetworking with TCP/IP - Design, Implementation, and Internals*, volume 2. Prentice Hall, second edition, 1994.

[13] R. Cronk, P Callahan, and L. Bernstein. Rule based expert systems for network managements and operations. *IEEE Network*, 2:7–21, 1988.

[14] Adnan Darwiche. Model-based diagnosis under real-world constraints. *AI Magazine*, 21(2):57–73, 2000.

[15] E. Decker, P. Langille, A. Rijsinghani, and K. McCloghrie. *Definitions of Managed Objects for Bridges*, July 1993.

[16] G. Dreo and R. Valta. Using master tickets as a storage for problem solving expertise. In *IFIP/IEEE International Symp. on Integrated Network Management IV*, pages 328–340, 1995.

[17] Oskar Dressler and Peter Struss. Model-based diagnosis with the default-based diagnosis engine: Effective control strategies that work in practice. In *European Conference on Artificial Intelligence*, pages 677–681, 1994.

[18] R. Droms. *RFC 2131: Dynamic Host Configuration Protocol*, March 1997.

[19] International Organization for Standardization / International Eletrotechnical Committee. *Information Processing Systems - Open System Interconection - Basic Reference Model Part 1*, 1984. International Standard 7498-1.

[20] Peter Frohlich. *DRUM-II, Efficient Model-based Diagnosis of Technical Systems*. PhD thesis, Universidade de Hannover, 1998.

[21] Johann Gamper. *A Temporal Reasoning and Abstraction Framework for Model-Based Diagnosis Systems*. PhD thesis, Universidade de Hannover, 1996.

[22] D. E. Heckerman and R. A. Miller. Towards a better understanding of the internist-1 knowledge base. In *Proceedings of Medinfo*, Washington, DC, pages 27–31. North-Holland, New York, 1986.

[23] Robert M. Hinden. Ip next generation overview. In *Toronto IETF meeting on July 25, 1994*. IETF, 1995.

[24] K. Houck, S. Calo, and A. Finkel. Towards a practical alarm correlation system. In *IFIP/IEEE International Symp. on Integrated Network Management IV*, pages 226–237, 1995.

[25] Micromuse Inc. *NETCOOL Precision 2.0*, June 2000.

[26] Micromuse Inc. *NETCOOL Visionary*, August 2000.

[27] G. Jakobson and M. Weissman. Alarm correlation. *IEEE Network*, 7:52–59, 1993.

[28] G. Jakobson and M. Weissman. Real-time telecomunication network management extending event correlation with tmeporal constraints. *IFIP/IEEE 9 (ISINM'95)*, pages 290–301, 1995.

[29] I. Katzela and M. Schwartz. Schemes for fault identification in communication networks. *IEEE/ACM Transactions on Networking, IEEE Computer Society and the ACM with its Special Interest Group on Data Communication (SIGCOMM), ACM Press*, 3, 1995.

[30] L. Lewis. A case-based reasoning approach to the resolution of faults in communication networks. In *ISINM - International Symposium on Integrated Network Management*. Elsevier, 1993.

[31] G. Luderer, H. Ku, B. Subbiach, and A. Narayan. Network management agents supported by a java environment. Technical report, Arizona State University, 1996.

[32] K. McCloghrie and M. Rose. *Management Information Base for Network Management of TCP/IP-based internets*, May 1990.

[33] Robert E. McGrath. Discovery and its discontents: Discovery protocols for ubiquitous computing. Technical report, National Center for Supercomputing Applications, University of Illinois, 2000. UIUCDCS-R-2000-2154.

[34] D. McMaster and K. McCloghrie. *Definitions of Managed Objects for IEEE 802.3 Repeater Devices*, September 1993.

[35] Dilmar Malheiros Meira. *A Model For Alarm Correlation in Telecommunications Networks*. PhD thesis, Universidade Federal de Minas Gerais, Belo Horizonte, November 1997.

[36] Cristina Melchiors. Raciocínio baseado em casos aplicado ao gerenciamento de falhas em redes de computadores. Master's thesis, Universidade Federal do Rio Grande do Sul - Instituto de Informática, Porto Alegre, August 1999.

[37] P. Mosterman and G. Biswas. Model-based diagnosis of dynamic systems. *Seventh Journees du L.I.P.N.*, pages 143–154, september 1997.

