Um Esquema para o Gerenciamento do Tráfego de Aplicações em Redes TCP-IP

Carlos Kelner Silveira

Um Esquema para o Gerenciamento do Tráfego de Aplicações em Redes TCP-IP

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. Carlos Kelner Silveira e aprovada pela Comissão Julgadora.

Campinas, 7 de abril de 1995.

Edmundo R M Madeira

Prof. Edmundo R. M. Madeira
Orientador

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

Tese defendida e aprovada em, 07 de ABRIL de 1995

Pela Banca Examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). CARLOS BECKER WESTEPHALL

Prof(a). Dr(a). PAULO LICIO DE GEUS

Prof(a). Dr(a). EDMUNDO ROBERTO MAURO MADEIRA

Aos meus Pais e Avós

Agradecimentos

Ao Prof. Edmundo Madeira pela atenção e interesse com que sempre me recebeu. Um exemplo de competência e simplicidade.

Ao Prof. Oswaldo Franzin pela ajuda desde o início do projeto, sem a qual não alcançaríamos os resultados obtidos.

Aos Professores Paulo Lício e Paulo Centoducatte pelo apoio e facilidade de uso dos equipamentos de rede do DCC e esclarecimento de dúvidas.

Aos funcionários do DCC, especialmente, Luiz e Alda, pela ajuda sempre prestativa.

A todo o pessoal do DCC pelo ambiente amigo que sempre encontrei.

Ao pessoal da Engenharia Elétrica, com quem convivi no ano de 1993 e fiz muitos amigos.

Às muitas pessoas que me ajudaram, em especial, ao Carrilho, influência forte no trabalho aqui exposto e ajuda constante, à Karen, pelo grande apoio nas atividades de suporte, a Fábio, Ronaldo, Nabor, Victor e Nuccio, entre outros, que sempre estiveram prontos a discutir dúvidas e soluções.

A todos os amigos com quem convivi e que ajudaram a diminuir as saudades do Recife.

À minha querida Adriana que acreditou, mesmo contra toda "lógica", na construção de um amor.

À minha mãe, presença constante em minha vida com amor, inteligência e incentivo que muito influenciaram minha formação.

Ao meu pai, pelo incentivo ao desafio desde cedo e exemplo de competência e tranqüilidade. Seu amor e confiança em mim são fontes de segurança.

Aos meus avós Miriam e Salomão, exemplos de luta, correção, dedicação e amor. Vocês são responsáveis diretos por todas as vitórias obtidas por nós, seus descendentes.

Resumo

Com a utilização cada vez maior das redes de computadores, as pesquisas e produtos relacionados ao gerenciamento de redes têm crescido bastante. Nesse contexto, o ambiente de gerenciamento de redes SNMP (Simple Network Management Protocol) tem sido muito utilizado, surgindo como um padrão *de facto*.

Esta dissertação apresenta um esquema para o gerenciamento do tráfego de aplicações entre grupos de computadores em redes TCP-IP. Foi desenvolvido um agente de gerenciamento, seguindo o modelo baseado no protocolo SNMP, que coleta os dados (número de bytes e datagramas por aplicação) e responde às requisições de gerentes SNMP. Alguns dados reais foram obtidos, usando-se o software de gerenciamento SunNet Manager e o agente desenvolvido, com o intuito de validar o esquema e verificar as vantagens de tal tipo de gerenciamento.

Abstract

As computer networks are becoming more widespread, research and products related to network management are growing very fast and the Internet Management Framework, based on the SNMP (Simple Network Mangement Protocol), is the most used, appearing as a *de facto* standard.

This dissertation presents a scheme for the management of application traffic among groups of computers in TCP-IP networks. A management agent was developed, based on the SNMP management model, which collects data (bytes and datagrams per application) and replies requests from SNMP managers. Using the software SunNet Manager and the developed agent, some real data were obtained to validate the scheme and verify the advantages of this kind of management.

Lista de Abreviações

ARPANET: Advanced Research Projects Agency Network
ASN.1: Abstract Syntax Notation One
CMIP: Common Management Information Protocol
FTP: File Transfer Protocol
IAB: Internet Architecture Board
IANA: Internet Assigned Number Authority
IETF: Internet Engineering Task Force
IP: Internet Protocol
ISO: International Organization for Standardization
MIB: Management Information Base
NFS: Network File System
NSF: National Science Foundation
OSI: Open Systems Interconnection
RPC: Remote Procedure Call
SMI: Structure of Management Information
SMTP: Simple Mail Transfer Protocol
SNMP: Simple Network Management Protocol
SNMPv2: Simple Network Management Protocol Version 2
TCP: Transmission Control Protocol
TFTP: Trivial File Transfer Protocol
UDP: User Datagram Protocol
WWW: World Wide Web

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

As redes de computadores estão sendo cada vez mais utilizadas e a produtividade atingida com sua utilização, seja para ambientes eletrônicos de trabalho em grupo ou para compartilhamento de recursos, tornou-as indispensáveis para organizações eficientes.

Essa necessidade incentivou o surgimento de pesquisas na área de gerenciamento de redes, que originou dois modelos básicos: um baseado no CMIP (*Common Management Information Protocol*) proposto pela ISO (*International Organization for Standardization*) para ambiente OSI (*Open Systems Interconnection*) e outro baseado no SNMP (*Simple Network Management Protocol*) [CFSD90, Ros91], usado na Internet. Pela vasta quantidade de produtos que seguem o modelo baseado no SNMP, pode-se considerá-lo um padrão *de facto*.

O SNMP possibilita o gerenciamento através de gerentes e agentes de gerenciamento. Os agentes contêm bases de dados de gerenciamento que podem ser lidas (monitoração) ou escritas (controle) pelos gerentes. Portanto, neste esquema, a base de dados de gerenciamento desempenha um papel fundamental, sendo o fator determinante do escopo do gerenciamento. Foi definido para o ambiente de gerenciamento SNMP uma base de dados de gerenciamento padrão contendo informações diversas de um elemento de rede, entre elas, tabelas de rotas, conexões de transporte abertas e pacotes que chegam à camada de rede com erro. Entretanto, a base de dados de gerenciamento padrão não contém dados a respeito de protocolos de aplicação. Sentindo-se a necessidade desse tipo de informação, este trabalho propõe um esquema para gerenciar o tráfego de aplicações entre grupos de computadores em redes TCP-IP (*Transmission Control Protocol - Internet*

Protocol).

O objetivo do esquema proposto é monitorar o tráfego entre grupos de computadores em redes TCP-IP. Estes grupos podem estar em qualquer lugar da Internet, desde que se observem algumas restrições definidas adiante. Além disso, o tráfego pode ser classificado pela aplicação que o gerou (correio eletrônico, transferência de arquivos, entre outros).

As vantagens propiciadas por este gerenciamento são a identificação do tráfego dos diversos segmentos de uma dada rede, ou redes, possibilitando a localização de "gargalos". Essa localização pode ser bastante útil para a relocação de servidores diversos (de arquivos, de impressão, entre outros) e para a decisão de formar sub-redes.

No capítulo 2, descreve-se o conjunto de protocolos TCP-IP, no capítulo 3, o gerenciamento de redes, em particular, o ambiente de gerenciamento SNMP. No quarto capítulo, o modelo de gerenciamento de tráfego de aplicações é exposto, no quinto capítulo, a implementação de um protótipo é descrita e no último capítulo, a conclusão é apresentada.

Capítulo 2

Protocolos TCP-IP

Em meados da década de 60, o departamento de defesa dos Estados Unidos começou a incentivar pesquisas para viabilizar a interligação de computadores. Surgiu então a ARPANET (*Advanced Research Projects Agency Network*). A ARPANET deveria prover a comunicação entre computadores permitindo uma grande variedade de interações: login remoto; compartilhamento de arquivos e outros recursos e, embora não constasse do planejamento inicial, correio eletrônico [LR93].

Para permitir este tipo de interligação, os pesquisadores tiveram que criar um conjunto de protocolos que pudesse ser utilizado pelos diversos fabricantes de computadores. Esse conjunto foi o TCP-IP, que foi projetado nos anos 1973/74.

A ARPANET cresceu muito e em 1985 e 86 uma organização americana de incentivo à ciência, a NSF (*National Science Foundation*), financiou a proliferação da tecnologia para as universidades americanas.

Hoje em dia, universidades, centros de pesquisa, órgãos governamentais, empresas e outras organizações do mundo inteiro estão conectadas através desta grande rede que passou a se chamar Internet.

Alguns dos principais serviços da Internet são *gopher*, *www*, correio eletrônico e *login* remoto.

A atual política de controle de tecnologia e padronização da Internet é realizada pelo IAB (*Internet Architecture Board*), que é composto de um organismo responsável por pesquisas de longo prazo, o IETF (*Internet Engineering Task Force*), e duas secretarias, financiadas pelo governo americano, que dão suporte administrativo à comunidade da Internet.

A Internet já é utilizada por mais de quatro milhões de pessoas em mais de 100 países e o conjunto de protocolos TCP-IP tornou-se um padrão na prática.

2.1 Endereço Internet

Para que seja possível a comunicação entre quaisquer dois computadores conectados à Internet, tem de haver uma maneira única de identificar cada computador. Isto é feito através da atribuição de um número a cada computador, o endereço Internet.

Os endereços Internet, também chamados de endereços IP, são números binários, compactos e cuidadosamente escolhidos de modo que o roteamento seja eficiente. Um endereço IP é um número de 32 bits, que denota um par **(netid, hostid)** onde: **netid** identifica uma rede e **hostid** identifica um *host* nessa rede. Um endereço IP tem uma das formas apresentadas na figura 2.1 [Com91].

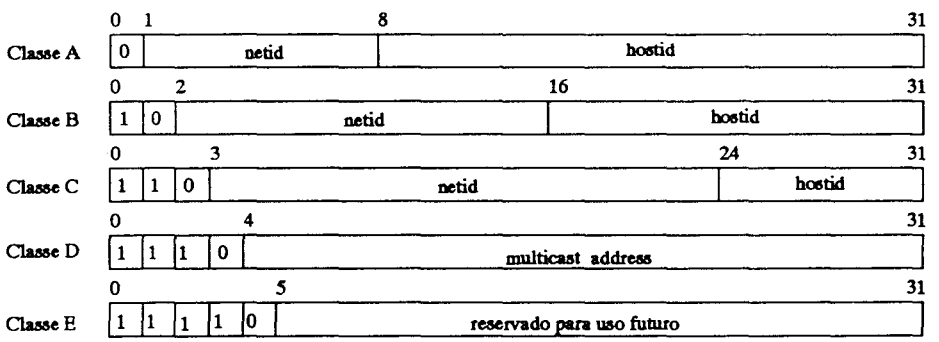

Figura 2.1: Classes de Endereços IP

Dado um endereço IP, distingue-se entre as três classes primárias, endereços classe A, classe B ou classe C, através dos dois primeiros bits. O que diferencia as classes é o número de redes endereçáveis e o número de máquinas por rede, sendo a classe A aquela que endereça menos redes (2^7) com mais *hosts* (2^{24}) e a classe C mais redes (2^{21}) e menos *hosts* por rede (2^8).

Para simplificar a leitura, representa-se o endereço IP por quatro números decimais, separados por pontos, cada um representando o valor de um octeto do endereço IP, formando assim 32 bits. A figura 2.2 apresenta alguns exemplos.

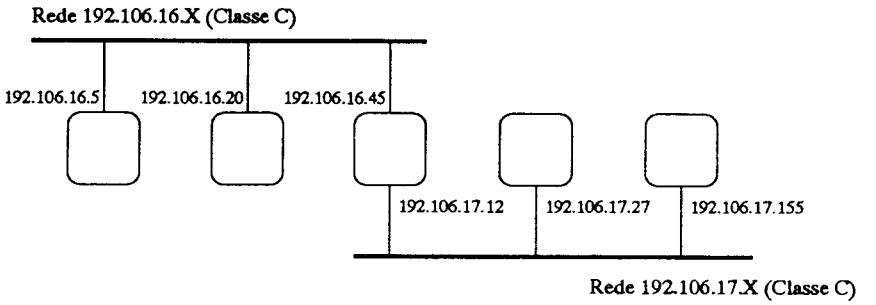

Figura 2.2: Exemplos de Endereços IP

2.2 Arquitetura de Camadas TCP-IP

O conjunto de protocolos TCP-IP é organizado, conceitualmente, em quatro camadas [Com91, LR93].

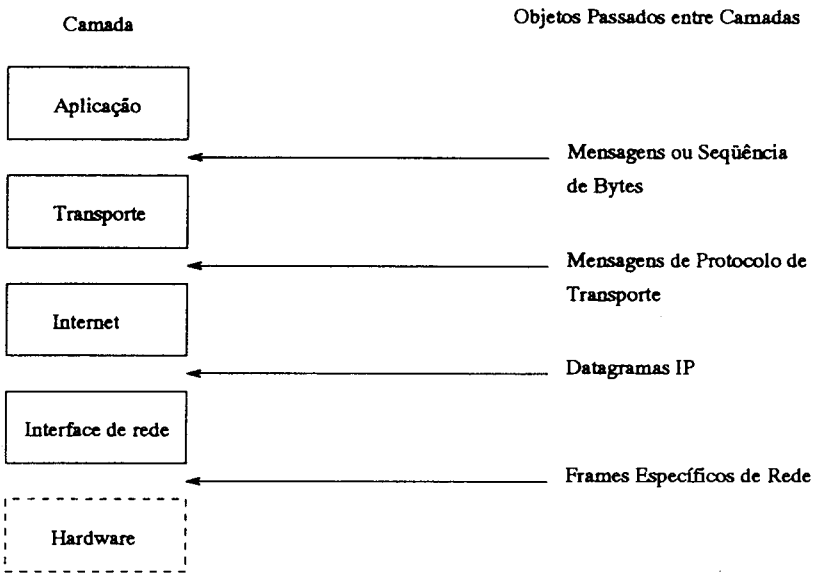

Figura 2.3: Arquitetura de Camadas TCP-IP

- **Camada de Aplicação**: No nível mais alto, usuários invocam programas de aplicação que acessam serviços de comunicação da Internet. Cada protocolo de aplicação escolhe o protocolo de transporte que se adapte às

suas necessidades, podendo ser uma seqüência de mensagens individuais ou ainda uma seqüência contínua de bytes.

- **Camada de Transporte**: A função primordial da camada de transporte é prover comunicação entre duas entidades de aplicação (comunicação *end-to-end*). A camada de transporte pode regular o fluxo de informação e fornecer um serviço confiável de transporte assegurando que os bytes enviados chegam sem erros e na seqüência correta. Para tanto, o lado destinatário deve enviar *acknowledgements* e o remetente retransmitir segmentos perdidos ou danificados. O software de transporte divide a seqüência de bytes em pequenos pedaços (mensagens) que são passados, juntamente com um endereço de destino, à próxima camada.

- **Camada Internet**: Processa a comunicação de uma máquina a outra. A camada Internet (camada de rede, na terminologia ISO-OSI) recebe uma requisição para enviar um pacote da camada de transporte juntamente com o endereço da máquina destino. Ela encapsula o segmento de transporte em um datagrama Internet, também chamado de datagrama IP, preenche o cabeçalho e usa o algoritmo de roteamento para determinar para onde enviar o datagrama. A camada Internet também processa datagramas recebidos, checando sua validade e usando o algoritmo de roteamento para saber se o datagrama deve ser processado localmente ou passado adiante. Se o datagrama for destinado à máquina local o software da camada IP retira o cabeçalho e passa para o protocolo de transporte correspondente o segmento de transporte.

- **Camada de Interface de Rede**: O software de nível mais baixo no TCP-IP é responsável por aceitar datagramas IP e transmiti-los em um formato específico de rede. É uma camada dependente do meio.

2.3 Internet Protocol (IP)

O serviço mais fundamental da Internet consiste de um sistema de transmissão de pacotes. É definido como um serviço sem conexão e não confiável. É dito não confiável porque a entrega não é garantida, podendo o pacote ser perdido ou atrasado e o serviço não informa a remetentes ou destinatários caso isso aconteça. É dito sem conexão porque cada pacote é tratado independentemente dos outros, podendo em uma seqüência de pacotes de uma máquina a outra alguns pacotes seguirem diferentes caminhos ou serem perdidos e outros não, por exemplo. Além disso, a ordem da seqüência não é garantida.

O IP [Pos81a, Alm92] provê a transmissão de datagramas de uma máquina origem a uma destino, possivelmente passando por um ou mais roteadores e diferentes redes. Um datagrama é um pacote de tamanho finito contendo um cabeçalho e uma parte de dados úteis.

Tanto *hosts* quanto roteadores estão envolvidos no processamento de cabeçalhos IP. Os *hosts* devem criá-los no envio e processá-los no recebimento, enquanto que os roteadores devem examiná-los para tomar decisões de roteamento e modificá-los quando necessário.

A estrutura do datagrama é apresentada na figura 2.4 [LR93].

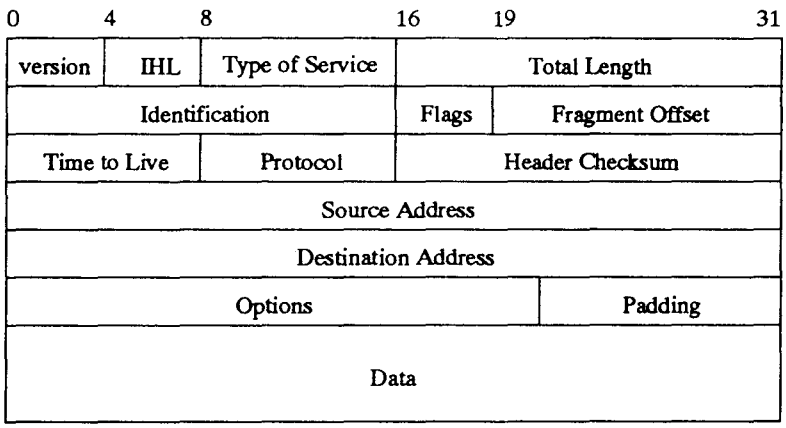

Figura 2.4: Estrutura do Datagrama IP

- **Version**: contém a versão do protocolo IP utilizado para criar o datagrama; é usado para verificar que o remetente, o destinatário e os roteadores estão em concordância com o formato do datagrama.
- **IHL (Internet Header Length)**: é o comprimento do cabeçalho IP em palavras de 32 bits; o mais usual é termos IHL igual a cinco, quando o campo de opções está vazio.
- **Type of Service**: especifica como o datagrama deve ser processado (ver figura 2.5):
  - **Precedence**: indica um esquema de prioridade variando de 0 (precedência normal) até 7 (controle de rede);
  - **D**: requer pequeno atraso no processamento (D=1);

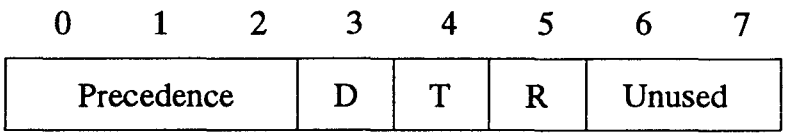

Figura 2.5: Campo *Type of Service*

- **T**: requer grande vazão de dados (T=1);
- **R**: requer alta confiabilidade (R=1).

- **Identification**: número inteiro que identifica unicamente um datagrama ou seus fragmentos.
- **Flags**: o primeiro bit é sempre 0; o segundo indica se o datagrama é ou não fragmentável; é chamado de *do not fragment* pois se tiver valor igual a 1 não permite a fragmentação do datagrama; o terceiro bit indica se o fragmento contém dados do meio do datagrama original ou do final dele; é chamado de *more fragments* pois se tiver valor igual a 1 sabe-se que se trata de dados do meio do datagrama original.
- **Fragment Offset**: especifica o offset em relação ao início do datagrama original em unidades de oito bytes, para o caso de fragmentos.
- **Time to Live**: especifica por quantos roteadores, no máximo, o datagrama pode ser processado; roteadores devem decrementar este campo para cada datagrama; tem a utilidade de descartar datagramas circulando na rede por falha de roteamento.
- **Protocol**: especifica que protocolo de nível superior está sendo usado: TCP (6) ou UDP (17), na maior parte dos casos.
- **Header Checksum**: serve para teste de integridade do cabeçalho IP; o *checksum* é calculado obtendo-se a soma, em complemento a um, do cabeçalho como uma seqüência de inteiros (palavras de 16 bits) e então fazendo-se o complemento a um do resultado.
- **Source IP Address**: endereço IP do remetente do datagrama.
- **Destination IP Address**: endereço IP do destinatário do datagrama.
- **Options**: o campo de opções é de tamanho variável, mas sempre ajustado através do **padding** para conter múltiplos de 32 bits; são usados principalmente para testes com a rede.

O roteamento IP consiste em decidir aonde enviar um datagrama, tomando-se por base o seu endereço IP de destino. O envio é direto se a máquina destino está na mesma rede física que a máquina origem, caso contrário, o roteamento é necessário. Em geral os *hosts* roteiam datagramas indiretamente para o roteador mais próximo e o datagrama vai de roteador em roteador até chegar à rede física de destino.

O roteamento IP produz o endereço IP da próxima máquina (*next hop*) para a qual o datagrama deve ser enviado.

O algoritmo de roteamento IP usa somente endereços IP. Ele baseia decisões de roteamento no endereço da rede de destino, ao invés do *host* de destino, o que diminui muito o tamanho das tabelas de roteamento.

2.4 User Datagram Protocol (UDP)

No conjunto de protocolos TCP-IP, o UDP [Pos80] é o protocolo de transporte que provê o mecanismo mais simples que protocolos de aplicação podem usar para enviar dados a outros programas de aplicação. O UDP distingue entre diferentes aplicações em uma máquina através de campos chamados de portas (*ports*). Cada mensagem UDP contém um número de porta de origem e um de destino tornando possível a comunicação entre duas aplicações particulares.

O UDP usa o IP e tem as mesmas características de não confiabilidade e serviço sem conexão. Assim, é de responsabilidade das camadas acima da camada de transporte resolver problemas como perda, duplicação, atraso e troca na seqüência de chegada de mensagens.

O pacote UDP tem o seguinte formato mostrado na figura 2.6 [LR93].

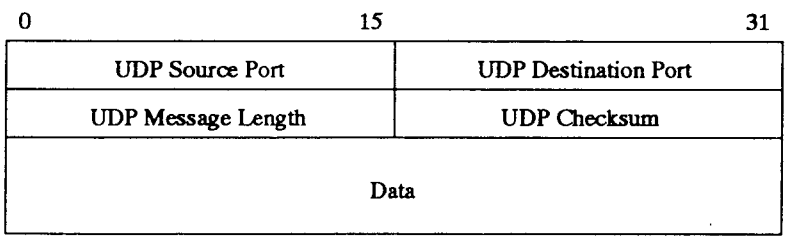

Figura 2.6: Formato do Pacote UDP

- **UDP Source Port**: número de 16 bits identificando a aplicação na máquina remetente.

Código da Porta (Servidor)	Palavra Chave	Descrição
0	-	Reservado
11	USERS	Usuários ativos
13	DAYTIME	Hora corrente
42	NAMESERVER	Servidor de Nomes
69	TFTP	Trivial File Transfer
111	SUNRPC	Sun Microsystems RPC
161	SNMP	Gerente SNMP
162	SNMP	Traps SNMP
513	WHO	UNIX rwho daemon

Tabela 2.1: Algumas portas reservadas para o UDP

- **UDP Destination Port**: número de 16 bits identificando a aplicação na máquina destino.
- **UDP Message Length**: indica o comprimento do pacote UDP, incluindo o cabeçalho e os dados, em unidades de oito bits.
- **UDP Checksum**: é opcional e um valor igual a zero significa que não foi calculado. Isso acontece para possibilitar implementações com um mínimo de sobrecarga.

Existem alguns números de portas que são padronizados, para que os serviços mais conhecidos sejam acessados com maior rapidez, sem que se precise uma combinação entre os protocolos de aplicação antes do início da transmissão propriamente dita. Um exemplo de uma aplicação que usa uma porta padronizada é o TFTP (Trivial File Transfer Protocol). Assim, sempre que quisermos acessar um servidor TFTP já saberemos em que porta ele estará (ver tabela 2.1).

2.5 Transmission Control Protocol (TCP)

Em um nível mais baixo, a comunicação entre computadores é não confiável. Pacotes podem ser perdidos quando erros de transmissão interferem com os dados, quando o hardware falha ou ainda quando as redes se tornam tão carregadas que não conseguem transmitir a carga desejada. Além disso, como as redes roteiam pacotes dinamicamente, pode haver inversões de ordem.

Em um nível mais alto, as aplicações enviam grandes volumes de dados de um computador para outro. Se todo protocolo de aplicação tiver de se ocupar

em saber se os pacotes enviados chegaram intactos e na ordem correta, teremos um trabalho além de extremamente tedioso sujeito à falhas de programação cada vez que quisermos fazer uma aplicação que se comunique com outro computador em rede.

Para evitar esses problemas, projetou-se um protocolo de transporte que provê transmissão de seqüência de bytes confiável. Esse protocolo é o TCP [Pos81b].

Como o UDP, o TCP está uma camada acima do IP. Também como o UDP, o TCP usa portas para identificar as aplicações que estão se comunicando. Contudo as portas do TCP são mais complexas, pois na verdade representam conexões, que são identificadas por um par de pontos. Isso é equivalente a dizer que existe uma abstração de um circuito virtual entre duas aplicações quando estas usam o TCP. Um par de pontos são identificados por duplas (máquina, porta).

Usando essa abstração e mais um conjunto de campos, o TCP pode implementar os dispositivos de controle de fluxo, retransmissão e reconhecimento de pacotes que o fazem confiável.

A unidade de transferência entre as camadas TCP em duas máquinas é chamada de segmento. Segmentos são trocados para estabelecer conexões, transferir dados, enviar *acknowledgements*, fechar conexões e acertar parâmetros.

O segmento TCP é apresentado na figura 2.7 [LR93]:

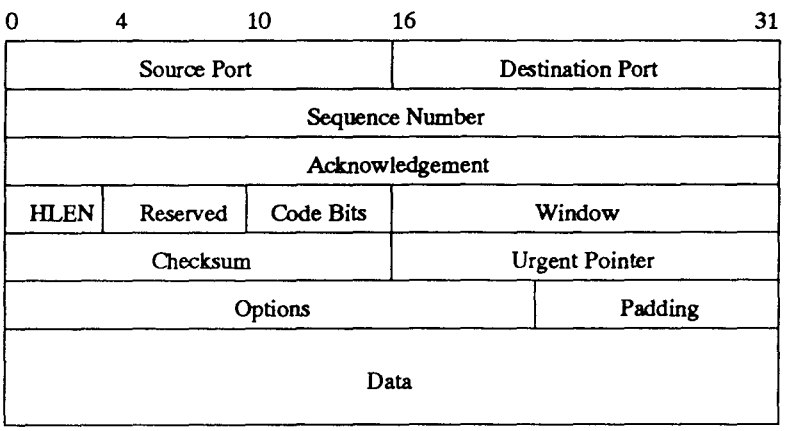

Figura 2.7: Formato do Segmento TCP

- **Source Port**: número que identifica a porta TCP remetente.
- **Destination Port**: número que identifica a porta TCP destinatária.

Cod. Porta (Servidor)	Chave	Descrição
0	-	Reservado
11	USERS	Usuários ativos
13	DAYTIME	Hora corrente
20	FTP-DATA	Dados de controle do File Transfer Protocol
21	FTP	Dados de arquivos do File Transfer Protocol
23	TELNET	Conexão remota de terminal
25	SMTP	Correio eletrônico
42	NAMESERVER	Servidor de nomes
70	GOPHER	servidor Gopher
79	FINGER	Finger
80	WWW	Servidor WWW
111	SUNRPC	SUN Microsystems RPC

Tabela 2.2: Algumas portas reservadas para o TCP

- **Sequence Number**: usado para o controle de seqüência dos dados de aplicação.
- **Acknowledgement**: usado para o reconhecimento de segmentos recebidos.
- **HLEN**: comprimento do cabeçalho do segmento em palavras de 32 bits.
- **Code Bits**: definem regras de prioridade no tratamento do segmento.
- **Window**: controla o tamanho da janela utilizada pela conexão.
- **Checksum**: utilizado para checar a integridade do segmento.
- **Urgent Pointer**: aponta para o lugar em que se localizam os dados urgentes; é válido caso o *Code Bit* mais significativo (URG) seja igual a um.

Como o UDP, o TCP tem algumas portas reservadas para aplicações largamente usadas (*well-known ports*). Alguns exemplos são apresentados na tabela 2.2.

2.6 Protocolos de Aplicação

Muito da funcionalidade associada ao conjunto de protocolos TCP-IP é devido a serviços providos por utilitários como o **mailtool** e **mosaic**. Os protocolos de

aplicação associados a estes programas usam os serviços de transporte do TCP-IP: entrega não confiável de datagramas (UDP) e entrega confiável de datagramas (TCP). Os protocolos geralmente seguem o modelo cliente-servidor, onde, como mencionado, os servidores operam em portas conhecidas de modo que os clientes possam acessá-los.

A grande vantagem de usar os serviços de transporte do TCP-IP é a simplificação dos protocolos de aplicação. Protocolos de transporte fim-a-fim podem garantir que um programa cliente em uma máquina fonte comunique-se diretamente com o servidor na máquina destino.

A figura 2.8 [Com91] mostra alguns dos protocolos de aplicação mais conhecidos.

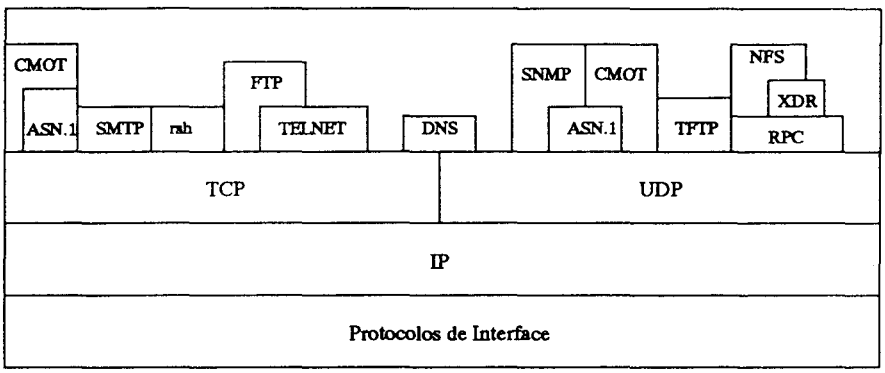

Figura 2.8: Alguns protocolos de aplicação e suas dependências

Alguns dos mais utilizados são:

- **Telnet**: permite a um usuário estabelecer uma conexão com um servidor de *login* remoto, passando então comandos do terminal do usuário diretamente para a máquina remota; é um serviço transparente pois dá a impressão que o terminal do usuário está diretamente conectado à máquina remota [PR83].

- **File Transfer Protocol (FTP)**: protocolo para transferência de arquivos entre máquinas [PR85].

- **Simple Mail Transfer Protocol (SMTP)**: protocolo para transferência de mensagens (correio eletrônico) entre máquinas da rede; difere da transferência de arquivos pois não exige uma comunicação fim-a-fim. A transferência é feita de modo indireto, mesmo a nível de aplicação, quando não

há conectividade fim-a-fim. Isso explica como algumas redes têm conectividade para correio eletrônico com a Internet e não para outros protocolos [Pos82].

- **Simple Network Management Protocol (SNMP)**: protocolo usado para o gerenciamento de redes; este protocolo será comentado no próximo capítulo.

Capítulo 3

Gerenciamento de Redes

3.1 Introdução

Existem muitas definições para o gerenciamento de redes, entre elas, podemos destacar a definição dada pela ISO. Segundo a ISO, o gerenciamento de redes provê mecanismos para a monitoração, controle e coordenação de recursos em um ambiente OSI e define padrões de protocolos OSI para troca de informações entre esses recursos [Car93].

Com o crescimento das redes e, principalmente, de sua importância para as organizações, o gerenciamento de redes tornou-se vital devido ao custo das redes e ao fato do uso de seus serviços estar diretamente ligado à disponibilidade e eficiência dos sistemas aplicativos das empresas.

O gerenciamento de redes envolve principalmente cinco áreas [Sta93, Car93]:

- **Gerenciamento de Falhas**: Possibilita a deteção, isolamento e correção de operações anormais em um ambiente distribuído. Para que se mantenha uma rede complexa em operação deve-se observar o sistema como um todo e os componentes mais importantes individualmente. Quando uma falha ocorre é importante, no menor tempo possível, que:
  - determine-se com exatidão onde a falha ocorreu;
  - isole-se o restante da rede da falha para que o ambiente continue a funcionar sem interferência;

  - reconfigure-se ou modifique-se a rede de modo a minimizar os efeitos danosos causados pela retirada de operação do(s) componente(s) faltoso(s);
  - repare(m)-se o(s) componente(s) faltoso(s) para reconduzir a rede ao estado plenamente operacional.

- **Gerenciamento de Configuração**: Exerce controle, identifica, coleta dados, provê dados, com o objetivo de oferecer de maneira contínua os serviços da rede. O gerenciamento de configuração tem como encargos a iniciação da rede e a retirada de componentes de modo "suave". E ainda, manter, adicionar e atualizar componentes e seus relacionamentos.

- **Gerenciamento de Contabilidade**: Possibilita o estabelecimento de tarifas para o uso de objetos gerenciados e custos a eles correspondentes. É de vital importância para que os custos e o volume de recursos utilizados pelos usuários sejam identificados e registrados de forma correta. A contabilidade abrange todos os recursos passíveis de monitoração. O gerenciamento de contabilidade dedica-se, principalmente, a mensurar o conjunto de facilidades de rede utilizadas.

- **Gerenciamento de Desempenho**: Possibilita a avaliação do comportamento de objetos gerenciados e da eficiência das atividades relacionadas à comunicação. O gerenciamento de desempenho de uma rede de computadores compreende duas categorias principais: monitoração e controle. A monitoração é a função que registra as atividades da rede. A função de controle possibilita ao gerenciamento de desempenho a realização de ajustes que melhorem o desempenho da rede. Alguns dos itens relacionados ao desempenho são:

  - capacidade de utilização;
  - tráfego excessivo;
  - vazão;
  - "gargalos";
  - tempo de resposta.

  Para que se observem esses itens, o gerente de rede deve definir um conjunto de recursos a serem monitorados. Para que níveis de desempenho possam

ser definidos, deve-se associar uma métrica adequada aos recursos monitorados. Por exemplo, quantas retransmissões em uma conexão de transporte caracterizam um baixo desempenho? Sendo assim, o gerenciamento de desempenho deve monitorar o máximo de recursos da rede de modo a fornecer uma indicação do nível de desempenho. Coletando informações e analisando-as o gerente de rede torna-se mais apto a reconhecer situações que indiquem degradação no desempenho da rede.

- **Gerenciamento de Segurança**:É função do gerenciamento de segurança controlar o acesso à rede de computadores assim como às informações obtidas através de nós desta rede. Também está relacionado com a geração, a distribuição e o armazenamento de chaves de criptografia. Senhas e outros modos de acesso e autorização devem ser mantidos e distribuídos.

Atualmente, o gerenciamento de redes está disponível pelo emprego de ferramentas e aplicativos proprietários. Utilizam-se ferramentas diversas, advindas de diferentes fornecedores, que não se comunicam entre si, aumentando os custos e a ineficiência do gerenciamento de redes. Os sistemas de gerenciamento de redes de um único fornecedor trabalham de forma integrada, mas com certos limites de abrangência, além de forçar o usuário a utilizar somente produtos de gerenciamento deste fornecedor, independentemente dos custos e da eficiência técnica.

O **Simple Network Management Protocol** é uma peça chave em ambientes abertos de gerenciamento de redes. O SNMP foi inicialmente projetado para uso em redes TCP-IP, mas tem sido aplicado em um conjunto bem mais abrangente de redes. O ambiente de gerenciamento SNMP constitui um padrão aberto para o gerenciamento de redes.

O SNMP tem sido implementado pelos vendedores, comprado por clientes e os gerentes de rede o estão usando efetivamente. Por sua demanda e sua capacidade, o SNMP tornou-se um padrão *de facto* de gerenciamento de redes.

3.2 O Ambiente de Gerenciamento SNMP

O ambiente de gerenciamento SNMP é baseado em quatro componentes [Ros91]:

- agentes de gerenciamento em elementos de rede;
- gerente de rede;
- protocolo de comunicação;

- informação de gerenciamento.

3.2.1 Agentes de Gerenciamento

Os agentes de gerenciamento são encontrados em elementos de rede, que são os dispositivos que formam a rede. Entre eles temos: estações de trabalho, servidores de arquivos, servidores de terminais, ou ainda, sistemas intermediários como roteadores, pontes, repetidores. Elementos de rede gerenciáveis têm um agente de gerenciamento que recebe e responde mensagens de gerenciamento de rede. Em geral, os agentes são passivos, somente respondendo quando perguntados. Entretanto, podem ser programados para enviar mensagens (*traps*) dado que algum evento ocorreu.

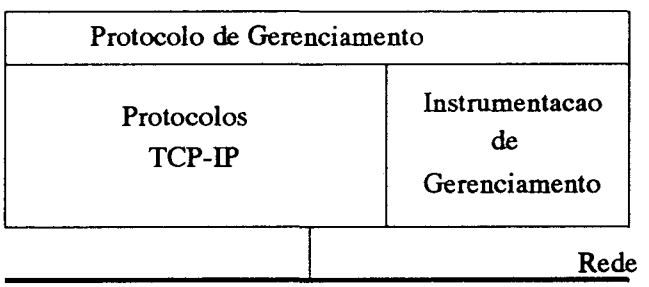

Figura 3.1: Modelo de Comunicação do Agente SNMP

O agente de gerenciamento comunica-se com estações de gerenciamento através do protocolo de gerenciamento que utiliza os demais protocolos TCP-IP (ver fig. 3.1 [Ros91]).

Alguns sistemas não implementam os protocolos necessários para uma comunicação direta com estações de gerenciamento, contudo, podem fazer parte de um ambiente de gerenciamento através de um agente *proxy*. Um agente *proxy* no SNMP é um agente que tem os recursos necessários e a autorização para responder a perguntas e comandos indiretamente por um sistema gerenciado. Para realizar tal tarefa, o agente *proxy* comunica-se com o sistema gerenciado através de um protocolo privado acordado por ambos.

3.2.2 Gerentes de Rede

No ambiente SNMP uma ou mais estações de gerenciamento executam aplicações que monitoram e controlam elementos de rede. Aplicações de gerenciamento executando nas estações de gerenciamento formulam e transmitem as requisições apropriadas e, então, esperam e processam as respostas recebidas dos agentes.

Em geral, uma estação de gerenciamento é responsável por vários elementos de rede. Um mesmo elemento de rede pode comunicar-se com mais de uma estação de gerenciamento.