[38] Francisco Watzko Neto. Gerenciamento pró-ativo: Consideração do tráfego de aplicação e utilização de series temporais. Master's thesis, Universidade Estadual de Campinas - Instituto de Computção, Campinas, 1997.

[39] Hewlett Packard. *Managing Your Network with HP OpenView Network Node Manager*, January 2000.

[40] Y. PENG and J.A. REGGIA. *Abductive Inference Models for Diagnostic Problem Solving*. Springer-Verlag, 1990.

[41] J. Postel. *RFC 792: Internet Control Message Protocol*, September 1981.

[42] J. B. Postel. *User Datagram Protocol*, 1980. Request for Comments 768.

[43] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. *Address Allocation for Private Internets*, February 1996.

[44] M. Rose. *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*, March 1991.

[45] M. Rose and K. McCloghrie. *Structure and Identification of Management Information for TCP/IP-based Internets*, May 1990.

[46] M. T. Rose. *The Simple Book - An Introduction to Internet Management*. Prentice Hall, second edition, 1994.

[47] L. F. Soares, G. Lemos, and S. Colcher. *Redes de Computadores, das LANs, MANs e WANs às Redes ATM*, volume 1. Editora Campus, Rio de Janeiro, second edition, 1995.

[48] Friedrich Steimann, Peter Frohlich, and Wolfgang Nejdl. Model-based diagnosis for open systems fault management. *AI Communications*, 12(1-2):5–17, 1999.

[49] Mani Subramanian. *Network Management: Principles and Practice*, volume 1. Addison-Wesley, Georgia, 2000.

[50] Andrew S. Tanembaum. *Redes de Computadores*, volume 1. Editora Campus, Rio de Janeiro, terceira edition, 1997.

[51] S. Waldbusser. *Remote Network Monitoring Management Information Base*, February 1995.

[52] K. Yoshihara, K. Sugiyama, H. Horiuchi, and S. Obana. Dynamic polling algorithm based on network management information values. *IEICE TRANS. COMUNN.*, E82-B(6):868–877, 1999.

Apêndice A

Este apêndice contém o código em Prolog Sistema Central de Diagnóstico embutido de algumas funções do Subsistema de Coleta de Status. Este código inclui além da busca binária utilizada nos testes, a implementação dos outros tipos de busca. Várias funções auxiliares estão ao longo do código, no entanto, os principais predicados são: query_manager, poll, e active, explicados um a um posteriormente. O funcionamento geral do sistema tem como início a geração da seqüência de polling e a monitoração da rede através de pings nos dispositivos desta seqüência. A análise da latência dá origem, em caso de falha, ao processo de diagnóstico seja de performance ou falha de comunicação, através da variável "observação". "Poll" implementa ainda o mecanismo de controle de dispositivos em estado falho na sequência.

A hipótese refinada é a solução do probelma de diagnóstico retornada através do predicado "active", ou "performance".

Arquivos auxiliares:

1. "biblioteca" possui uma série de predicados tanto para o parsing de informações coletadas quanto para o cômputo das rotas.

2. "poll" contém os predicados de geração de uma árvore à partir da topologia e o cômputo da seqüência inteligente de polling.

3. "perfor" possui o módulo de diagnóstico de performance.

4. "network" é o modelo lógico-funcional da rede descrito em cláusulas.

Predicados para interpretação do resultado de um ping. Os parâmetros são o IP do dispositivo destino do ping, seu status, número de pacotes perdidos, latências mínima, máxima e média.

```
:-ensure_loaded(biblioteca).
:-ensure_loaded(poll).
:-ensure_loaded(perfor).
:-ensure_loaded(network).


read_result_obs(Device, Status, Num_pack, Min_lat,Max_lat,Time_stamp):-
        read_words(Entrada),
        Entrada = [X,'.',Y,'.',Z,'.',W, Status, Num_pack, Min_lat, Max_lat, Time_stamp],
        Device = ip(X,Y,Z,W).

read_result(ID, Status, Num_pkt, Min_lat, Max_lat,Time_stamp):-
        read_words(Entrada),
        Entrada = [ID, Status, Num_pkt, Min_lat, Max_lat, Time_stamp].
```

Predicado query_manager, solicita o ping em determinado dispositivo durante o processo de diagnóstico. Caso o dispositivo consultado não tenha agente SNMP, um switch ou hub não gerenciável, serão procurados representantes no caminho. Os arquivos request e result são as filas controladas pelo sistema operacional para controle das solicitações e resultados dos pings. O predicado sucede se o dispositivo for alcançável.