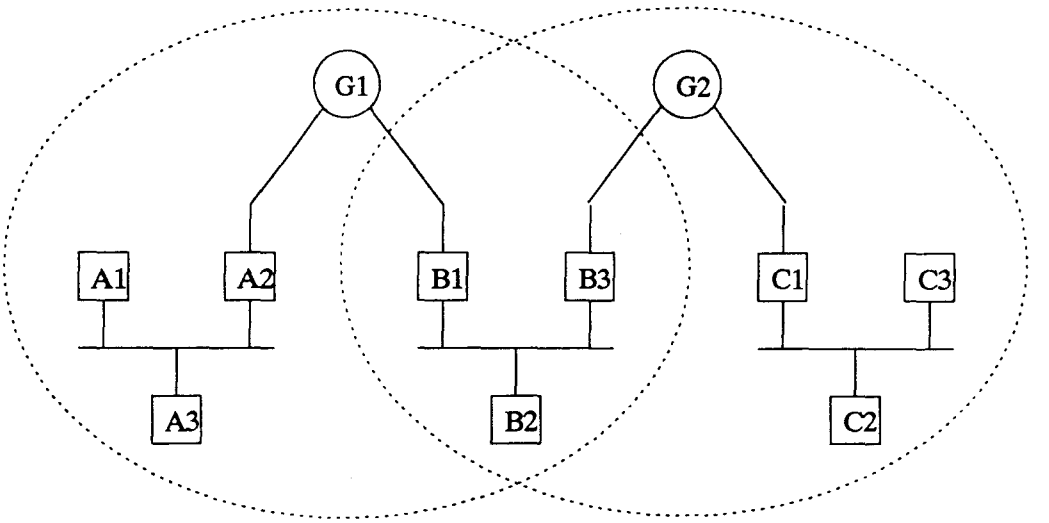

Figura 3.2: Estrutura de Gerentes e Agentes

No exemplo da figura 3.2 poderíamos ter:

- G1: gerenciando A1, A2, A3, B1, B2 e B3;
- G2: gerenciando B1, B2, B3, C1, C2 e C3.

3.2.3 Protocolo de Gerenciamento de Rede

O terceiro componente da arquitetura de gerenciamento é o protocolo de gerenciamento de rede. Ele é implementado em agentes e gerentes de rede.

O SNMP suporta dois tipos de transações [LR93, Ros91]:

- **polling**: mecanismo síncrono baseado no envio de requisições pelos gerentes e espera por respostas dos agentes;
- **trap**: mensagens assíncronas dos agentes aos gerentes.

Cada mensagem SNMP consiste de um cabeçalho de autenticação e um PDU (*Protocol Data Unit*) SNMP. O cabeçalho de autenticação consiste de um número de versão e um nome de comunidade, para que se realizem procedimentos de

autenticação e autorização. Existem cinco possíveis PDUs: **get-request**, **get-next-request**, **set-request**, **get-response** e **trap**.

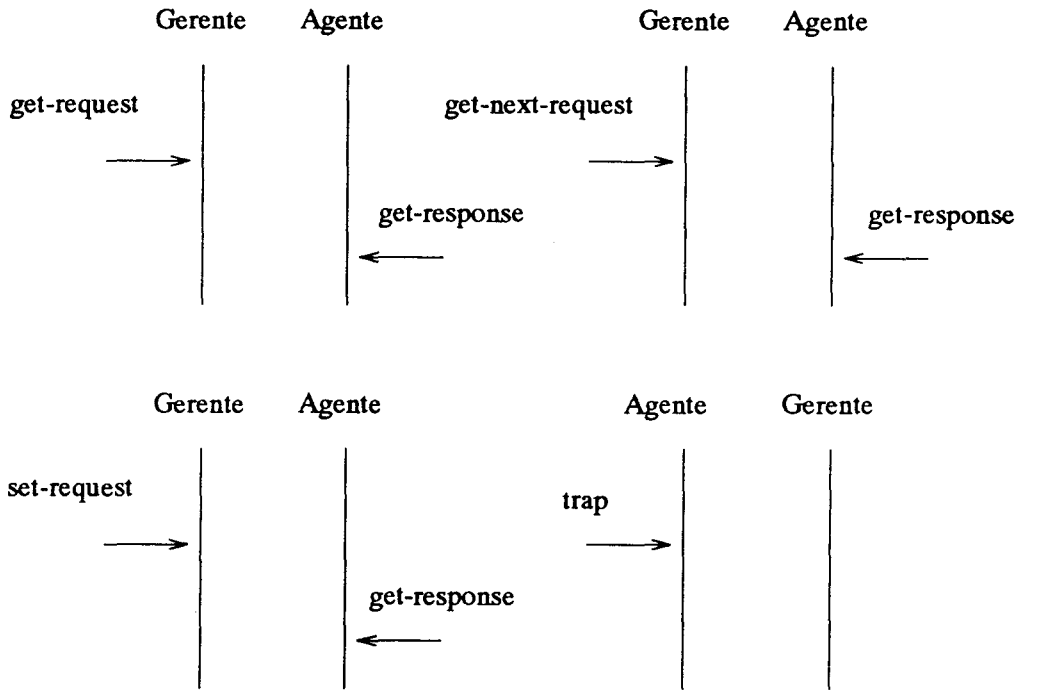

Figura 3.3: Troca de Mensagens do SNMP

A maior parte da comunicação no SNMP é através dos PDUs de requisição e resposta. Quando efetuam o *polling*, as estações de gerenciamento enviam as requisições apropriadas aos agentes nos elementos de rede. Quando um agente recebe uma requisição, ele a examina quanto à corretude, autenticação e autorização, e então, se a requisição for apropriada é processada.

As estações de gerenciamento realizam as requisições nomeando as variáveis a serem inspecionadas ou alteradas. Uma estação de gerenciamento pode requisitar várias informações em uma mesma mensagem.

Os operadores **get-request** e **get-next-request** são usados para a monitoração dos agentes. O operador **get** é usado para a leitura de valores de variáveis de gerenciamento. O operador **get-next** é capaz de realizar as funções do **get** acrescidas da habilidade de pesquisar o descendente lexicográfico, em relação ao identificador de variável, presente na mensagem. Isso o torna bastante robusto na caso de erros, já que mesmo que o identificador da variável não esteja inteira-

mente correto, obtém-se o valor da variável com o identificador lexicograficamente posterior. É muito útil também para se percorrer tabelas, onde os identificadores das variáveis nas colunas estão em seqüência lexicográfica.

O operador **set-request** é usado para o controle dos agentes. Estações de gerenciamento podem usá-lo para modificar informação de gerenciamento de rede e criar novas instâncias de informação (criar linhas em tabelas, por exemplo). No SNMP as ações são realizadas indiretamente através da alteração das variáveis de gerenciamento. Isso diminui em muito a complexidade do protocolo e aumenta a generalidade do ambiente.

O operador **get-response** é utilizado pelo agente de gerenciamento na resposta a qualquer requisição.

Como mencionamos, pode-se requisitar os valores de várias variáveis de gerenciamento de rede em uma mesma mensagem, porém, não existe suporte para recuperação ou alteração de objetos agregados, como uma tabela.

O SNMP define um quinto tipo de mensagem, a mensagem de **trap**. Um agente pode emitir um **trap** para indicar uma situação de exceção. Recebendo uma mensagem de **trap** a estação de gerenciamento toma as providências necessárias. Essa estratégia é conhecida como *trap-directed polling.*

O envio de **traps** não são garantidos em redes sujeitas a falhas, de modo que o ambiente de gerenciamento SNMP usa primariamente o *polling*, relegando o *trap* a uma posição secundária.

A figura 3.3 [Ros91] mostra a troca de mensagens entre agentes e gerentes.

3.2.4 Informação de Gerenciamento de Rede

O quarto componente da arquitetura SNMP é a informação de gerenciamento de rede que é compartilhada entre gerentes e agentes através do protocolo de gerenciamento. Isto é, os valores correntes das variáveis de gerenciamento de rede podem ser lidas ou escritas pelos gerentes e são sempre atualizadas pelos agentes.

A informação de gerenciamento pode tomar várias formas no SNMP. Algumas variáveis são inteiras, outras seqüências de octetos ou ainda identificadores de objetos.

A estrutura da informação é definida na SMI (Structure of Management Information) [MR90b]. A SMI define um modelo de objeto de informação e especifica regras para descrever os objetos existentes na MIB (Management Information Base) assim como regras para descrever novos objetos para a MIB [MR90a, MRe91].

A MIB contém a informação que é gerenciada (variáveis de gerenciamento de rede). Cada variável da MIB é parte de uma árvore que representa o modelo de informação definido pelo SMI.

Em suma, o SMI especifica as estruturas comuns e o esquema de identificação da informação de gerenciamento enquanto que a MIB especifica os objetos que são gerenciados, incluindo sua denominação, sintaxe, regras de acesso e estado.

3.3 Representação de Dados

As mensagens trocadas entre as camadas abaixo da aplicação têm uma estrutura relativamente simples. Por isso, somente um esquema de ordenação de bytes é suficiente para que os dados tenham sentido. No caso dos dados do protocolo de gerenciamento, que é implementado a nível de aplicação, há estruturas bem mais complexas. Assim, usa-se uma sintaxe abstrata para se representar tais dados. No ambiente de gerenciamento Internet utiliza-se um sub-conjunto da linguagem OSI Abstract Syntax Notation One (ASN.1), com os seguintes propósitos:

- definição do formato dos PDUs trocados pelo protocolo de gerenciamento;
- definição dos objetos que são gerenciados, isto é, a informação de gerenciamento.

Além de se poder descrever as estruturas de dados de maneira independente da máquina através de uma sintaxe abstrata, deve-se poder transmitir esta informação na rede de maneira não ambígua, através de uma sintaxe de transferência.

A idéia da utilização de um subconjunto da ASN.1 surgiu para aproveitar os princípios da sintaxe abstrata mas evitar complexidades desnecessárias, seguindo sempre uma tendência de minimizar o esforço computacional para diminuir os requisitos em nós gerenciados.

3.3.1 Tipos ASN.1 Utilizados no SNMP

O ambiente de gerenciamento utiliza quatro tipos ASN.1:

- **Simple Types**
  - **Integer**: tipo de dado que representa um número cardinal; não existe limitação quanto ao tamanho.
  - **Octet String**: tipo de dado composto por zero ou mais octetos.

- **Object Identifier**: tipo de dado representando uma denominação de objeto.
- **Null**: tipo de dado que age como um delimitador de espaço sem representação efetiva.

- **Constructed Types**
  - **Sequence**: denota uma lista ordenada de elementos ASN.1.
  - **Sequence of**: denota uma lista ordenada de elementos de um mesmo tipo ASN.1.

- **Tagged Types**: fornece uma forma de definir novos nomes para tipos pré-existentes.

- **Sub-Types**: fornece uma forma de refinar a semântica de um tipo, limitando-a a um subconjunto dos valores possíveis.

3.3.2 Denominação de Variáveis

No ambiente de gerenciamento SNMP, os objetos de gerenciamento de rede ou variáveis de gerenciamento de rede são denominados através de identificadores de objetos (*object identifiers*). Os objetos de gerenciamento são organizados conceitualmente em uma estrutura de árvore.

A árvore consiste de uma raiz conectada a um número de sub-nós. Cada sub-nó, por sua vez, tem um número arbitrário de filhos. Esse processo pode continuar de maneira indefinida. Cada nó, (exceto a raiz) tem um rótulo, um número inteiro e uma descrição a ele associados. O primeiro nível da árvore é apresentado na figura 3.4.

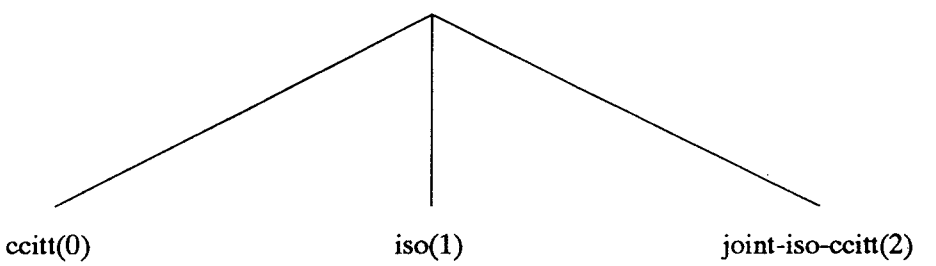

Figura 3.4: Primeiro Nível da Árvore de Objetos

O identificador de objeto associado a um objeto de gerenciamento em particular corresponde ao caminho na árvore da raiz até o mesmo. O identificador de

objeto é a seqüência de inteiros associados aos nós que constituem tal caminho, da raiz ao objeto. Assim, cada objeto é unicamente denominado.

Um identificador de objeto é um nome designado administrativamente em que a autoridade de definir variáveis é delegada a vários grupos para diferentes sub-árvores.

Identificadores de objetos podem ser usados também para denominar objetos que não são variáveis de gerenciamento de rede, tais como, documentos de padrões, elementos de rede, entre outros.

As variáveis de gerenciamento de rede são identificadas por nomes de variáveis. Identificadores de objeto podem ser representados de quatro formas, por exemplo:

- **nome**: sysLocation
- **número**: 1.3.6.1.2.1.1.6
- **nome e número**: iso(1) org(3) dod(6) internet(1) mgmt(2) mib(1) system(1) 6
- **número com prefixo**: system 6

A árvore representando o objeto **sysLocation** é mostrada na figura 3.5.

3.4 Base de Informação de Gerenciamento - MIB

A MIB padrão Internet [MRe91, MR90a] define as variáveis que devem ser implementadas por todos os sistemas contendo o conjunto de protocolos TCP-IP. Existem formas de se incluir extensões à MIB específicas de um produto que permite às empresas padronizar seus produtos de acordo com o SNMP.

Como mencionamos, todas as variáveis SNMP são parte de uma árvore de denominação de objetos global. A SMI define, através de uma macro ASN.1, a forma de se definir a semântica de um objeto [Ros91].

```
OBJECT-TYPE MACRO ::=
BEGIN
  TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                    \"ACCESS" Access
                    "STATUS" Status
  VALUE NOTATION ::= value (VALUE ObjectName)
```

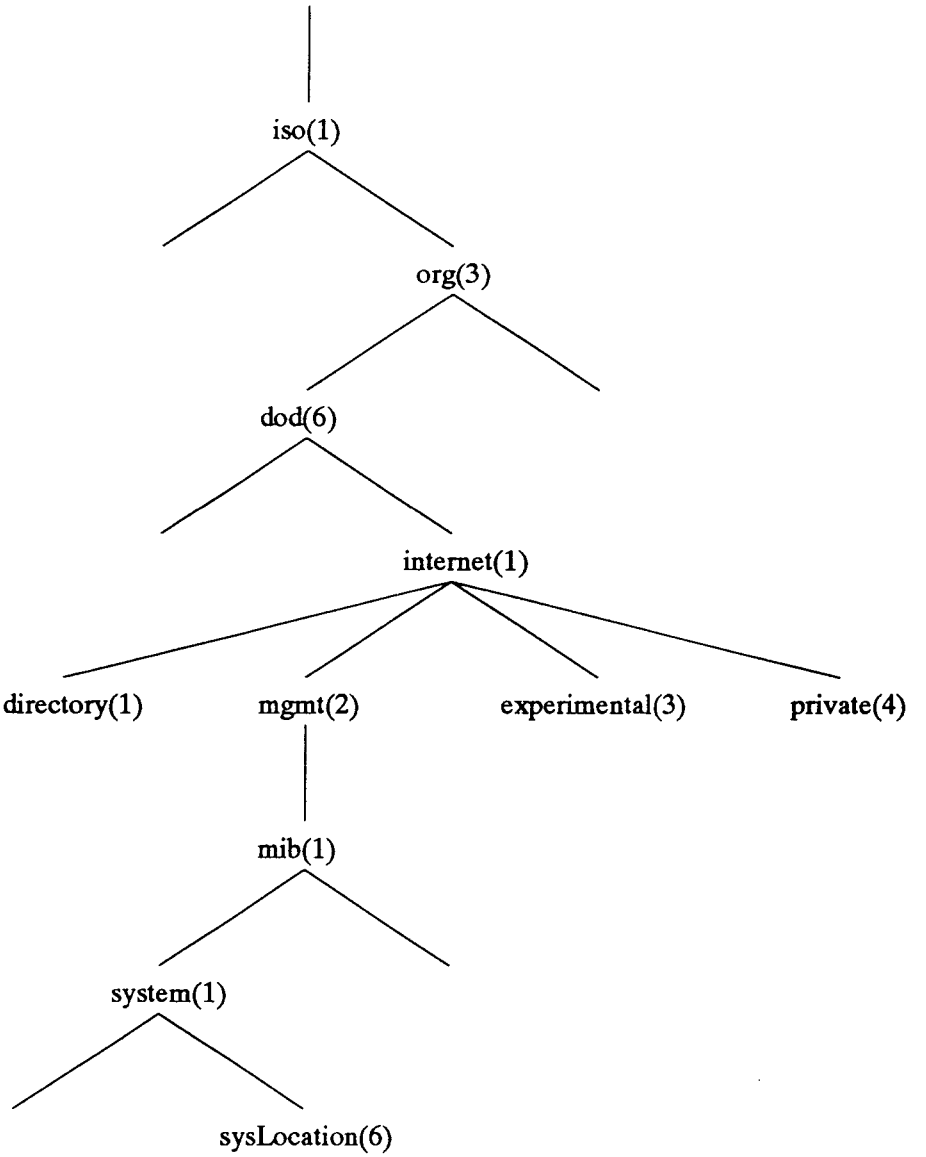

Figura 3.5: Árvore Representando a Variável **sysLocation**

```
  Access ::= "read-only"
             | "read-write"
             | "write-only"
             | "not-accessible"
  Status ::= "mandatory"
             | \"optional"
             | \"obsolete"
END
```

Onde temos os seguintes significados para os campos:

- **Syntax**: define o tipo de dado que modela o objeto.
- **Access**: indica o nível de acesso ao objeto gerenciado; podemos ter os seguintes níveis:
  - somente leitura;
  - leitura-escrita;
  - somente escrita;
  - não acessível.
- **Status**: indica os requisitos de implementação para o objeto:
  - obrigatório;
  - opcional;
  - obsoleto;
- **ObjectName**: indica o tipo de dado associado ao valor do objeto.

Com esse formalismo, um objeto gerenciado é definido sem possibilidade de ambigüidades.

Como exemplo, podemos considerar a definição do objeto sysLocation:

```
sysLocation OBJECT-TYPE
    SYNTAX DisplayString (SIZE 0..255)
    ACCESS read-only
    STATUS mandatory
    ::={system 6}
```

O organismo responsável por delegar a autoridade na sub-árvore relativa aos objetos de gerenciamento de rede é o IANA (Internet Assigned Number Authority), que tem quatro sub-árvores sob sua autoridade.

A primeira é reservada para uso futuro.

A segunda contém a MIB padrão além de documentos associados à MIB.

A terceira é usada para identificar objetos em desenvolvimento.

A quarta sub-árvore é para a definição de variáveis de gerenciamento privadas. Empresas podem requisitar do IANA uma parcela da árvore para representar bases de gerenciamento associadas aos seus produtos.

A primeira MIB (MIB-I) [MR90a] foi projetada para incluir um número mínimo de objetos para ser útil no gerenciamento Internet.

Todos os dispositivos a serem gerenciados devem implementar a MIB, contudo, nem todas as funções estão presentes em todos os nós. Por isso a MIB foi dividida em grupos e cada nó implementa os grupos relacionados às funções que executa. No final de 1989 foi criada a MIB-II [MRe91], que criava novos objetos, mantendo a compatibilidade com os da MIB-I.

Vejamos, então, os grupos presentes na MIB-II:

- **System Group**: deve ser implementado por todos os nós gerenciados e contém informações genéricas de configuração. Exemplos de objetos:
  - sysDescr: descrição do dispositivo;
  - sysLocation: localização física do dispositivo.
- **Interfaces Group**: deve ser implementado por todos os nós gerenciados e contém informações sobre as entidades na camada de interface. Exemplos:
  - ifSpeed: taxa de transmissão em bits/segundo;
  - ifOutOctets: total de octetos enviados ao meio físico.
- **Address Translation Group**: deve ser implementado por todos os nós gerenciados e contém informação sobre resolução de informação de endereço. Exemplos:
  - atPhisAddress: endereço físico;
  - atNetAddress: endereço IP.

  Na MIB-II as funções do **Address Translation Group** foram englobadas pelo **IP Group** e devem se tornar obsoletas.