Predicado que controla o mecanismo de polling. Itera indefinidamente (repeat) e controla os dispositivos em estado falho na lista de polling.

Predicado para análise da quantidade de pacotes perdidos e da latência. Os thresholds podem ser definidos neste predicado de acordo com cada rede.

Predicado "active" utiliza os algoritmos de cômputo do caminho e faz o refinamento da hipótese inicial.

São implementados, para fins comparativos, quatro tipos de busca inteligente por um elemento do caminho entre Gerente e Destino. 1- intelligent_query_backward: pega sempre o último elemento do caminho de ida entre Gerente e Destino que possua um agente e verifica o status deste elemento. 2- intelligent_query_foward: pega sempre o primeiro elemento do caminho de ida e que seja agente para verificar o status. Possui a vantagem de nao precisar carregar sempre o caminho todo. 3- Busca binária no caminho 4- Busca probabilística no caminho.

Busca binária pelo dispositivo com falha. Deve-se notar que nth_elem retorna um elemento que não seja um cabo.

```
query_manager(Dispositivo,dia(Hipotese,_,_),[ID,Status,Num_pack,Min_lat,Max_lat,Time_s]):-
        (not(dev_agent(Dispositivo))->
        (dev_type(Dispositivo,switch); dev_type(Dispositivo,hub)->!,
                filhos(Dispositivo,Filhos),
                intersection(Filhos,Hipotese,Intersec),
                filhos_no_caminho(Intersec,Lista),
                subtract(Filhos,Lista,Represent),
                subtract(Represent,Hipotese,Repres),
                query_son(Dispositivo,Repres) ) );
        dev_ip(Dispositivo,IP,_),
        iptonumber(IP,Numero),
        tell('/tmp/request'),
        gensym(id,ID),
        format('observation ping ~ w ~ w ~ w ~ w ~ w ~ w ~ n',[ID,Numero,0,2,10,1]),
        told,
        see('/tmp/result'),
        read_result(ID,Status,Num_pack,Min_lat,Max_lat,Time_s),
        seen,
        Status = reachable.
```

```
poll:-
        generate_poll_sequence(Sequence),
        length(Sequence,N),
        repeat, get_time(T1),
        poll(Sequence),
        get_time(T2),
        T_ef is T2 - T1,
        estatisticas(N,T_ef),
        fail.


poll([]).


poll([Head|Tail]):-
        format('~ n-> ping ~ w ~ n',Head),
        (query_manager(Head,_,[_,_,Num_pack,Min_lat,Max_lat,Time_stamp])->
                flag(Head,A,A),
                (A > 1 ->
                format('Dispositivo ~ w retornou ao estado normal',[Head]),
                flag(Head,_,0);
                true),
                format('-> ~ w OK Lat = ~ w ~ n ',[Head,Max_lat]),
                analyse_lat(Head,Num_pack,Min_lat,Max_lat,Time_stamp);
                gerente(G),
                Observacao = obs(los,_,G,Head),
                flag(Head,N,N+1),
                (N = 0 ->
                get_time(T1),
                active(Observacao),
                get_time(T2),
                T_ef is T2 - T1,
                assert(diatime(T_ef)),
                flag(active,Act,Act+1),
                flag(Head,L,L+1);
                (N = 60 ->
                format('Device ~ w ainda com falha',[Head]),
                flag(Head,_,0);
                flag(Head,Z,Z+1) ) ) ),
        sleep(1),
        poll(Tail).
```

```
analyse_lat(Dev,Num_pack,_,Max_lat,_):-
        (Num_pack > 3 ->
                format('~ n--> Alta perda de pacotes ~ n'),
                flag(pkt,Pkt,Pkt+1);
        true),
        dr_threshold(Dev,Threshold),
        Ms is Max_lat / 1000,
        (Ms > Threshold ->
                sleep(60),
                performance(Dev,Resposta),
                flag(performance,Per,Per+1),
                (Resposta = []->
                        true;
                format('~ n Alta taxa de ocupação ~ n',Resposta) );
        true ),
        !.
```

```
active:- poll.


active(Observacao) :-
        format('Iniciando diagnóstico ativo ~ n'),
        get_machine(Observacao,Maquina),
        gerente(Gerente),
        caminho_limpo(Gerente,Maquina,Ida),
        Hipotese = dia(Ida,_,_),
        format('~ n ---------- Busca Binária ----------- ~ n~ n'),
        intelligent_binary_query(Observacao, Hipotese, Resultado),
        limpa(Resultado,Res),
        format('Possível falha em um dos seguintes componente(s): ~ w', Res),
        format('~ n Fim do diagnóstico ativo.~ n~ n').
```

```
intelligent_query_backward(_,dia([X],_,_),dia([X],_,_)).


intelligent_query_backward(obs(los,_,Gerente,Gerente),Hipotese,Hipotese).


intelligent_query_backward(obs(los,_,Gerente,_), Hipotese, Resul):-
        get_last_adjacent(Hipotese,Adjacente),
        caminho_limpo(Gerente,Adjacente,Ida),
        caminho_limpo(Adjacente, Gerente, Volta),
        (subtract(Ida,Volta,[])->
                (query_manager(Adjacente,Hipotese,_) ->
                        is_reachable(Hipotese,Ida,Resul);
                is_unreachable(Hipotese,Ida,Nova_Hipotese),
                intelligent_query_backward(obs(los,_,Gerente,Adjacente),Nova_Hipotese,Resul)
        (query_manager(Adjacente,Hipotese,_) ->
                is_reachable(Hipotese,Ida,Resul);
        split_backward(obs(_,_,Gerente,Adjacente),Ida,Volta,Resul))).


split_backward(obs(_,_,Origem,Destino),Ida,Volta,dia(Resultado,_,_)):-
        format('~ n Diagnosticando Ida ~ n~ n'),
        intelligent_query_backward(obs(_,_,Origem,Destino),
        dia(Ida,_,_),dia(Result,_,_)),
        intersection(Ida,Volta,Inter),
        (membro(Result,Inter)->
                subtract(Volta,Inter,Disjoint),
                format('~ n Diagnosticando Volta ~ n~ n'),
                intelligent_query_foward(obs(_,_,Origem,Destino),
                dia(Disjoint,_,_),
                dia(Resultado,_,_));
        Resultado = Result).
```

```
intelligent_query_foward(_,dia([X],_,_),dia([X],_,_)).


intelligent_query_foward(obs(los,_,Destino,Destino),Hipotese,Hipotese).


intelligent_query_foward(obs(los,_, Gerente, Destino), Hipotese, Resultado):-
        get_first_adjacent(Hipotese, Adjacente),
        caminho_limpo(Gerente,Adjacente,Caminho),
        (query_manager(Adjacente,Hipotese,_)->
                is_reachable(Hipotese,Caminho, Nova_Hipotese),
                intelligent_query_foward(obs(los,_, Adjacente, Destino),Nova_Hipotese,Resultad
        is_unreachable(Hipotese, Caminho, Resultado)).
```

```
intelligent_binary_query(obs(los,_,Gerente,Destino),dia(Lista,_,_), Resultado):-
        length(Lista,N),
        Y is ceiling(N/2),
        nth_elem(Lista,X,Y),
        caminho_limpo(Gerente,X,Caminho),
        (X=Destino->
                Resultado = dia(Lista,_,_);
        (X=Gerente->
                Resultado = dia(Lista,_,_);
        (query_manager(X,dia(Lista,_,_),_) ->
                is_reachable(dia(Lista,_,_),Caminho, Nova_Hipotese),
                intelligent_binary_query(obs(los,_,X,Destino),Nova_Hipotese,Resultado);
        is_unreachable(dia(Lista,_,_), Caminho, Nova_Hipotese),
        intelligent_binary_query(obs(los,_,Gerente,X),Nova_Hipotese,Resultado)))).
```

Busca Probabilística. Note que se houver empate de probabilidades o próximo device vai ser o próximo do caminho.

```
intelligent_probable_query(obs(los,_,Gerente,Destino),Hipotese,Resul):-
        sub(Hipotese,[Gerente,Destino],Hipo),
        get_most_probable(Hipo, Possible),
        (Possible = []->
                Resul = Hipotese;
        caminho_limpo(Gerente,Possible,Caminho),
        (query_manager(Possible,Hipotese,_)->
                is_reachable(Hipotese,Caminho,Nova_Hipotese),
                intelligent_probable_query(obs(los,_,Possible,Destino),Nova_Hipotese,Resul);
        is_unreachable(Hipotese,Caminho,Nova_Hipotese),
        intelligent_probable_query(obs(los,_,Gerente,Possible),Nova_Hipotese,Resul))).
```