- **IP Group**: deve ser implementado por todos os nós gerenciados. Exemplos:
  - ipForwarding: agindo como *host* ou roteador;
  - ipInDelivers: datagramas passados à camada superior.
- **ICMP Group**: deve ser implementado por todos os nós gerenciados. Exemplos:
  - icmpOutRedirect;
  - icmpInErrors.
- **TCP Group**: deve ser implementado pelos nós gerenciados que implementem TCP. Exemplos:
  - tcpActiveOpens: número de conexões ativas abertas;
  - tcpOutSegs: número de segmentos enviados.
- **UDP Group**: deve ser implementado pelos nós que implementam UDP. Exemplos:
  - udpNoPorts: datagramas destinados a portas desconhecidas;
  - udpOutDatagrams: datagramas enviados da aplicação.
- **EGP Group**: é obrigatório para os nós gerenciados que implementam o EGP (External Gateway Protocol). Está relacionado com informações de alcance entre sistemas autônomos. Exemplos:
  - egpNeighAddr: endereço IP de um vizinho;
  - egpNeighInMsgs: mensagens vindas de um vizinho.
- **Transmission Group**: relativo a dispositivos específicos a meios físicos.
- **SNMP Group**: deve ser implementado em todos os nós gerenciados. Exemplos:
  - snmpInTooBigs: PDUs recebidos com error-status de tooBig;
  - snmpInTotalReqVars: número de objetos pesquisados.

3.5 O Protocolo

Os quatro PDUs de requisição e resposta (ver figura 3.6) têm o mesmo formato. Somente o PDU de trap tem forma diferente. Os PDUs de requisição e resposta foram projetados com a mesma forma por questões de simplicidade [CFSD90, LR93, Ros91].

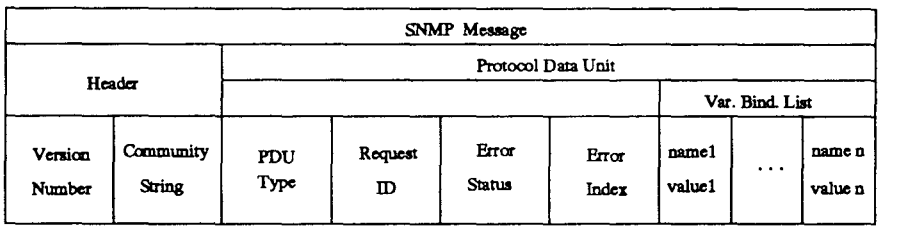

Figura 3.6: Mensagem SNMP de Requisição e Resposta

Em conseqüência desta igualdade de formatos alguns campos não terão valores úteis para algumas mensagens, como por exemplo, os campos de valor em um **get-request**. Vejamos o que significa cada campo da mensagem:

- **Version Number**: inteiro indicando a versão do SNMP utilizada; caso a versão da mensagem e a versão utilizada pelo elemento de rede gerenciado não coincidam a mensagem é descartada.
- **Community String**: campo contendo o nome da comunidade do remetente da mensagem; é utilizado com fins de autenticação.
- **PDU-Type**: indica qual o tipo do PDU recebido (get, get-next, set - request ou get-response).
- **Request-id**: valor inteiro usado pelo gerente para diferenciar entre requisições distintas.
- **Error-status**: se for diferente de zero, indica que uma exceção ocorreu.
- **Error-index**: se for diferente de zero, indica em que variável está o (primeiro) erro (a primeira variável da lista é dita ter offset igual a um).
- **Variable Bindings**: uma lista contendo nome e valor de variáveis.

Outro formato utilizado é do PDU de *Trap* [Ros91, LR93]:

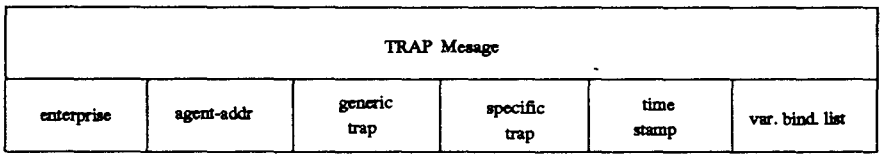

Figura 3.7: Mensagem SNMP de Trap

- **Enterprise**: valor do objeto **sysObjectID** do agente.
- **Agent-addr**: valor do objeto **NetworkAddress** do agente.
- **Generic Trap**: indica um entre alguns eventos extraordinários mais usuais.
- **Specific Trap**: identifica o *trap* específico que ocorreu, caso contrário seu valor é zero.
- **Time-stamp**: valor do objeto **sysUpTime** do agente quando o evento que causou o *trap* ocorreu.
- **Variable Bindings**: mesmo significado que nas mensagens de requisição.

3.6 SNMP Versão 2 (SNMPv2)

A versão 2 do SNMP, padronizada em abril de 1993, representa uma mudança grande em relação à primeira versão, cobrindo algumas lacunas lá existentes. O modelo da versão 2 utiliza identidades distintas para nós trocando mensagens SNMPv2, deixando de lado o conceito de comunidade presente na primeira versão. Isto foi feito para permitir um modelo de controle mais conveniente e para permitir o futuro uso de protocolos de segurança [CMRW93b, CM94c].

3.6.1 Modelo

- **Parceiro SNMPv2** O Parceiro SNMPv2 é um conceito significando um ambiente de execução de gerenciamento cuja operação é restrita a um subconjunto de todas as possíveis operações de uma entidade SNMPv2.

  Cada parceiro SNMPv2 compreende:

  - uma identificação única na rede;
  - um endereço de rede lógico no qual o parceiro executa, caracterizado por um protocolo de transporte e informação de endereçamento;

- um protocolo de autenticação próprio e parâmetros associados;
- um protocolo de privacidade próprio e parâmetros associados.

- **Entidade SNMPv2**

  A entidade SNMPv2 é o processo que, de fato, realiza as operações de gerenciamento. Quando uma entidade SNMPv2 está atuando como um parceiro SNMPv2 específico, torna-se restrita ao sub-conjunto de operações definidas para tal parceiro.

  Uma mesma entidade pode suportar mais de um parceiro, não sendo o processamento necessariamente concorrente.

  Cada entidade mantém uma base de dados local representando todos os parceiros que suporta, sejam locais ou remotos. Existe também uma base de dados que representa todos os objetos gerenciados e uma política de controle de acesso.

- **Estação de Gerenciamento SNMPv2**

  A estação de gerenciamento é o papel assumido por um parceiro SNMPv2 quando este inicia operações de gerenciamento ou recebe notificações (*traps*).

- **Agente de Gerenciamento SNMPv2**

  É o papel assumido por um parceiro SNMPv2 quando este realiza operações de gerenciamento em resposta a mensagens SNMPv2 de gerentes.

- **Comunicação de Gerenciamento SNMPv2**

  É a comunicação de um determinado parceiro a outro sobre informações de gerenciamento contidas em um contexto acessível pela entidade SNMPv2 apropriada. A comunicação de gerenciamento pode ser dos seguintes tipos:

  - requisição de informação (getRequest, getNextRequest ou getBulkRequest);
  - indicação de informação (InformRequest ou SNMPv2_Trap);
  - indicação ordenativa de informação (setRequest);
  - confirmação de informação (Response).

- **Comunicação Autenticada**

  É aquela em que o parceiro SNMPv2 que originou de uma mensagem pode ser identificado de forma confiável, e, além disso, a integridade do conteúdo

da mensagem é garantida. Uma mensagem autenticada contém informações do protocolo de autenticação e dados úteis.

- **Comunicação com Privacidade**

  É a informação autenticada que é protegida contra leituras não autorizadas. A comunicação de gerenciamento com privacidade é composta por uma informação que identifica o destinatário da mensagem (somente este será autorizado a ler o conteúdo) e a mensagem original criptografada.

- **Política de Controle de Acesso**

  É a especificação de uma política de acesso local em função do contexto visto pela entidade e das classes de comunicação autorizadas (mensagens autorizadas) entre pares de parceiros SNMPv2. A especificação compreende quatro partes:

  - os alvos do controle de acesso: parceiros que podem realizar operações em resposta a requisições (agentes);
  - os sujeitos do controle de acesso: parceiros que podem requisitar informações de gerenciamento (gerentes);
  - recursos gerenciados: são os contextos identificando informações de gerenciamento que podem ser acessadas;
  - política que especifica as classes permitidas a cada contexto particular.

3.6.2 Principais diferenças da Versão 2 e Versão 1 do SNMP

- **Segurança:** a versão 2 fornece dispositivos que possibilitam a inclusão de protocolos de autenticação e privacidade às suas mensagens [CMRW93a, CMRW93c].

- PDUs

  - getBulkRequest: Tem como objetivo a transferência de dados de uma quantidade tão grande quanto o possível de uma só vez, aplicando-se principalmente a transferências de tabelas grandes. O tamanho do PDU é tal que seja o menor entre os máximos possíveis das duas partes envolvidas [CMRW93e].
  - InformRequest: É utilizado por uma aplicação atuando como gerente para outra atuando também como gerente para fornecer informação de um parceiro SNMPv2 local à aplicação remetente [CMRW93e].

- **Comunicação Gerente-Gerente:** Com a finalidade de facilitar o gerenciamento de grandes redes, ou redes compostas de domínios administrativos autônomos, a versão 2 do SNMP provê mecanismos para a comunicação entre estações de gerenciamento [CMRW93d].

Capítulo 4

Esquema de Gerenciamento de Aplicações

4.1 Introdução

Tem havido uma diferenciação bastante clara entre os papéis dos monitores de rede e os sistemas de gerenciamento de rede [Rei91]. A monitoração tem consistido primariamente de três atividades: coleta de dados de baixo nível em segmentos de rede, informação estatística detalhada do estado da rede a nível físico e obtenção de dados (sem tratamento) de nós de rede. Na maior parte dos casos, os monitores têm sido agentes passivos.

Estações de gerenciamento, por outro lado, têm operado em um nível mais alto que monitores pois não estão diretamente relacionadas à rede física. Os nós gerenciadores geralmente têm informações como a topologia da rede e o comportamento dos nós da rede, especialmente de dispositivos como pontes e roteadores. Através do protocolo de gerenciamento as estações podem realizar controle ativo, recuperando informação e modificando o estado de nós.

Com o SNMP sendo considerado cada vez mais como protocolo padrão de gerenciamento de redes é possível considerar um mecanismo que possibilite a comunicação entre monitores de rede e estações de gerenciamento, unindo a capacidade de recolher dados dos monitores com a visão centralizada das estações de gerenciamento de rede.

Como vimos no capítulo referente a TCP-IP, de acordo com a camada da arquitetura que se observa, temos acesso a informações de diferentes tipos:

- **Camada de Interface**: O cabeçalho de cada pacote tem os endereços de origem e destino de nível físico, além de outras informações também dependentes do meio físico.

- **Camada de Rede**: No cabeçalho IP temos os endereços de origem e destino, cada um deles referindo-se a uma máquina na rede a nível global, entre outras informações.

- **Camada de Transporte**: Seja o protocolo de transporte UDP ou TCP, temos as portas de origem e destino que identificam um par de processos se comunicando, representando o uso de uma aplicação na rede, além de outras informações.

Os analisadores de rede em geral contabilizam pacotes a nível de interface [SC87]. Com esses dados pode-se obter quantos *frames* cada elemento de rede (computador, impressora conectada à rede, entre outros) transmite e/ou recebe em intervalos de tempo, detetar mensagens com erro, mensagens com problemas de atraso, entre outras tarefas, sempre relacionadas a uma rede física.

Queremos investigar também as informações disponíveis nas camadas acima da interface. Acessando a camada IP, podemos ter controle sobre o tráfego entre grupos de computadores quaisquer na Internet. Se fizermos o mesmo com relação à camada de transporte, teremos também a informação relativa à aplicação que gerou tal tráfego.

Sendo assim, um monitor que observe as camadas de rede e de transporte pode contabilizar o tráfego entre quaisquer grupos de computadores na Internet, estejam eles na mesma rede física ou não, e discriminar o tráfego de acordo com a aplicação utilizada.

Suponhamos uma topologia como a descrita na figura 4.1. Como exemplo, poderíamos obter os seguintes dados:

1. Tráfego entre o servidor A1 e os outros computadores (incluindo as redes , 2 e 3), classificando o tráfego devido a NFS e outras aplicações.

2. Tráfego das máquinas A1, B1 e C1 à máquina E1, isto é, quanto tráfego ocorre na rede 1 devido à utilização da impressora laser pelas máquinas desta rede.

3. Tráfego entre as redes 1 e 2 classificado de acordo com a utilização devido a correio eletrônico, transferência de arquivos, login remoto e outras aplicações.

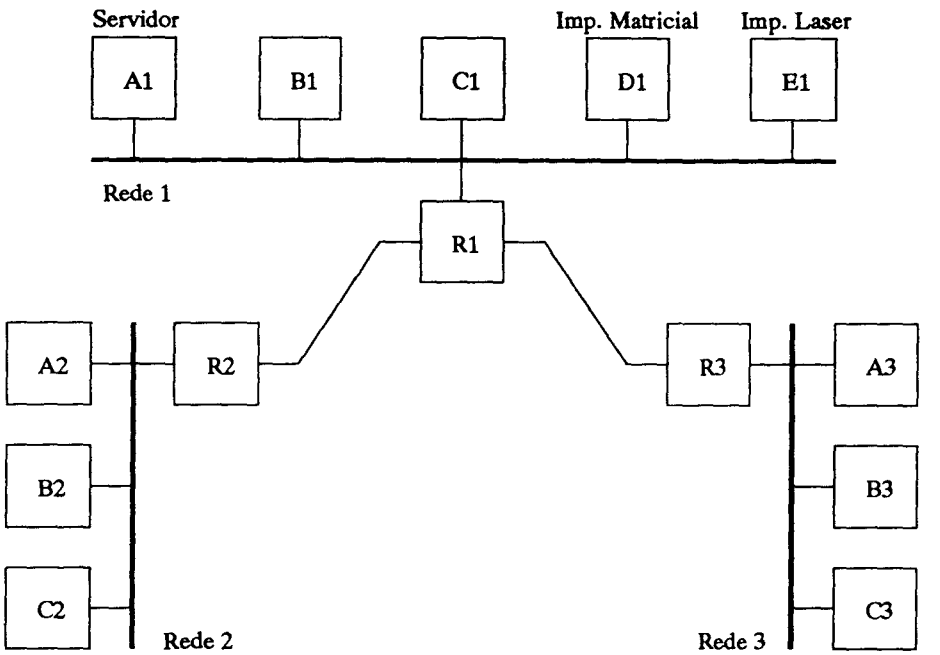

Figura 4.1: Exemplo de Topologia de Redes

4. Tráfego entre os computadores B2 e B3 classificado por login remoto, correio eletrônico e outras aplicações.

Nossa intenção não é medir o tráfego instantâneo, mas o tráfego acumulado em intervalos, cuja duração é configurável. O agente de gerenciamento mantém informações sobre o tráfego acumulado no intervalo imediatamente anterior ao atual. Essas informações permanecem invariáveis até o final do intervalo, quando são atualizadas com o tráfego relativo ao intervalo recém terminado. Cabe ao gerente fazer requisições periódicas, com período igual à duração do intervalo configurado no agente, para ler as informações do agente e guardá-las para posterior consulta. Fizemos a opção do gerente guardar os dados dos vários intervalos, pois simplifica a implementação do agente e fornece ao usuário, que usa o aplicativo gerente, um maior controle sobre o processo de monitoração. Pode-se observar que, pelo fato dos dados no agente permanecerem invariáveis durante todo o intervalo, as requisições do gerente podem ser iniciadas em qualquer instante do primeiro intervalo de medição, não exigindo uma sincronização sofisticada [SM].

Como exemplo, suponhamos que tivéssemos monitorado o tráfego entre as 14:00h e as 15:00h, em intervalos de 15 minutos, entre as redes 1 e 2 da figura

4.1, classificado por correio eletrônico, transferência de arquivos, login remoto e outras aplicações. Ao final da monitoração, o aplicativo gerente poderia ter coletado os seguintes dados (consideramos como referência de origem o primeiro grupo descrito do domínio):

- **14:00 às 14:15:**

Aplicação	Bytes Enviados	Bytes Recebidos
SMTP	15.400	12500
FTP	230.000	122.300
TELNET	12.300	1.400
Outros	1.023.320	1.523.000

- **14:15 às 14:30:**

Aplicação	Bytes Enviados	Bytes Recebidos
SMTP	13.300	16.200
FTP	340.000	220.000
TELNET	560	12.200
Outros	1.230.000	870.000

- **14:30 às 14:45:**

Aplicação	Bytes Enviados	Bytes Recebidos
SMTP	9.300	13.200
FTP	115.200	130.200
TELNET	5.500	3.000
Outros	2.030.000	1.890.320

- **14:45 às 15:00:**

Aplicação	Bytes Enviados	Bytes Recebidos
SMTP	34.000	12.230
FTP	230.450	14.000
TELNET	2.450	14.234
Outros	1.400.230	1.564.340

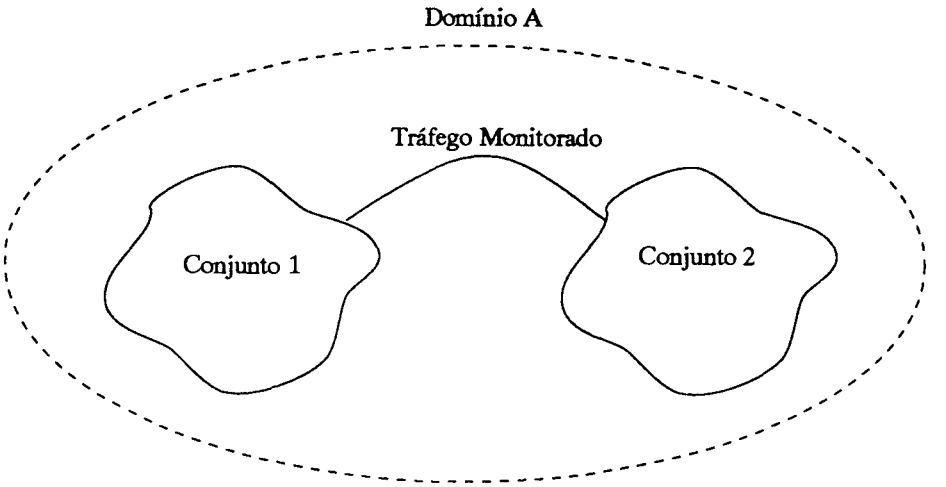

Figura 4.2: Estrutura de um Domínio

4.2 Domínios Gerenciáveis

Definimos domínio como sendo um par de conjuntos de elementos de rede entre os quais se quer monitorar o tráfego.

Na figura 4.2 vemos dois conjuntos de elementos de rede formando um domínio. Os conjuntos que compõem domínios gerenciáveis seguem a seguinte regra de formação:

Seja um conjunto definido por dois endereços IP, **firstAddr** e **lastAddr**. Pertencem ao conjunto todos os elementos de rede com endereços IP, **ipAddr**, tal que:

$$\mathbf{firstAddr} =< \mathbf{ipAddr} =< \mathbf{lastAddr}$$

Definindo-se conjuntos desta forma, pode-se monitorar o tráfego tanto entre dois elementos de rede como entre duas redes ou sub-redes. Convém relembrar que um endereço no ambiente Internet é formado por um par (netid, hostid) [Com91], portanto, podemos sempre englobar todos os elementos de uma rede em conjuntos que formam domínios gerenciáveis.

Podemos ver na figura 4.3 três redes interligadas, estando os elementos de rede com os respectivos endereços IP. Vejamos alguns possíveis domínios gerenciáveis para este caso:

1. Domínio formado por dois elementos de rede de uma mesma rede física, por exemplo, domínio = $(A1);(B1)$. Neste caso, temos:

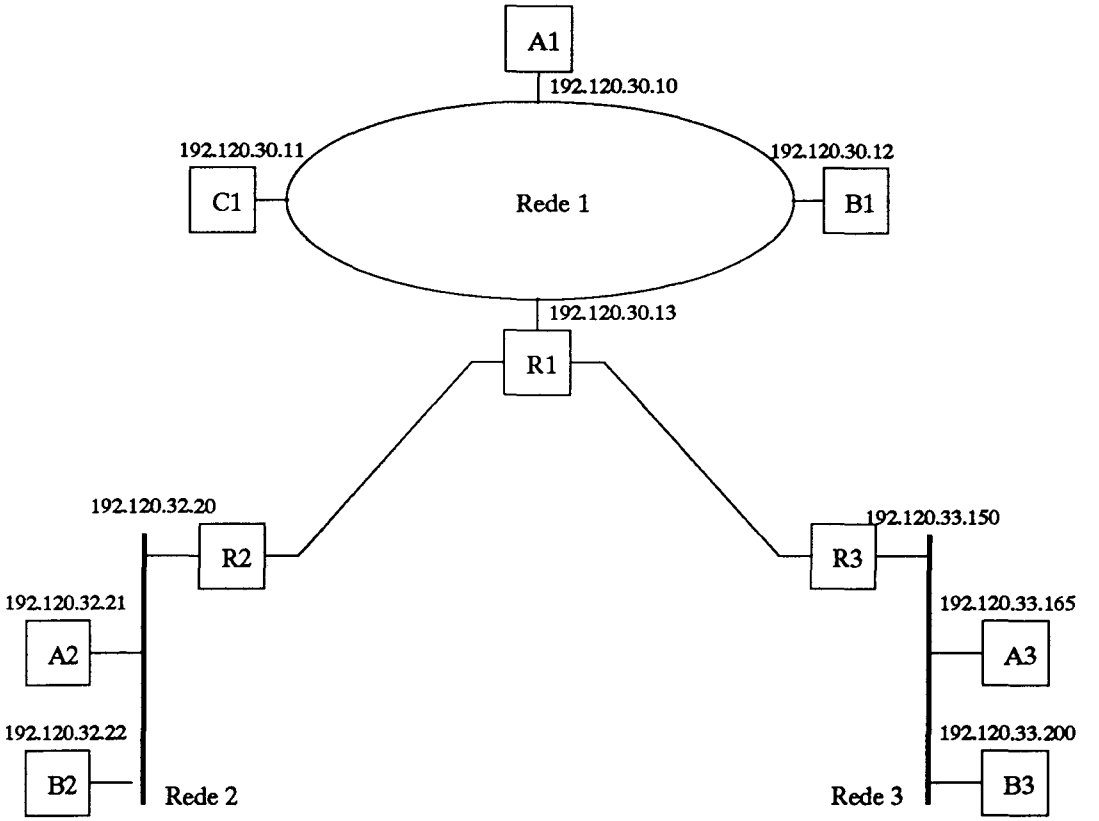

Figura 4.3: Redes com Endereços IP

Conjunto 1: firstAddr = 192.120.30.10
lastAddr = 192.120.30.10
Conjunto 2: firstAddr = 192.120.30.12
lastAddr = 192.120.30.12

2. Domínio formado por grupos de elementos de rede de uma mesma rede física, por exemplo, domínio = $(A1, C1); (B1, R1)$. Neste caso, temos:

Conjunto 1: firstAddr = 192.120.30.10
lastAddr = 192.120.30.11
Conjunto 2: firstAddr = 192.120.30.12
lastAddr = 192.120.30.13

3. Domínio formado por grupos de elementos de rede em redes físicas distintas, por exemplo, domínio = $(A1, C1); (A3, B3)$. Neste caso, temos:

   Conjunto 1: firstAddr = 192.120.30.10
   lastAddr = 192.120.30.11
   Conjunto 2: firstAddr = 192.120.33.165
   lastAddr = 192.120.33.200

4. Domínio formado por redes, por exemplo, domínio = $(rede1); (rede2)$. Neste caso, como ambas as redes são classe C, poderíamos definir os conjuntos da seguinte forma:

   Conjunto 1: firstAddr = 192.120.30.00
   lastAddr = 192.120.30.255
   Conjunto 2: firstAddr = 192.120.32.00
   lastAddr = 192.120.32.255

4.3 Base de Informações de Gerenciamento (MIB)

Vamos definir a estrutura de dados que dá suporte ao modelo de gerenciamento de aplicações [MR90a]. Como vimos, a base de dados de gerenciamento é o que, de fato, define as informações que serão gerenciadas e, conseqüentemente, o escopo das operações que poderão ser realizadas. Somente através de leituras e escritas o controle e a monitoração do elemento de rede são realizados, sendo a MIB (Management Information Base) assim, de importância primordial no modelo.

Sendo o nosso modelo privado, poderíamos requisitar ao IANA uma sub-árvore para nosso uso. Como vimos, existe um nó chamado **private**. Um dos nós filhos deste nó é o **enterprise**. O IANA fornece uma sub-árvore cuja raiz é um nó filho do **enterprise** para as organizações que desejam definir suas próprias MIBs.

Como nosso modelo visa monitorar o tráfego classificando-o por aplicações, poderíamos rotular o nó raiz da nossa sub-árvore de **apTraffic**.

Precisamos de uma estrutura de dados que armazene os domínios que estão sendo gerenciados e, para cada domínio, as aplicações cujo tráfego se deseja medir.

Vamos usar duas variáveis para informações do estado geral do agente:

- **apMode**: indica se o agente está em monitoração (1) ou não (2).
- **apPeriod**: indica a duração, em segundos, do intervalo entre atualizações da MIB.

Usaremos duas tabelas para armazenar informações sobre domínio e tráfego. A primeira, **apDomTable**, armazena os domínios a serem monitorados, tendo a seguinte estrutura:

- **apDomIndice**: inteiro identificando um domínio em particular.
- **apFirstAddr1**: menor endereço IP do primeiro conjunto do domínio.
- **apLastAddr1**: maior endereço IP do primeiro conjunto do domínio.
- **apFirstAddr2**: menor endereço IP do segundo conjunto do domínio.
- **apLastAddr2**: maior endereço IP do segundo conjunto do domínio.
- **apDomStatus**: estado corrente da linha da tabela: válida (1) e inválida (2).

A segunda tabela armazena os dados relativos ao tráfego acumulado, discriminado por aplicações, e deve ser atualizada na freqüência determinada pela variável **apPeriod**. Chamamos esta tabela de **apDataTable**. As suas linhas têm as seguintes colunas:

- **apDataIndice**: identifica o domínio a monitorar, relacionando-se com a tabela **apDomTable** através do **apDomIndice**.
- **apDataProt**: código do protocolo de transporte usado pela aplicação a ser monitorada.
- **apDataAplic**: código da porta usada pela aplicação a ser monitorada.
- **apDataBytesEnv**: número de bytes enviados do primeiro conjunto ao segundo conjunto do domínio para uma dada aplicação.
- **apDataDatEnv**: número de datagramas enviados do primeiro conjunto ao segundo conjunto do domínio para uma dada aplicação.
- **apDataBytesRec**: número de bytes recebidos pelo primeiro conjunto do segundo conjunto do domínio para uma dada aplicação.

- **apDataDatRec**: número de datagramas recebidos pelo primeiro conjunto do segundo conjunto do domínio para uma dada aplicação.
- **apDataStatus**: estado corrente da linha da tabela: válida (1) e inválida (2).

A figura 4.4 mostra a estrutura da base de dados proposta para o gerenciamento do tráfego de aplicações. A definição em ASN.1 da MIB **apTraffic** é apresentada no apêndice A.

Como exemplo, vamos mostrar como seria a base de dados de um agente que estivesse gerenciando o tráfego da rede da figura 4.3 para os quatro domínios lá descritos, sendo que em cada domínio estaríamos interessados em medir o tráfego devido às seguintes aplicações:

- **Domínio 1:** FTP (protocolo de transporte TCP - 6 e porta 21) e TELNET (protocolo de transporte TCP - 6 e porta 23).
- **Domínio 2:** TELNET (protocolo de transporte TCP - 6 e porta 23) e TFTP (protocolo de transporte UDP - 17 e porta 69).
- **Domínio 3:** SMTP (protocolo de transporte TCP - 6 e porta 25).
- **Domínio 4:** SUNRPC (protocolo de transporte TCP - 6 e porta 111) e SUNRPC (protocolo de transporte UDP - 17 e porta 111).

Supondo que a medição não tivesse sido iniciada e que o agente estivesse programado para acumular dados a cada 10 minutos, isto é, 600 segundos, teríamos os seguintes dados na MIB:

apMode = 2;
apPeriod = 600;

DomIndice	FirstAddr1	LastAddr1	FirstAddr2	LastAddr2	DomStatus
1	192.120.30.10	192.120.30.10	192.120.30.12	192.120.30.12	1
2	192.120.30.10	192.120.30.11	192.120.30.12	192.120.30.13	1
3	192.120.30.10	192.120.30.11	192.120.33.165	192.120.33.200	1
4	192.120.30.0	192.120.30.255	192.120.32.0	192.120.32.255	1

apDomTable

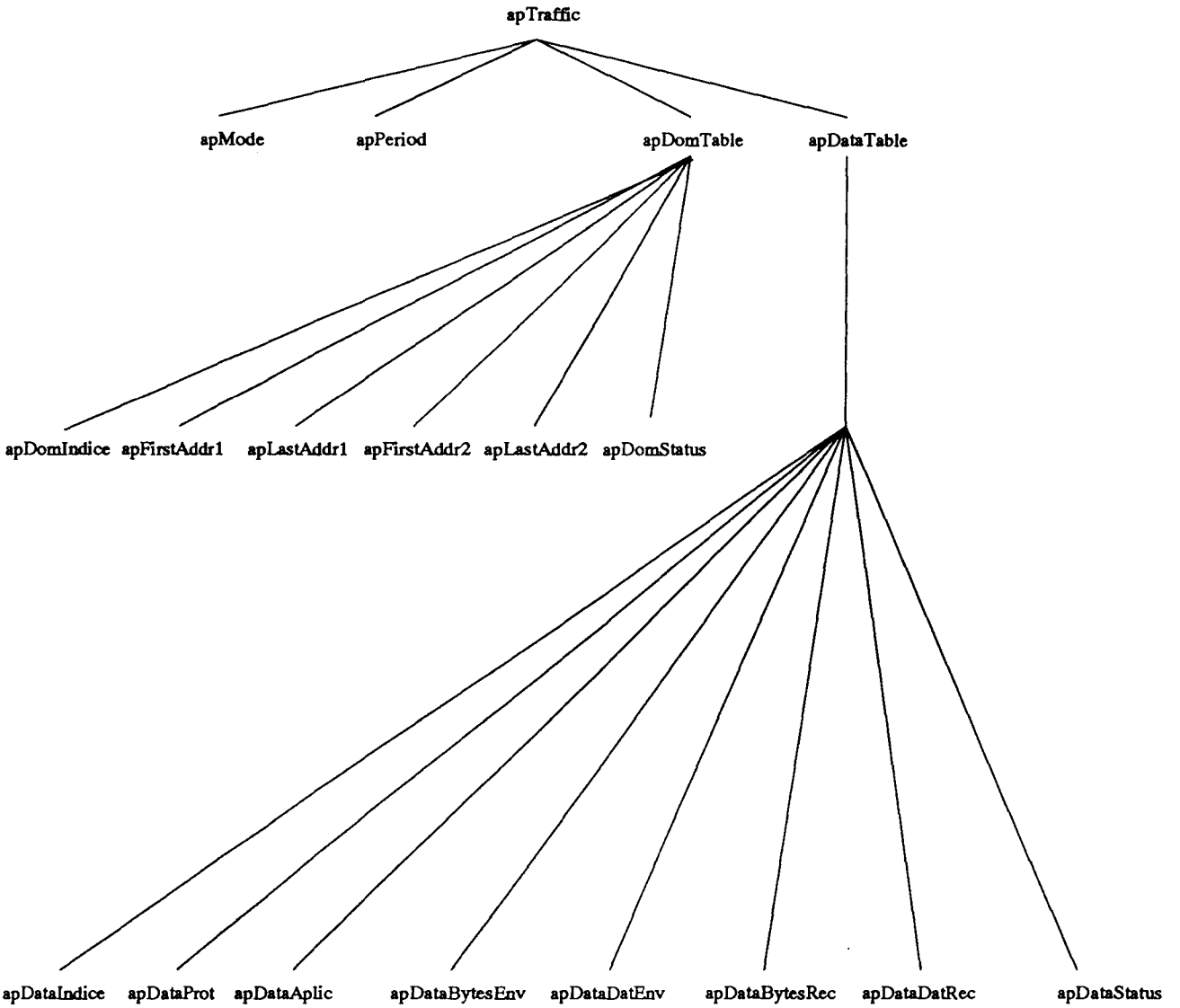

Figura 4.4: Sub-árvore da MIB apTraffic

Indice	Prot	Aplic	BytesEnv	DatEnv	BytesRec	DatRec	Status
1	6	21	0	0	0	0	1
1	6	23	0	0	0	0	1
1	127	127	0	0	0	0	1
2	6	23	0	0	0	0	1
2	17	69	0	0	0	0	1
2	127	127	0	0	0	0	1
3	6	25	0	0	0	0	1
3	127	127	0	0	0	0	1
4	6	111	0	0	0	0	1
4	17	111	0	0	0	0	1
4	127	127	0	0	0	0	1

apDataTable

Obs: as linhas cujos valores do *apDataProt* e *apDataAplic* são ambos iguais a 127 são usadas para acumular o tráfego devido às outras aplicações somadas.

Como mencionado, a MIB é atualizada com a periodicidade dada por **apPeriod**. Durante cada intervalo de monitoração, o tráfego é acumulado em variáveis temporárias, e no final do intervalo o acumulado é escrito na variável da MIB (pelo agente, como veremos adiante). Sendo assim, a MIB contém sempre valores relativos ao tráfego acumulado no intervalo anterior ao atual, somente mudando de valor ao fim de cada intervalo. Supondo um cenário hipotético, vamos acompanhar o processo de acumulação de dados para o domínio 1, protocolo de transporte 6 e protocolo de aplicação 23. Na figura 4.5, cada seta representa um datagrama recebido ou enviado e o número corresponde ao tamanho em bytes. Note-se que na figura 4.5, admite-se que o valor de **apPeriod** é 600, equivalente a um intervalo de 10 minutos.

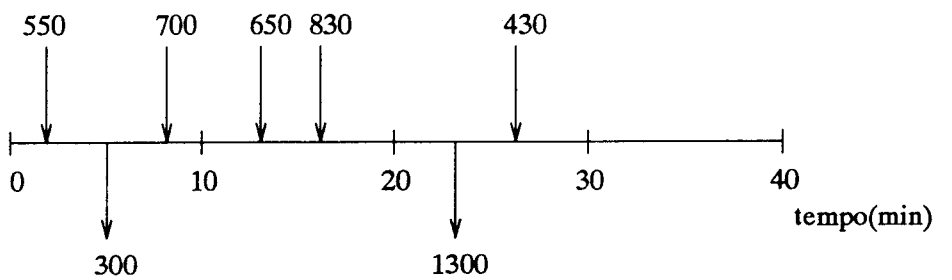

Figura 4.5: Tráfego entre grupos de um domínio

- **No início da medição:**

Indice	Prot	Aplic	BytesEnv	DatEnv	BytesRec	DatRec	Status
1	6	23	0	0	0	0	1

- **Depois da atualização feita aos 10 minutos:**

Indice	Prot	Aplic	BytesEnv	DatEnv	BytesRec	DatRec	Status
1	6	23	300	1	1250	2	1

- **Depois da atualização feita aos 20 minutos:**

Indice	Prot	Aplic	BytesEnv	DatEnv	BytesRec	DatRec	Status
1	6	23	0	0	1480	2	1

- **Depois da atualização feita aos 30 minutos:**

Indice	Prot	Aplic	BytesEnv	DatEnv	BytesRec	DatRec	Status
1	6	23	1300	1	430	1	1

Como na MIB só há dados relativos ao intervalo imediatamente anterior ao corrente, para que se tenham os dados relativos a vários períodos, o gerente deve realizar, com a periodicidade de **apPeriod**, a leitura das variáveis da MIB localizada no agente e armazenar estes dados para posterior consulta. No caso da figura 4.5, o gerente poderia ter sido programado para fazer três requisições (leituras da MIB) com intervalos de 10 minutos, começando, por exemplo, no minuto 12. Assim, na requisição realizada no minuto 12 o gerente armazenaria os dados do tráfego acumulado do minuto 0 ao 10 (período imediatamente anterior ao corrente), no minuto 22, os dados do tráfego acumulado do minuto 10 ao 20 e no minuto 32, os dados do tráfego acumulado do minuto 20 ao 30.

Pode-se notar que a determinação do instante em que deve ser feita a primeira requisição do gerente é bastante flexível já que os dados da MIB permanecem invariáveis durante todo o intervalo de monitoração, somente mudando na transição entre intervalos.

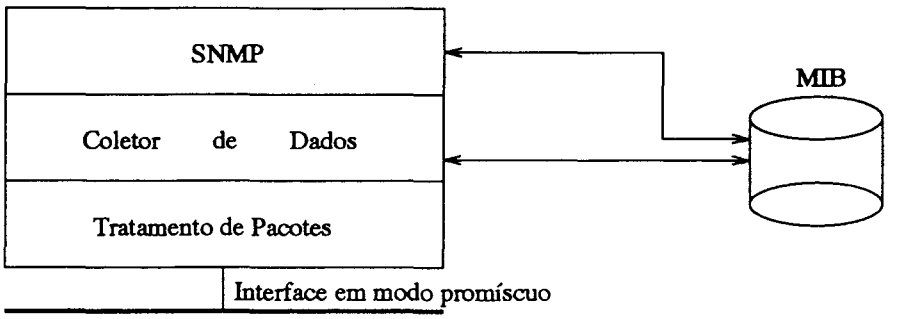

Figura 4.6: Diagrama de Blocos do Agente

4.4 Agente de Gerenciamento

O agente de gerenciamento tem como objetivos a coleta de dados (preenchimento da MIB) e o atendimento das requisições do(s) gerente(s) autorizado(s).

A figura 4.6 mostra os blocos funcionais básicos do agente de gerenciamento.

O módulo de tratamento de pacotes é dependente do meio físico. Este módulo retira o cabeçalho relativo à camada de interface e passa o datagrama para o módulo de coleta de dados.

O módulo de coleta de dados examina o datagrama, compara os endereços IP de origem e destino com os endereços dos domínios da tabela **apDomTable** e as portas de origem e destino com as aplicações classificadas na tabela **apDataTable**, acumula os dados em variáveis temporárias e atualiza a MIB periodicamente.

O módulo SNMP recebe as mensagens SNMP e as decodifica. Realiza a tarefa requerida e devolve uma mensagem de resposta.

O agente executa o seguinte algoritmo;

- início:
  - inicia a variável **apMode** = 2 (modo **não medição**);
  - seta os valores das outras variáveis para zero e deixa as tabelas vazias;
- enquanto chegarem pacotes:
  - o módulo de tratamento de pacotes retira o cabeçalho dependente do meio e passa o datagrama para o módulo de coleta de dados;
  - se o agente estiver em modo de medição, o módulo de coleta de dados faz a contabilização do tráfego; independentemente do modo de medição ou não, se o datagrama contiver uma mensagem SNMP, os

cabeçalhos IP e TCP ou UDP são retirados e a mensagem é passada ao módulo SNMP;

* o módulo SNMP recebe a requisição do gerente e monta uma resposta apropriada que é entregue ao módulo de coleta de dados;
* o módulo de coleta de dados monta os cabeçalhos UDP ou TCP e IP e passa o datagrama para o módulo de tratamento de pacotes;
* o módulo de tratamento de pacotes monta o cabeçalho dependente do meio e solicita ao hardware que envie o pacote;

Como o agente está ligado a uma placa de rede em modo promíscuo, o agente tem acesso a todos os pacotes que representam tráfego efetivo na rede. Isto é. todos os pacotes enviados ao meio físico exceto os que sofrem colisões ou que o *checksum* da camada de enlace esteja errado sendo, portanto, rejeitado pela placa de rede.

Desta forma, o agente de gerenciamento pode ser programado para monitorar o tráfego de domínios discriminado por aplicações e apresentar os dados quando requisitado.