Intersecções Apropriadas segundo o Status do dispositivo consultado. "is_reachable" implementa o cômputo $h = h - (h \cap p)$ "is_unreachable" implementa o cômputo $h = h \cap p$. Utilizados em todos os procedimentos de diagnóstico por "reachability".

Predicados utilizados para retornar o próximo elemento a ser inquirido, segundo política de busca. get_last_adjacent unifica Adjacente com o último elemento da hipótese que seja não seja um cabo. get_first_adjacent retorna o próximo elemento do caminho que não seja um cabo. get_most_probable retorna o elemento de maior probabilidade de falha que não seja um cabo.

```
is_reachable(dia(Lista,_,_),Caminho,dia(Nova_Hipotese,_,_)):-
        intersection(Lista,Caminho,Interseccao),
        subtract(Lista,Interseccao,Nova_Hipotese).


is_unreachable(dia(Lista,_,_),Caminho,dia(Nova_Hipotese,_,_)):-
        intersection(Lista, Caminho, Nova_Hipotese).


get_first_adjacent(dia([Adjacente|_Tail],_,_), Adjacente):-
        not(dev_type(Adjacente,cabo)).


get_first_adjacent(dia([_|Tail],_,_),Adjacente):-
        get_first_adjacent(dia(Tail,_,_),Adjacente).


get_last_adjacent(dia(Lista,_,_),Adjacente):-
        pre_last_elem(Lista,Elemento),
        (dev_type(Elemento,cabo)->
                subtract(Lista,[Elemento],Lista2),
                get_last_adjacent(dia(Lista2,_,_),Adjacente);
        Adjacente = Elemento).


get_most_probable(dia([],_,_),[]).


get_most_probable(dia(Lista,_,_),Resultado):-
        predsort(fault_probability,Lista,[Res|_]),
        (not(dev_type(Res,cabo))->
                Resultado = Res;
        subtract(Lista,[Res],Nova_Lista),
        get_most_probable(dia(Nova_Lista,_,_),Resultado)).
```

```
fault_probability(X,Y):-
        dev_type(X,Type),
        dev_type(Y,Type2),
        probability(Type,Prob),
        probability(Type2,Prob2),
        Prob > Prob2.

fault_probability(>,X,Y):-
        dev_type(X,Type),
        dev_type(Y,Type2),
        probability(Type,Prob),
        probability(Type2,Prob2),
        Prob >= Prob2.

fault_probability(<,Y,X):-
        dev_type(X,Type),
        dev_type(Y,Type2),
        probability(Type,Prob),
        probability(Type2,Prob2),
        Prob >= Prob2.
```

```
limpa(dia(X,_,_),[X]).


estatisticas(N,Poll_time):-
        flag(active,Act,Act),
        flag(performance, Perf, Perf),
        flag(pkt, Pkt, Pkt),
        Stat_act is Act - 1,
        Stat_perf is Perf - 1,
        Stat_pkt is Pkt - 1,
        Perc_act is (Stat_act * 100)/N,
        Perc_perf is (Stat_perf * 100)/N,
        Perc_pkt is (Stat_pkt * 100)/N,
        findall(X,diatime(X),Time),
        calculate_mean(Time,Time_mean),
        sum_elements(Time, Time_sum),
        convert_time(Time_mean,_,_,_,_,Min,Sec,Ms),
        convert_time(Time_sum,_,_,_,_,Tot_min,Tot_sec,Tot_ms),
        convert_time(Poll_time,_,_,_,_,M,S,M_sec),
        format('~ n ---------- ~ n'),
        format('~ n Estatísticas de Falhas e Performance: ~ n'),
        format('~ n Percentual de Falhas diagnosticadas: ~ w, ...),
        format('~ n Percentual de Diagnosticos de Performance: ~ w,...),
        format('~ n Percentual de Perda de Pacotes: ~ w,...),
        format('~ n Tempo total em estado de diagnóstico: ~ w,...),
        format('~ n Tempo médio de diagnóstico: ~ w,...),
        format('~ n Tempo total do ciclo de polling: ~ w,...),
        format('~ n ---------- ~ n'),
        flag(active,_,0), flag(performance,_,0), flag(pkt,_,0),
        retractall(diatime/1),
        sleep(60).
```

```
filhos_no_caminho([],[]).

filhos_no_caminho([Head|Tail],Lista):-
        filhos(Head,Lista1),
        filhos_no_caminho(Tail,Lista2),
        append(Lista1,Lista2,Lista).

query_son(Dispositivo,[Filho|Cauda]):-
        (query_manager(Filho,dia([Filho],_,_),_)->
                true;
        query_son(Dispositivo,Cauda) ).

get_machine(obs(_,_,_,Maquina),Maquina).

dr_threshold(Dev,Threshold):-
        dev_dr(Dev,DR,_),
        threshold_dr(DR,Threshold).


membro([],_).


membro([X|Cauda],List):-
        member(X,List),
        membro(Cauda,List).
```