Um fator importante a considerar é o conjunto de domínios cujo tráfego possa ser monitorado em função da localização do agente. São domínios monitoráveis por um determinado agente aqueles que satisfizerem pelo menos uma das seguintes condições:

- Tiverem pelo menos um dos conjuntos que o formam inteiramente contido na mesma rede física do agente. Desta forma, como o agente examina todos os pacotes que circulam na rede, terá acesso a todos os pacotes enviados ou recebidos pelo conjunto.

- A rede à qual está ligado o agente seja passagem obrigatória para o tráfego entre os conjuntos que formam o domínio, assim, o agente examinará todos os pacotes que circularem entre os conjuntos.

Na figura 4.7, são exemplos de domínios monitoráveis pelo agente de gerenciamento localizado na rede 2:

- (A2)-(R23);
- (A1, B1)-(A3, B3);
- (R12, A2)-(A3, B3);

Não são monitoráveis:

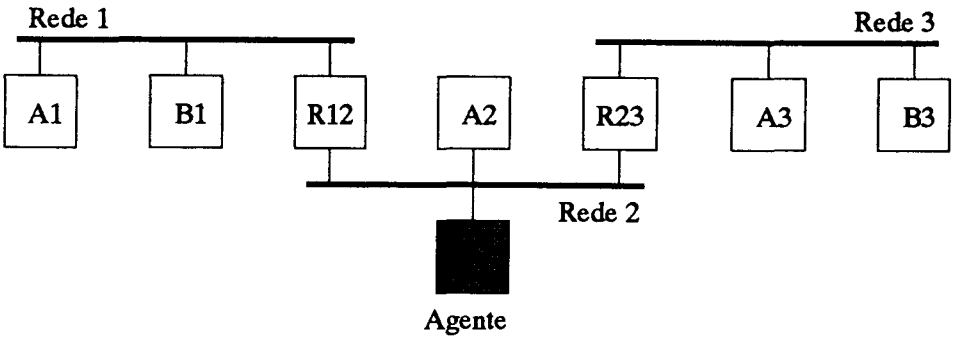

Figura 4.7: Rede com um Agente de Gerenciamento

- (A1)-(B1);
- (B1, R12, A2)-(A1, B1);
- (A1)-(R12);

Obs: podemos observar que no último caso de domínio não monitorável, apesar do elemento de rede R12 pertencer à rede 2, comunica-se com o elemento de rede A1 sem utilizar a interface com a rede 2, portanto o agente de gerenciamento não tem acesso ao tráfego gerado entre eles.

4.5 Gerente de Rede

O gerente tem como objetivo programar agentes a ele submetidos para que estes monitorem o tráfego entre domínios discriminado por aplicações, ler a MIB do agente periodicamente para recolher os dados sobre o tráfego do intervalo anterior e apresentar os dados para o usuário. O gerente comunica-se com o agente através das primitivas SNMP **get-request**, **get-next-request** e **set-request**.

Enviando requisições para escrever na tabela **apDomTable**, controlam-se os domínios a serem monitorados. Na tabela **apDataTable** controlam-se as aplicações que se quer medir para cada domínio. Através da escrita nas variáveis **apMode** e **apPeriod** controla-se o estado do agente e a duração do intervalo entre atualizações da MIB.

Utilizou-se um software gerente que realiza as requisições necessárias e tem ferramentas para exibição de gráficos (figura 4.8).

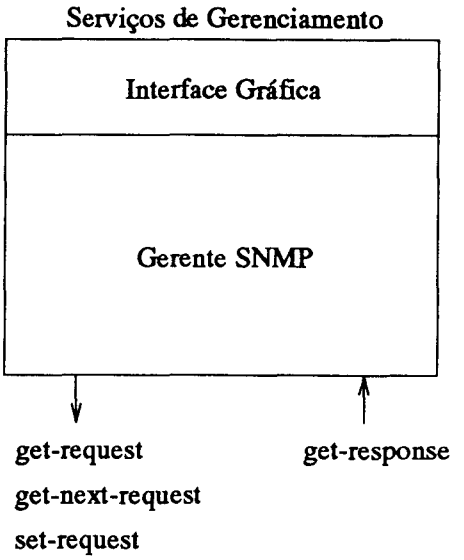

Figura 4.8: Aplicação de Gerenciamento de Rede

Quanto à localização, o gerente só necessita ter conectividade com os agentes a ele submetidos. Entretanto, como o SNMP usa o protocolo de transporte sem conexão da internet, o UDP, é conveniente que a ligação do gerente com os agentes seja confiável.

4.6 Exemplo de Utilização

Na figura 4.9 vemos um exemplo de um conjunto de redes interligadas. Suponhamos que se queira medir o tráfego entre os elementos de rede (A2, B2, C2) e (A3, B3). Suponhamos também que a máquina C3 executa o software agente do nosso modelo e a máquina B1, o software gerente. Por fim, suponhamos que se queira discriminar o tráfego entre os computadores citados devido a transferência de arquivos (FTP), correio eletrônico (SMTP) e outros.

Inicialmente, como vimos, o agente está em modo de não medição e com as tabelas **apDomTable** e **apDataTable** vazias. Vejamos a seqüência de ações para informar ao agente o que monitorar.

Através do aplicativo de gerenciamento o usuário identifica o agente a ser programado, no caso, o elemento de rede C3 com o endereço IP 192.30.18.23.

Escrevendo na tabela **apDomTable**, o gerente informa ao agente o domínio a ser monitorado. Escrevendo na tabela **apDataTable**, o gerente informa as

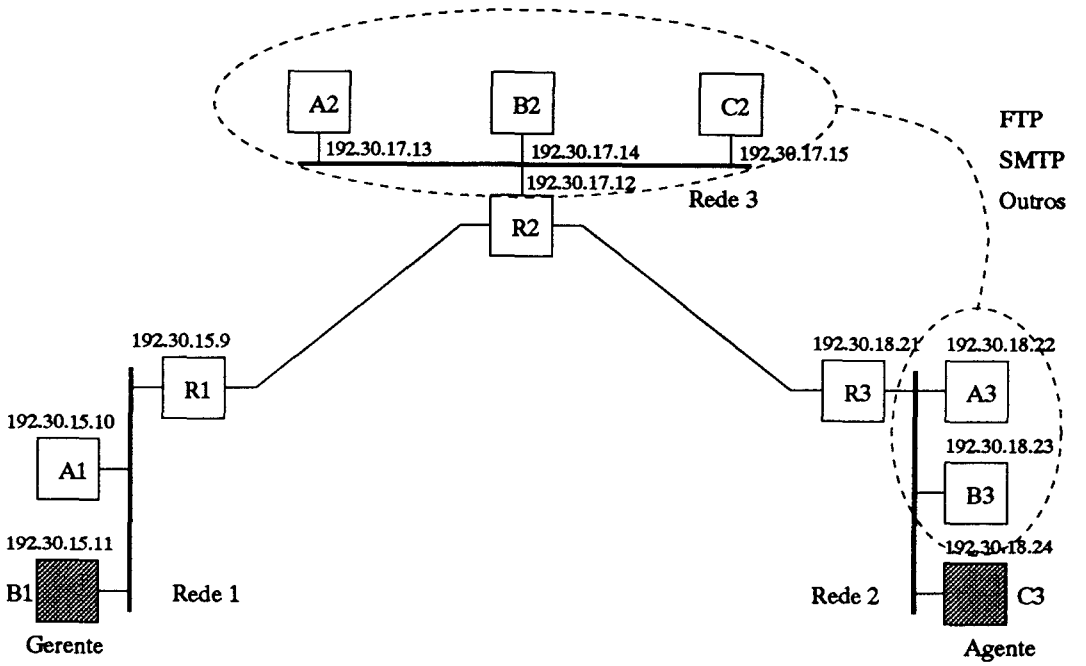

Figura 4.9: Exemplo de domínio e aplicações monitoráveis

aplicações que se quer monitorar.

Suponhamos que se queira medir o tráfego acumulado em períodos de 10 minutos. Então o gerente escreve o valor correspondente (600 segundos) na variável **apPeriod**.

Logo antes de se iniciar as medições, a MIB do agente C3 tem o seguinte aspecto:

apMode = 2 (não medição);
apPeriod = 600;

DomIndice	FirstAddr1	LastAddr1	FirstAddr2	LastAddr2	DomStatus
1	192.30.17.13	192.30.17.15	192.30.18.22	192.30.18.23	1

apDomTable

Indice	Prot	Aplic	BytesEnv	DatEnv	BytesRec	DatRec	Status
1	6	21	0	0	0	0	1
1	6	25	0	0	0	0	1
1	127	127	0	0	0	0	1

apDataTable

O gerente tem mecanismos para armazenar os dados relativos a cada intervalo de medição e apresentá-los. Veremos, no capítulo seguinte, como fazê-lo com mais detalhes.

4.7 Trabalhos Relacionados

Existem alguns trabalhos na área de gerenciamento de aplicações, contudo, com enfoques diferentes do esquema proposto.

Em [RW94] foram criados 5 agentes de gerenciamento com o objetivo de fornecer informações de protocolos de aplicação. O primeiro agente coleta dados do tráfego global de um barramento *ethernet*, classificando tal tráfego nas aplicações **Telnet**, **FTP** e **SMTP**. O agente 2 examina o conteúdo de um arquivo gerado pelo utilitário **Netwatch** e fornece estatísticas do tráfego de algumas aplicações. O agente 3 oferece um serviço de confirmação ao protocolo de correio eletrônico **SMTP**. O agente 4 exibe, de forma "amigável", pacotes TCP-IP trafegando pela rede. Por fim, o agente 5 fornece meios para se monitorar o congestionamento de um barramento *ethernet*. Todos os agentes foram desenvolvidos usando o pacote de gerenciamento SunNet Manager.

Há em [Kil94b] um trabalho definindo uma MIB para o gerenciamento de aplicações de uma maneira geral. Os mesmos autores, em [Kil94a], falam sobre o gerenciamento de correio eletrônico. Define-se uma porção de uma MIB compatível com os protocolos de gerenciamento da Internet (SNMP e SNMPv2), que permite a monitoração de Agentes de Transferência de Mensagens (MTA) presentes nos roteadores.

Em [TdS94] apresenta-se um modelo de gerenciamento do protocolo **MHS X.400** (correio eletrônico). Foram analisados os requisitos para a gerencia do correio eletrônico e realizada uma modelagem, incluindo-se a definição de uma MIB, que cumprisse tais requisitos. Na ocasião da apresentação do trabalho, um protótipo estava sendo implementado.

Em [Cic94] tem-se o desenvolvimento de um agente SNMP para plataformas DOS, tendo sido implementada toda a MIB II, a MIB RMON e alguns objetos adicionais que possibilitam o gerenciamento de protocolos de aplicação como o SNMP, por exemplo.

Em [Car94], [CM94a] e [CM94b] apresenta-se um modelo de gerenciamento do protocolo FTP baseado em domínios. Foi criada uma MIB sub-dividida em quatro grupos que poderiam ser utilizados de acordo com as necessidades de gerenciamento. As informações disponíveis são, entre outras, número de conexões ftp abertas, arquivos mais transferidos. A implementação do modelo foi realizada utilizando o ISODE, sendo construído um sub-agente comunicando-se com agentes SNMP usando o protocolo SMUX.

Os trabalhos [Car94], [CM94a] e [CM94b] foram os precursores do esquema apresentado. Deles extraíram-se as idéias de gerenciar aplicações e coletar dados direto da rede, lá apresentado como sugestão. O agente 1 de [RW94] apresenta algumas semelhanças com o esquema de gerenciamento de tráfego por aplicações aqui exposto, não permitindo, no entanto, a definição de domínios de gerenciamento nem a livre escolha da aplicação a gerenciar. [Kil94a] e [TdS94] apresentam esquemas para o gerenciamento de uma determinada aplicação, no caso o correio eletrônico, com um nível maior de detalhes. [Cic94] mostra o gerenciamento de aplicações como validação do seu trabalho, não sendo o tópico principal.

Capítulo 5

Implementação do Esquema Proposto

Neste capítulo descrevemos os aspectos da implementação do sistema desenvolvido que segue o modelo de gerenciamento de aplicações proposto.

O agente de gerenciamento, cuja função é de coletar os dados da MIB e comunicar-se através do protocolo de gerenciamento SNMP, foi desenvolvido usando-se um IBM-PC 486, com sistema operacional DOS, conectado a uma rede padrão Ethernet através de uma placa de rede IBM compatível com o padrão NE2000. O software foi desenvolvido em **Borland C**. O software gerente utilizado foi o SunNet Manager, rodando em estações SUN.

5.1 Base de Informações de Gerenciamento

Foi criada uma estrutura de dados para armazenar a MIB no agente. Cada objeto tem os seguintes campos:

- **name:** nome da variável SNMP (string);
- **id:** identificador de objeto (vetor de inteiros);
- **id_len:** comprimento do identificador de objeto (inteiro);
- **syntax:** código da sintaxe associada ao objeto (byte);

- **access:** regra de acesso ao objeto; pode ser somente leitura - RO, somente escrita - WO ou leitura e escrita - RW (string);
- **num_value:** conteúdo numérico da variável da MIB, se for o caso (inteiro longo);
- **char_value:** conteúdo caractere da variável da MIB, se for o caso (string);
- **acumulator:** acumula valores temporários do objeto.

A MIB é construída criando-se uma lista de tais registros. Quando se trata de uma variável numérica da MIB, o campo **num_value** é utilizado, caso contrário, o campo **char_value** é utilizado.

Como exemplo, o objeto **sysLocation** poderia ter os seguintes campos:

- **name:** sysLocation
- **id:** 1.3.6.1.2.1.1.6
- **id_len:** 8
- **syntax:** caractere
- **access:** RO
- **num_value:** não utilizado
- **char_value:** Laboratório de Redes - DCC
- **acumulator:** não utilizado

Algumas funções que manipulam a lista de objetos, ou seja, a MIB, foram implementadas, entre elas: mib-fill (adiciona um objeto à MIB), mib-delete (extrai um objeto da MIB), mib-point (aponta para um objeto da MIB). A lista é manipulada de forma que esteja sempre ordenada pelo campo **id**.

São mantidas na lista do agente apenas os objetos tipo folha da árvore da MIB. No nosso caso, implementamos o grupo **System** da MIB II [MRe91], que serve para identificar o agente, e o grupo **apTraffic**, criado para monitorar o tráfego de aplicações. Ao ser iniciado, o agente cria a lista da figura 5.1.

Quando alguma linha das tabelas deve ser inserida, seguindo a regra do SNMP, o gerente deve enviar uma requisição de escrita (**set-request**) contendo todos os objetos componentes da linha. Quando o agente recebe tal requisição,

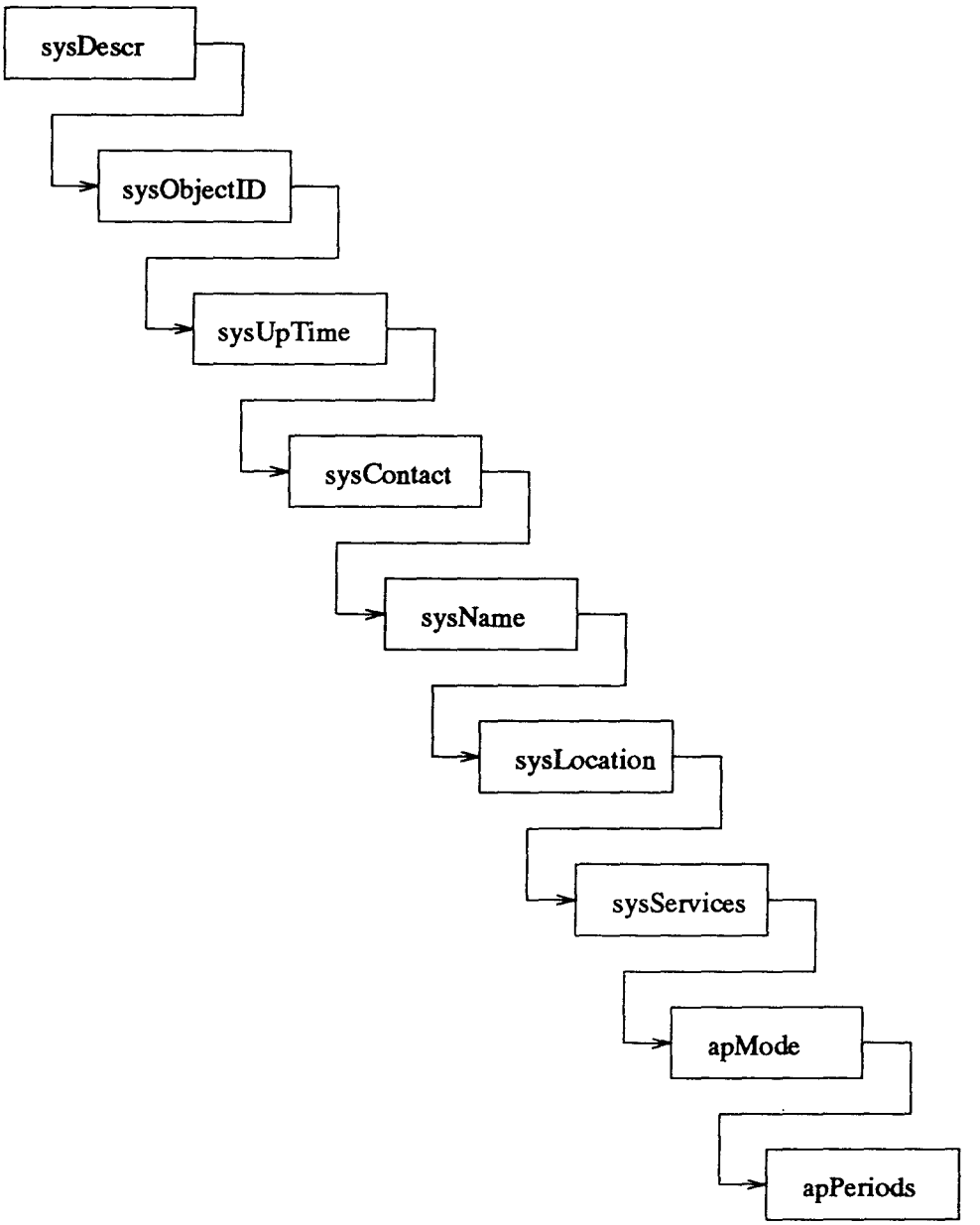

Figura 5.1: Lista de Objetos Inicial

insere novos elementos na lista, correspondentes aos objetos da linha da tabela em questão.

Os objetos que contém, de fato, os valores do tráfego monitorado estão na tabela **apDataTable**. A figura 5.2 mostra a parte da lista correspondente aos objetos de uma linha da tabela **apDataTable**.

Os objetos **apDataIndice**, **apDataProt** e **apDataAplic** identificam um domínio, um protocolo de transporte utilizado e uma aplicação. Os objetos **apDataBytesEnv**, **apDataDatEnv**, **apDataBytesRec** e **apDataDatRec** correspondentes representam, respectivamente, o número de bytes enviados do primeiro grupo ao segundo grupo do domínio, o número de datagramas enviados do primeiro grupo ao segundo grupo do domínio, o número de bytes recebidos pelo primeiro grupo do segundo grupo do domínio e o número de datagramas recebidos pelo primeiro grupo do segundo grupo do domínio.

O valor correspondente ao valor da MIB de cada um dos quatro objetos citados acima é armazenado no campo **num_value** de cada objeto.

5.2 Agente de Gerenciamento

O software que implementa o agente é constituído das seguintes partes (ver figura 5.3):

- **Packet Driver:** Módulo servindo de interface entre o hardware de rede e o sistema operacional do micro. Fica residente em memória.
- **Waterloo TCP:** Módulo que implementa o conjunto de protocolos TCP-IP para microcomputadores IBM-PC e compatíveis. Utilizamos apenas as suas rotinas de gerenciamento de memória para armazenamento de pacotes de rede, recebimento e envio de pacotes de rede.
- **Coleta de Dados:** Módulo desenvolvido no projeto, cuja função é receber *frames* tipo Ethernet do módulo Waterloo TCP (wattcp), fazer as atualizações necessárias na MIB referentes à aplicações que utilizem o protocolo de rede IP e passar para o módulo SNMP os pacotes correspondentes. Além disso, recebe os pacotes SNMP de resposta, monta os cabeçalhos UDP, IP e Ethernet e envia pela rede usando o wattcp.
- **SNMP:** Módulo responsável pela comunicação SNMP do agente. Recebe o pacote SNMP do módulo de coleta de dados, decodifica a estrutura ASN.1,

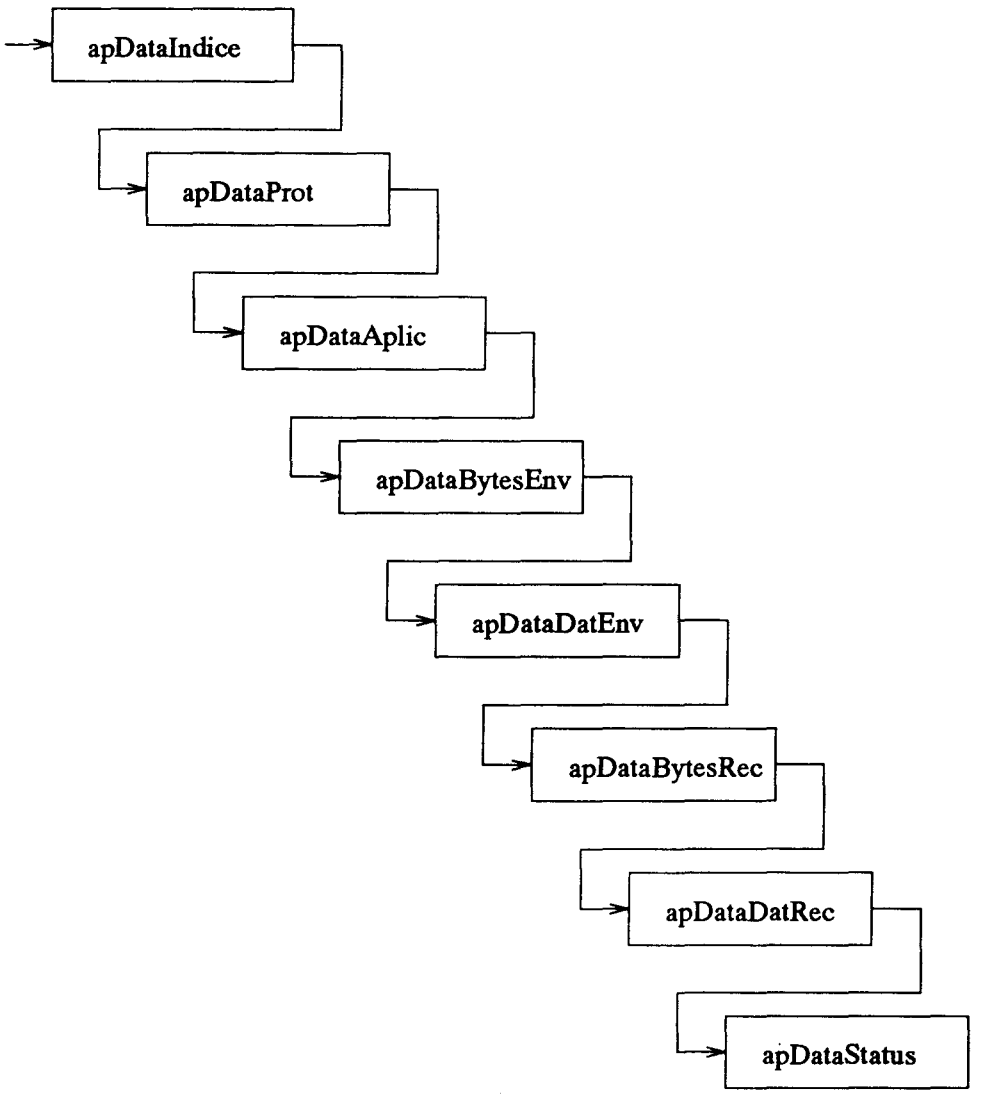

Figura 5.2: Lista Correspondente a uma Linha de **apDataTable**

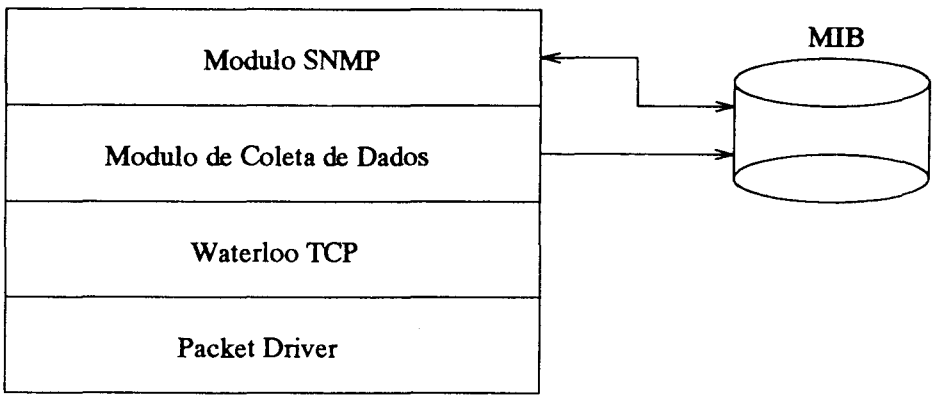

Figura 5.3: Camadas de Software do Agente

realiza as ações requeridas e monta o pacote SNMP de resposta que é entregue ao módulo de coleta de dados.

Vejamos, em detalhe, cada um dos módulos.

5.2.1 Packet Driver

O packet driver provê uma interface de programação simples que permite acesso aos pacotes de rede no nível de enlace.

Nós utilizamos a versão 1.09 do packet driver [Rom89], produzida pela FTP software Inc. e obtida por *anonymous ftp.*

Esta interface permite que a implementação da pilha de protocolos seja independente da versão ou modelo de interface de rede a ser utilizada. Aplicações que usem o packet driver podem ser portadas sem modificações quando o hardware de rede for modificado, desde que se forneça um novo packet driver.

- **Identificando interfaces de rede:** As interfaces de rede são identificadas para o packet driver por uma tripla de inteiros: classe, tipo e número. A classe indica qual o tipo de mídia que a interface suporta (DIX Ethernet, IEEE 802.3 Ethernet, Token Ring, entre outras). O tipo identifica a instância particular de interface que suporta a classe (3Com, NE1000, NE2000, entre outras). O número serve para distinguir entre interfaces de uma mesma classe e tipo, em uma determinada máquina.

- **Iniciando as operações:** O packet driver é invocado por uma interrupção de software cujo número pode estar entre 60H e 80H (é um valor configurável

para evitar conflito com interrupções pré-existentes). O programa de instalação do packet driver provê mecanismos para que o usuário especifique a interrupção escolhida.

- **Interface de Programação:** Todas as funções são acessadas através da interrupção descrita anteriormente. O registrador AH contém o código da função a ser invocada. Vamos descrever as principais funções implementadas nesta versão de packet driver:

  - driver_info(): Retorna informação sobre a interface. A interrupção deve ser chamada com AH=1 e AL=255 e retorna a versão, classe, tipo, número, nome e funcionalidade da interface.
  - access_type(): Inicia o acesso a pacotes de um determinado tipo. Quando um pacote é recebido, a função **receiver** é chamada duas vezes pelo packet driver. Na primeira, devolve um buffer para a alocação do pacote e na segunda realiza operações, pré-determinadas pelo usuário, sobre o pacote. A função **access_type** devolve um *handle* através do qual se identifica um canal com a rede.
  - release_type(): Encerra o acesso a pacotes associados a um determinado *handle*. O *handle* passa a não mais ser válido.
  - send_pkt(): Transmite um buffer de dados pela rede. A aplicação que chama a função deve fornecer o pacote completo, incluindo cabeçalhos referentes a redes locais.
  - set_rcv_mode(): Configura o modo de recepção de uma interface associada a um determinado *handle*. Os seguintes modos são possíveis:

    * **modo 1:** desativar a função receiver;
    * **modo 2:** receber apenas pacotes destinados à interface;
    * **modo 3:** modo 2 + pacotes de *broadcast*;
    * **modo 4:** modo 3 + pacotes de multicast limitados;
    * **modo 5:** modo 3 + pacotes de multicast;
    * **modo 6:** todos os pacotes (modo promíscuo).

    É preciso notar que nem todas as interfaces suportam todos os modos, além disso, o modo de recepção afeta todos os pacotes recebidos pela interface associada ao *handle*, e não apenas aqueles referentes ao *handle* específico.

5.2.2 Waterloo TCP

O conjunto de programas Waterloo TCP [Eng91] é uma implementação completa do TCP-IP para DOS. Nós utilizamos apenas alguns módulos do Waterloo TCP: os que lidam com o acesso às rotinas do packet driver.

Os módulos que utilizamos gerenciam a alocação e liberação da memória à medida que os pacotes chegam. Desta forma, ao invés de ter de tratar os pacotes no instante em que chegam, a aplicação trata os pacotes quando lhe for mais conveniente.

As funções utilizadas foram as seguintes:

- pkt_init(): Chama a rotina **access_type** do packet driver e cria um buffer, de tamanho configurável, onde armazena os pacotes que eventualmente chegarem;
- pkt_release(): Chama a rotina **release_type** do packet driver;
- pkt_send(): Chama a rotina **send_pkt** do packet driver;
- pkt_received(): Inspeciona se o buffer criado pela função **pkt_init** tem algum pacote não tratado. Se tiver, retorna um ponteiro para o início do pacote mais antigo no buffer.
- pkt_buf_release(): Marca um pacote do buffer como já tratado, permitindo que o **wattcp** libere o espaço de memória correspondente.
- pkt_init_all(): Função **pkt_init** modificada para que o modo de recepção seja iniciado como modo=6, ou seja, modo de recepção promíscuo.

Vejamos um exemplo de função que captura n pacotes, imprime o primeiro byte do buffer de cada um dos pacotes e o libera:

```
void print_first( int n )
{
  int i;
  unsigned char *ptr:

  ptr = NULL;
  pkt_init();
  for( i=0; i<n; i++ ) {
    while( !ptr )
```

```
      ptr = pkt_received();
    printf("Primeiro byte: %d\n", *ptr);
    pkt_buf_release( ptr + 14 );
  }
  pkt_release();
}
```

5.2.3 Coleta de Dados

Este módulo examina os pacotes, acumula os dados e passa para o módulo SNMP o PDU SNMP, se for o caso. E ainda, encapsula PDUs enviados pela camada SNMP para serem enviados na rede.

Vamos ver os passos referentes à coleta de dados:

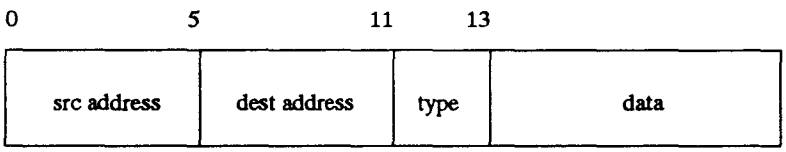

Figura 5.4: Cabeçalho Ethernet

1. O módulo Coleta de Dados consulta a função **pkt_received()** do wattcp. Quando há pacotes novos, retira o cabeçalho Ethernet (figura 5.4) [Tan91] e segue para o passo 2. Quando não, consulta-se o relógio. Ao se verificar que o intervalo de monitoração, cujo período é dado por **apPeriod**, acabou, para todos os objetos **apDataBytesEnv**, **apDataDatEnv**, **apDataBytesRec** e **apDataDatRec** presentes na lista (MIB) do agente, faz-se:

   - copia-se o conteúdo do campo **acumulator** para o campo **num_value**;
   - faz-se o conteúdo do campo **acumulator** igual a zero.

   Volta ao início do passo 1.

2. Armazenam-se os endereços IP de origem e destino e o comprimento do datagrama (ver figura 2.4), e as portas origem e destino (ver figuras 2.7 e 2.6). A função **pkt_buf_release()** é chamada para liberar a memória ocupada pelo pacote de rede;

3. Consulta-se a tabela **apDomTable** da MIB para verificar se há domínio(s) em que um dos seguintes casos, ou ambos, sejam verdadeiros:

- **pacote enviado** apFirstAddr1 $\leq$ EndOrigemPacote $\leq$ apLastAddr1 e apFirstAddr2 $\leq$ EndDestinoPacote $\leq$ apLastAddr2
- **pacote recebido** apFirstAddr1 $\leq$ EndDestinoPacote $\leq$ apLastAddr1 e apFirstAddr2 $\leq$ EndOrigemPacote $\leq$ apLastAddr2

Se nenhum domínio satisfizer às condições acima, segue-se o passo 5.

4. Consulta-se a tabela **apDataTable** da MIB e verifica-se, para o(s) domínio(s) encontrado(s) no passo anterior, se a aplicação, caracterizada por uma porta que pode ser tanto a de origem quanto a de destino, é uma das monitoradas. Se for o caso, haverá na MIB objetos **apDataIndice**, **apDataProt** e **apDataAplic** correspondentes ao domínio, ao protocolo de transporte e à *well-known-port* da aplicação. De acordo com o passo 3, uma das seguintes situações, ou ambas (para o caso dos grupos que compõem o domínio terem uma interseção não vazia e o pacote seja simultaneamente enviado e recebido), podem acontecer:

   - **Pacote Enviado:** neste caso os objetos **apDataBytesEnv** e **apDataDatEnv** correspondentes têm, cada um, os seus campos **acumulator** incrementados do tamanho do datagrama (armazenado no passo 2) e de uma unidade, respectivamente;
   - **Pacote Recebido:** os objetos **apDataBytesRec** e **apDataDatRec** correspondentes têm, cada um, os seus campos **acumulator** incrementados do tamanho do datagrama e de uma unidade, respectivamente;

   Caso contrário, se a aplicação não estiver sendo especificamente monitorada, localiza-se na lista de objetos do agente aqueles onde **apDataIndice** seja correspondente ao domínio, **apDataProt** = 255 e **apDataAplic** = 255. Esse conjunto representa a soma do tráfego gerado pelo restante das aplicações. Realizam-se os procedimentos de acumulação de forma análoga ao caso acima.

5. Se o datagrama for referente a uma requisição SNMP de um gerente, porta de origem = 161, passa-se um ponteiro para o início do pacote SNMP para o módulo SNMP e aguarda-se a resposta à requisição. Caso contrário, descarta-se o datagrama e volta-se ao primeiro passo.

6. Quando o módulo SNMP devolve a resposta à requisição SNMP, o módulo de coleta de dados monta os cabeçalhos UDP, IP e Ethernet de resposta. Os dados referentes a endereços origem e destino Ethernet, endereços de origem

e destino IP e portas de origem e destino são invertidos no novo pacote. O *checksum* UDP é zero (possibilidade do UDP para simplificar aplicações), o comprimento do novo pacote é calculado e escrito no cabeçalho IP e o *checksum* do cabeçalho IP é calculado e escrito no lugar apropriado (ver figura 2.4). Com o novo pacote formado, chama-se a função do **wattcp** que envia o pacote pela rede.

Note-se que toda requisição SNMP feita ao agente irá interagir com o campo **num_value**, no caso de variáveis da MIB numéricas.

5.2.4 Módulo SNMP

Este módulo tem a função de receber requisições SNMP, fazer as consultas ou alterações correspondentes na MIB, e enviar respostas.

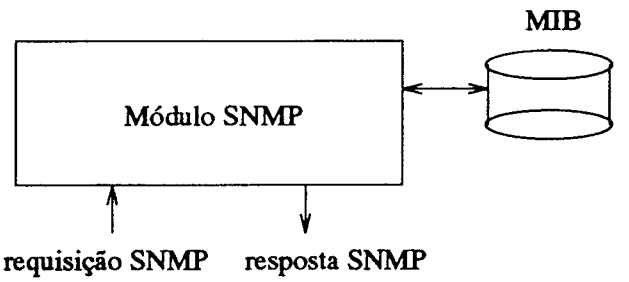

Figura 5.5: Interface do Módulo SNMP

Neste módulo, utilizamos duas funções de manipulação de objetos ASN.1 extraídas da SNMPLIB [Cro91a, Cro91b]:

- **getobjptrs():** dado um ponteiro para o início de objeto ASN.1, devolve os seguintes dados:
  - ponteiro para o campo de tipo;
  - ponteiro para o campo de comprimento;
  - ponteiro para o campo de dados;
  - ponteiro para o final do campo de dados;
  - ponteiro para o final do objeto.

  Esses dados ficam armazenados em uma variável cuja estrutura também é definida na SNMPLIB.

- **getntxobj()**: dado um ponteiro para o início de um objeto ASN.1, devolve um ponteiro para o início do próximo objeto ASN.1.

Vamos ver os passos referentes à implementação do módulo SNMP (ver figura 3.6

1. Testa-se o campo versão;
2. Testa-se o nome da comunidade;
3. Armazena-se o tipo de pacote SNMP (get-request, get-next-request ou set-request);
4. Constrói-se uma lista com as identidades dos objetos da MIB que a requisição contém e os seus valores;
5. Realiza-se um teste que identifica exatamente cada variável da lista para os casos de get-request ou set-request ou identifica o objeto da MIB lexicograficamente mais próximo, no caso do get-next-request. Se houver erro, segue-se o passo 7.
6. Montagem de pacote sem erro: neste caso, a lista de variáveis a ser lida ou escrita está com as identificações corretas. Se o tipo do pacote for de leitura (get-request ou get-next-request):

   - cada variável da MIB presente na lista é lida;

   Caso contrário (set-request):

   - os valores das variáveis da MIB presentes na lista são atualizados;

   Continua no passo 8.
7. Montagem de pacote com erro: a primeira variável da lista que apresenta erro é passada como valor de **error-index** e o tipo do erro é passado em **error-status**.
8. O pacote de resposta é montado de acordo com o modelo SNMP e é entregue ao módulo de coleta de dados para que seja enviado ao gerente.

5.3 Gerente

Como já mencionado, o software gerente utilizado foi o **SunNet Manager**, versão 2.2, executado em estações SUN.

5.3.1 SunNet Manager

O SunNet Manager compreende uma série de ferramentas e serviços para serem utilizados em tarefas de gerenciamento de redes [Sun93a, Sun93b].

- **Aplicações de Gerenciamento e Agentes**

  O pacote provê aplicações de gerenciamento e agentes. As aplicações de gerenciamento são processos que permitem ao usuário iniciar tarefas de gerenciamento e coletar dados. Agentes são processos que acessam o elemento de rede a ser gerenciado quando requisitados por uma estação de gerenciamento.

  Grande parte dos agentes providos pelo SunNet Manager fornecem informações de entidades em estações de trabalho Sun. Entretanto, o SunNet Manager provê também um agente *proxy*, que possibilita o gerenciamento de entidades que não as da Sun. O SNMP *proxy* comunica-se com os agentes SNMP.

  As aplicações de gerenciamento e os agentes SunNet comunicam-se através de RPC (Remote Procedure Call). Os agentes *proxy* traduzem o protocolo RPC para o protocolo que os elementos de rede a serem gerenciados utilizam.

- **Suporte ao SNMP**

  O agente *proxy* SNMP é executado em estações Sun. As aplicações de gerenciamento do SunNet Manager comunicam-se com o agente *proxy* através de RPC e este, por sua vez, comunica-se com os outros dispositivos através do SNMP (ver figura 5.6) [Sun93b].