Apêndice B

Este apêndice apresenta o código em Prolog dos predicados para diagnóstico de performance. Estes predicados estão integrados à alguns já apresentados no apêndice anterior.

O diagnóstico de performance irá tratar primeiramente do problema de latência, ou seja, quando a demora pela resposta de um ping ultrapassa um "threshold" de tempo. Para isso, os caminhos entre gerente e destino irão considerar os domínios de repetição por onde passam os pacotes.

O sistema faz a verificação de taxa de ocupação nos domínios de colisão (DR): neste caso o diagnóstico de performance será executado quando o módulo de falhas (ativo) encontrar uma latência elevada. Então, o módulo de performance percorre os DRs (Domínios de Colisão) perguntando sua taxa de ocupação.

```
performance(Dev,Resposta):-
        gerente(Gerente),
        caminho_limpo(Gerente,Dev,Cam),
        limpa_cabo(Cam, Caminho),
        path_to_dr(Caminho,Drep),
        list_to_set(Drep,DR),
        performance_ocupacao(DR,Resposta).


Caminha pela lista de DRs procurando pelo
domínio responsável pelo problema de performance
performance_ocupacao([],[]).


performance_ocupacao([DR|Cauda],Resposta):-
        (dev_type(DR,switch)->
                Queryable = DR;
        dev_dr(_,DR,Queryable)),
        (rmon_ask(Queryable)->
Sucede se a taxa de ocupacao for normal
                performance_ocupacao(Cauda,Resposta);
        Resposta=DR).
```

Faz a solicitação de busca de informações na MIB-RMON para um IP específico

```
rmon_ask(Queryable):-
        dev_ip(Queryable,IP,_),
        tell('/tmp/request'),
        gensym(id,ID),
        iptonumber(IP,Num),
        format('observation performance  w  w  n', [ID,Num]),
        told,
        see('/tmp/result'),
        rmon_parse(Usage_mean),
        seen,
        calculate_mean(Usage_mean,Mean),
        MEAN is Mean/100,
        (MEAN > 0.7 ->
                fail;
        true ).
```

Media aritimética simples de elementos (inteiros em uma lista).

```
calculate_mean(List,Mean):-
        length(List,N),
        sum_elements(List,Sum),
        Mean is Sum/N.


sum_elements([X],X).


sum_elements([Head|Tail],Sum):-
        sum_elements(Tail,Temp),
        Sum is Temp + Head.


path_to_dr([],[]).


path_to_dr([Head|Tail],[Head|Result]):-
        (dev_type(Head,switch);dev_type(Head,router)),
        path_to_dr(Tail,Result).


path_to_dr([Head|Tail],[DR|Result]):-
        dev_dr(Head,DR,_),
        path_to_dr(Tail,Result).
```

Apêndice C

Rede criada parcimoniosamente, emulando a rede completa do LSI com finalidade de testes reais possui todas as informações de IP de redes e um domínio de colisão completo.

Definição das Redes
net(net1, ip(143, 107, 161, 128), mask(255, 255, 255, 128)).
net(net2, ip(10, 0, 160, 0), mask(255, 255, 254, 0)).
net(net3, ip(10, 0, 10, 0), mask(255, 255, 255, 0)).
net(net4, ip(192, 168, 10, 0), mask(255, 255, 255, 0)).
net(net5, ip(192, 168, 200, 0), mask(255, 255, 255, 0)).
net(net6, ip(192, 168, 210, 0), mask(255, 255, 255, 0)).
net(net7, ip(143, 107, 160, 0), mask(255, 255, 255, 0)).

Identificação do dispositivo gerente
gerente(dev25).

Identificação dos nomes
dev_name(dev84, 'gateway.lsi.usp.br').
dev_name(dev13, aeolus).
dev_name(dev14, apolo).
dev_name(dev99, 'galaxy2.intranet').
dev_name(dev126, 'ASX-1000_1').
...
Dispositivos e seus IPs
dev_ip(dev1,ip(10,0,160,35),net2).
dev_ip(dev2,ip(10,0,160,36),net2).
dev_ip(dev3,ip(10,0,160,37),net2).
dev_ip(dev4,ip(10,0,160,38),net2).
dev_ip(dev5,ip(10,0,160,39),net2).
dev_ip(dev6,ip(10,0,160,42),net2).
dev_ip(dev7,ip(10,0,160,43),net2).
dev_ip(dev8,ip(10,0,160,46),net2).
dev_ip(dev9,ip(10,0,160,47),net2).
dev_ip(dev10,ip(10,0,160,48),net2).
dev_ip(dev11,ip(10,0,160,15),net2).
dev_ip(dev12,ip(10,0,160,137),net2).
dev_ip(dev13,ip(10,0,160,133),net2).
dev_ip(dev14,ip(10,0,160,158),net2).
...

```
Tipos dos dispositivos
dev_type(dev1,hub).
dev_type(dev2,hub).
dev_type(dev3,hub).
...
dev_type(dev11,switch).
...
dev_type(dev12,computador).
dev_type(dev13,computador).
dev_type(dev14,computador).
dev_type(dev15,computador).
...
dev_type(c(_,_),cabo).

Dispositivos com agente SNMP
dev_agent(dev1).
dev_agent(dev2).
dev_agent(dev3).
...
Conexão física entre dispositivos
l(dev1,dev6).
l(dev1,dev12).
l(dev1,dev13).
l(dev1,dev14).
l(dev2,dev15).
...
Quais dispositivos podem estar desligados
desligavel([]).

Domínios de Colisão
dev_dr([dev1,dev6,dev7,dev12,dev13,dev14,dev25,dev26,dev27,dev28],dr1,dev7).
dev_dr([dev2,dev15,dev16,dev17,dev18],dr2,dev2).
dev_dr([dev3],dr3,dev3). dev_dr([dev4,dev19],dr4,dev4).
dev_dr([dev5,dev20,dev21,dev22,dev23,dev24],dr5,dev5).
...

dev_dr(X,Y,_):-
        dev_dr(Lista,Y,_),
        member(X,Lista).

lat_rate(dr1,30).
lat_rate(dr2,30).

treshold_dr(dr1,700).
treshold_dr(dr2,700).
```

Tabelas de Roteamento

rotax(ip(10, 0, 10, 157), ip(0, 0, 0, 0), mask(0, 0, 0, 0), ip(10,0, 10, 1), 1, 3).
rotax(ip(10, 0, 10, 157), ip(10, 0, 10, 0), mask(255, 255, 255,0), ip(0, 0, 0, 0), 0, 3).
rotax(ip(10, 0, 10, 157), ip(127, 0, 0, 0), mask(255, 0, 0, 0),ip(0, 0, 0, 0), 0, 1).
rotax(ip(10, 0, 10, 157), ip(127, 0, 0, 2), mask(255, 255, 255,255), ip(0, 0, 0, 0), 0, 2).
rotax(ip(10, 0, 10, 199), ip(0, 0, 0, 0), mask(0, 0, 0, 0), ip(10,0, 10, 1), 1, 2).
rotax(ip(10, 0, 10, 199), ip(10, 0, 10, 0), mask(255, 255, 255,0), ip(0, 0, 0, 0), 0, 2).
rotax(ip(10, 0, 10, 199), ip(127, 0, 0, 0), mask(255, 0, 0, 0),ip(0, 0, 0, 0), 0, 1).
rotax(ip(10, 0, 10, 2), ip(0, 0, 0, 0), mask(0, 0, 0, 0), ip(10,0, 10, 1), 1, 2).
...