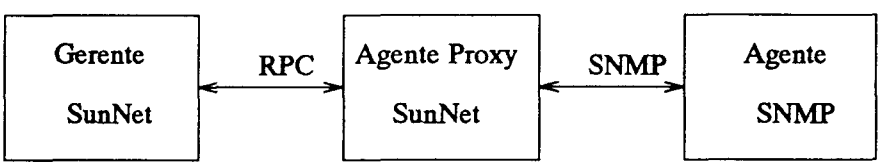

Figura 5.6: Agente *Proxy* do SunNet Manager

O agente *proxy* possibilita que se possa gerenciar tanto a MIB SNMP padrão quanto MIBs específicas. O agente *proxy* usa um arquivo tipo ".schema" (arquivos .schema são o formato usado pelo SunNet Manager para representar a MIB internamente) para mapear os objetos SNMP no formato SunNet

Manager. Sendo assim, o arquivo .schema contém o mesmo conjunto de definições que a MIB, mudando somente a forma de representação.

O arquivo **snmp-mibII.schema** [Sun93a], por exemplo, é fornecido pelo SunNet Manager e descreve a MIB II. Além disso, utilizando o utilitário **mib2schema** incluído no pacote, pode-se converter arquivos que descrevam MIBs específicas do formato ASN.1 para o formato utilizado pelo SunNet Manager, de forma que a aplicação de gerenciamento possa acessar objetos definidos por empresas ou pelos próprios usuários.

- **Leitura de Atributos**

  Pode-se requisitar a um agente que retorne valores de atributos. As requisições podem ser realizadas uma única vez, ou ainda, pode-se requerer relatórios periódicos de um ou mais atributos. Neste último caso, pode-se especificar as seguintes características:

  - quantidade de leituras requeridas;
  - intervalo entre leituras;
  - armazenamento das leituras em arquivos;
  - possibilidade de visualização gráfica.

- **Escrita em Atributos**

  O SunNet Manager fornece um utilitário que permite mudar o valor de variáveis de gerenciamento: **set-tool**.

  Através do **set-tool** pode-se enviar requisições de escrita.

- **Análise de Dados**

  - Analisando dados armazenados: **Results Browser**

    O utilitário **Results Browser** permite que se recuperem e organizem dados.

    Arquivos de dados armazenados de requisições de leitura a variáveis da MIB podem ser examinados em detalhe. Caso as requisições sejam periódicas, os dados relativos a cada leitura podem ser vistos de forma clara e organizada.

  - Analisando dados armazenados: **Results Grapher**

    O utilitário **Results Grapher** pode ser utilizado para visualizar graficamente atributos de arquivos armazenados, podendo ser chamado diretamente do utilitário **Results Browser**.

Selecionando-se os atributos que se deseja, o utilitário **Results Grapher** gera gráficos em função do tempo, sendo um instrumento bastante útil na análise de dados de gerenciamento.

5.3.2 Utilização do Gerente

Como o modelo proposto para o gerenciamento do tráfego de aplicações apresenta uma extensão da MIB SNMP, para que o SunNet Manager tivesse conhecimento das variáveis a serem gerenciadas, a nova MIB, descrita no formato ASN.1, foi compilada através da ferramenta **mib2schema**, gerando um arquivo que indica ao SunNet Manager quais as novas variáveis e suas sintaxes.

Isto feito, podem ser usadas todas as facilidades de requisições e gráficos presentes no SunNet Manager.

Para monitorar o tráfego de aplicações usando o SunNet Manager, seguiram-se os seguintes passos:

1. Definem-se os domínios a serem monitorados e escrevem-se os valores a eles correspondentes na tabela **apDomTable** do agente utilizando o **set tool**;

2. Definem-se as aplicações a monitorar para cada domínio e escrevem-se os valores na tabela **apDataTable** do agente utilizando o **set-tool**;

3. Define-se o intervalo de tempo, em segundos, dos períodos de gerenciamento e escreve-se na variável **apPeriod** do agente utilizando o **set-tool**;

4. Define-se o envio de requisições periódicas, periodicidade igual à definida na variável **apPeriods**, que lêem os valores da tabela **apDataTable** e guardam em um arquivo de *log*. O número de intervalos a serem monitorados também deve ser definido;

5. Terminado o processo de monitoração, tem-se um arquivo de *log* contendo dados relativos ao tráfego de aplicações dos domínios classificados na tabela **apDomTable**. As ferramentas *browser* e *grapher* do SunNet Manager são utilizadas para a visualização do arquivo de *log* e traçado de gráficos.

5.4 Resultados

Foram realizadas algumas medidas do tráfego de aplicações na rede do Departamento de Ciência da Computação (DCC) da Unicamp. A topologia da rede está apresentada na figura 5.7.

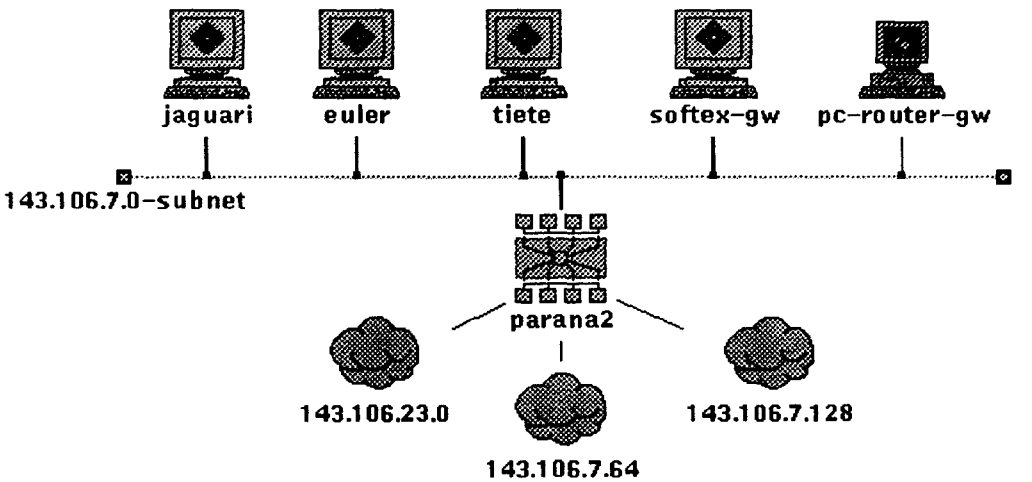

Figura 5.7: Rede do DCC - Unicamp

A máquina Jaguari é uma das utilizadas como servidora de disco. A máquina Tiete é a servidora de correio eletrônico e a roteadora de saída do DCC. Uma informação útil para o gerenciamento desta rede é relacionada à quantidade do tráfego do *backbone*, que é a sub-rede **143.106.7.0**, devido a correio eletrônico para fora do departamento e quanto é devido a serviço de disco da Jaguari.

O agente de gerenciamento foi colocado no *backbone* na rede do DCC. O software foi executado no **pc-router-gw**.

Para o caso da Jaguari, definimos dois domínios. O primeiro formado, por um lado, pelas sub-redes **143.106.7.64** e **143.106.7.128** e por outro pela máquina Jaguari. O segundo domínio formado por um lado pela sub-rede **143.106.23.0** e por outro, também pela máquina Jaguari. Configuramos a duração do intervalo de monitoração para 1200 segundos, ou seja, 20 minutos, e efetuamos medições das 9:00h às 11:40, totalizando, portanto, 5 intervalos de 20 minutos. Além disso, monitoramos especificamente a aplicação RPC (porta 111) sob UDP (protocolo 17), já que sabe-se que o maior usuário do RPC sob UDP é o NFS (Network File System).

A figura 5.8 apresenta um gráfico mostrando o tráfego, em número de bytes, das três redes acima mencionadas para a máquina Jaguari devido a RPC (bytes enviados e recebidos) [1].

Para o caso da Tiete, definimos dois domínios análogos aos anteriores, exceto pela presença da máquina Tiete ao invés da Jaguari. Configuramos a duração do intervalo para 600 segundos, ou seja, 10 minutos, e efetuamos medições das 13:00h às 13:50h, totalizando, portanto, 5 intervalos de 10 minutos. Além disso, monitoramos especificamente a aplicação SMTP (porta 25) sob TCP (protocolo 6).

Na figura 5.9, o gráfico mostra o tráfego somado dos dois domínios descritos, devido ao correio eletrônico (SMTP) para bytes enviados e bytes recebidos.

Através destes dados temos uma informação útil a respeito do tráfego na rede do DCC-Unicamp, e podemos ter idéia das informações que o esquema proposto pode oferecer. Pode-se verificar como é possível monitorar o tráfego entre grupos diversos de computadores, para aplicações diferentes, e com intervalos de monitoração adequados a cada caso.

[1] Os valores obtidos para cada um dos domínios foram somados, respectivamente, e apresentados no mesmo gráfico

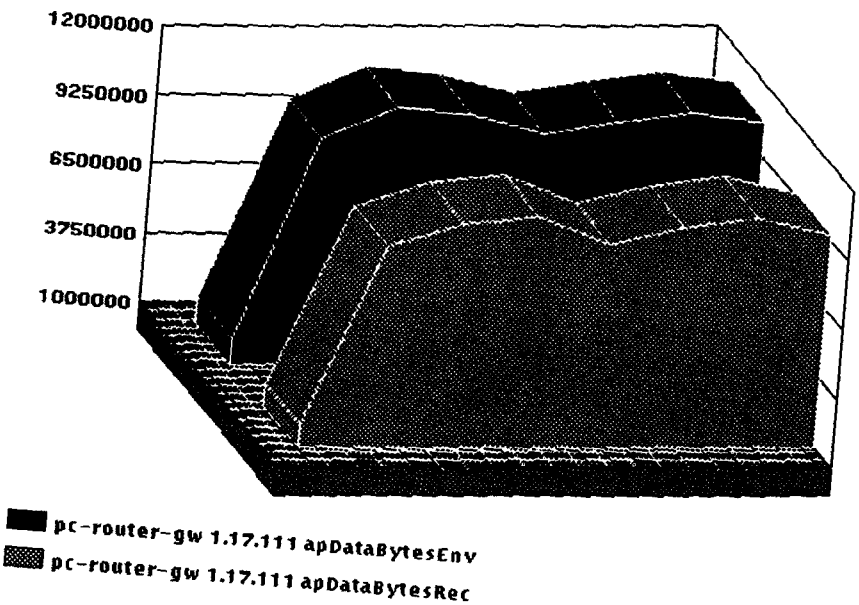

Figura 5.8: Tráfego de RPC para a máquina Jaguari

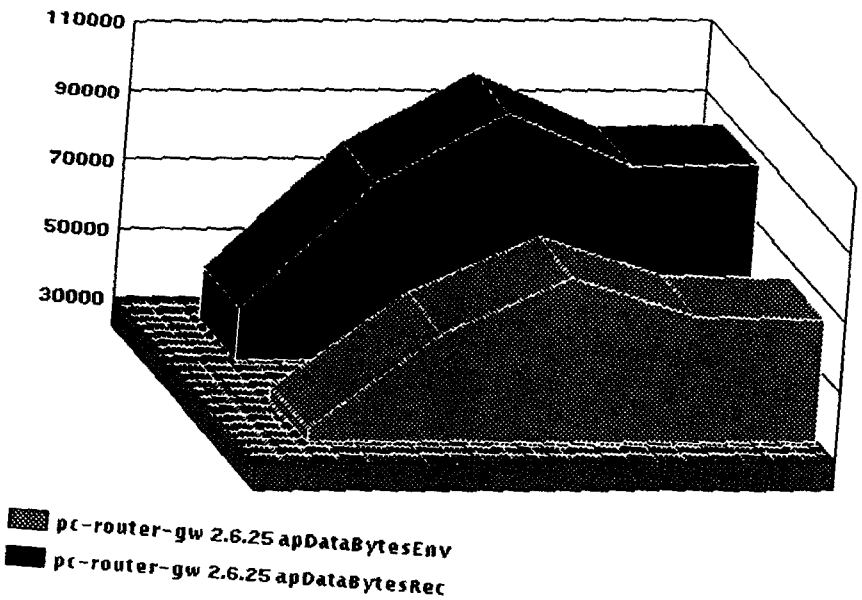

Figura 5.9: Tráfego de SMTP para a máquina Tiete

5.5 Testes da Implementação

Como visto, o agente de gerenciamento deve ser ligado à rede através de uma placa configurada para o modo de recepção promíscuo. Isso implica que o agente receberá todos os pacotes que circulam pela rede e que não tenham sofrido colisões nem estejam com os *checksum* errados. Por serem potencialmente muitos pacotes, foram realizados dois testes para verificar se o agente estava contabilizando todos os pacotes que deveria.

1. No primeiro teste utilizamos o software monitor de rede **Gobbler**, no modo de contabilidade de pacotes. Neste modo, o **Gobbler** é um software bem mais "leve" que o agente desenvolvido. Como só havia um PC diretamente conectado ao *backbone* do DCC (ver figura 5.7) fez-se uma série de 20 medidas de 1 minuto cada, 10 com o software **Gobbler** e 10 com o agente de gerenciamento, intercaladas. Para que o agente contabilizasse todos os pacotes da rede, criou-se um domínio onde os dois grupos tinham: **firstAddr = 0.0.0.0** e **lastAddr = 255.255.255.255**, sem especificar nenhuma aplicação. Além disso, foi criado um outro domínio e associado a este três aplicações para que o agente do esquema fosse testado de forma realista. Analisando as medidas, vimos que os máximos contabilizados tinham diferenças da ordem de 5% (os máximos foram 21.549 pacotes para o **Gobbler** e 22.328 pacotes para o agente de gerenciamento), o que poderia ser tolerado pelas medidas não serem simultâneas.

2. No segundo teste, procuramos eliminar o problema da falta de sincronia. Para tanto, usamos o agente com a mesma configuração do primeiro teste, com exceção do tamanho do intervalo de medição que passou para 10 minutos, e executamos o comando UNIX **netstat -i** nas três principais máquinas do *backbone*: tiete, parana e jaguari (somente executou-se o **netstat -i** nas três máquinas citadas pois as máquinas **euler** e **softex-gw** dão acesso a grupos auto-contidos, cuja utilização do *backbone* é mínima, ver figura 5.4). Executando o **netstat -i** aproximadamente no mesmo instante nas três máquinas, no início e no fim de um período de medição, obtivemos uma forma de comparar os pacotes capturados pelo agente com as estatísticas do *kernel* do UNIX. Neste caso, em duas medições, a diferença entre as medidas ficou em 2%, em média (as médias foram 85.343 pacotes medidos com o **netstat -i** e 83.249 pacotes medidos pelo agente do esquema de gerenciamento de aplicações.

5.6 Ferramentas de Apoio

Duas ferramentas foram desenvolvidas para complementar as informações disponíveis através do esquema proposto. Ambas foram desenvolvidas em ambiente UNIX (SunOs) e realizam operações no arquivo de *log* gerado pelo SunNet Manager com as informações do tráfego de aplicações nos domínios.

A primeira ferramenta desenvolvida foi um programa que fornece o percentual de tráfego para cada aplicação dada uma série de medidas. Este programa funciona especificamente quando a série de medidas é obtida através do SunNet Manager, armazenando-as em um arquivo de *log*.

O aplicativo lê o arquivo de *log* gerado pelo SunNet Manager e para cada domínio soma os valores respectivos a cada aplicação gerenciada fornecendo os percentuais de cada uma para toda a série de medidas.

Existe a opção, ao se executar o programa que calcula os percentuais, de não incluir o tráfego correspondente a *outras aplicações*. Ou seja:

```
marumbi% percent {\em arquivo-log}  (trafego total)
ou
marumbi% percent -e {\em arquivo-log} (trafego das aplicacoes especificadas)
```

Como exemplo, foi monitorado o tráfego total do *backbone* do DCC, classificando-o em transferência de arquivos, correio eletrônico, RPC e outros. O gráfico da figura 5.10 mostra o tráfego, em bytes, para cinco intervalos de 30 minutos cada, das aplicações.

Executando-se o programa que fornece os percentuais sobre o arquivo de *log* gerado pelo SunNet Manager, obtiveram-se os seguintes resultados:

```
Percentual por aplicacao:

1.6.21          1.16%
1.6.25          2.17%
1.17.111       96.67%
```

Em um segundo exemplo, foi monitorado o tráfego enviado pelos computadores do DCC para fora do departamento, classificando-o em tráfego devido a *login* remoto, correio eletrônico, WWW e RPC. O gráfico da figura 5.11 mostra o tráfego monitorado em quatro intervalos de 30 minutos.

Executando-se, mais uma vez, o programa que fornece os percentuais sobre o arquivo de *log* gerado pelo SunNet Manager, obtiveram-se os seguintes resultados:

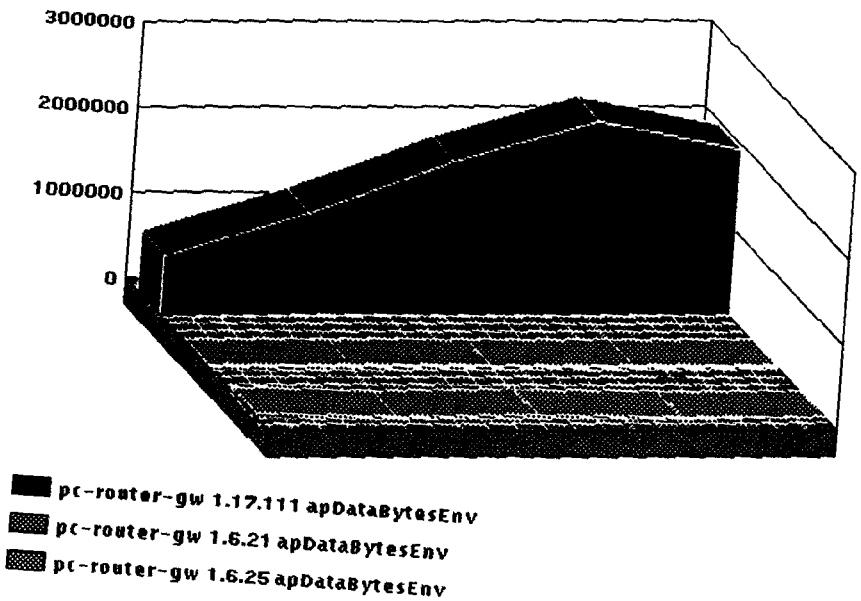

Figura 5.10: Tráfego no *backbone* da rede

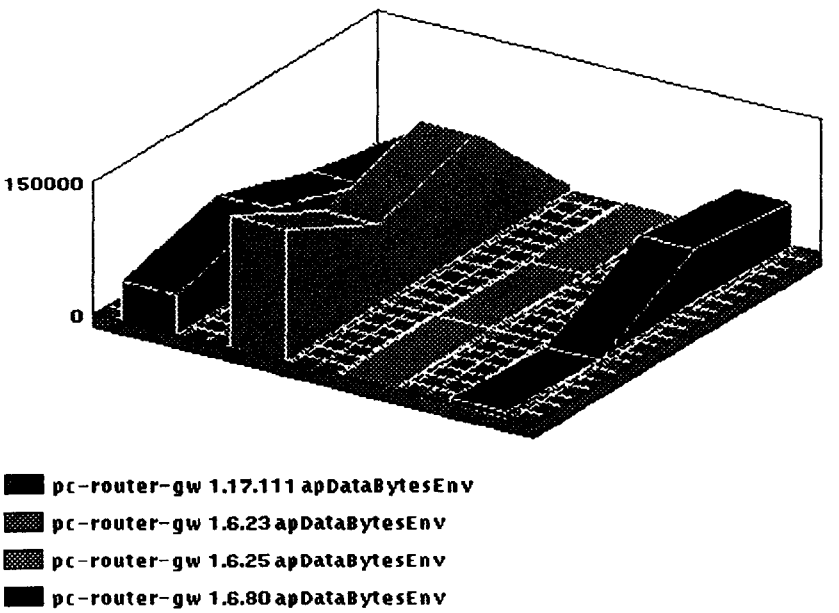

Figura 5.11: Tráfego para fora do departamento

```
Percentual por aplicacao:

1.6.25          1.38%
1.6.80         17.12%
1.17.111       26.60%
1.6.23         54.90%
```

Pelos dois exemplos acima, pode-se notar, com clareza, as diferentes características de tráfego interno e externo do DCC. Para o tráfego dentro do departamento há um predomínio absoluto do RPC sob UDP, que representa a grande maioria do tráfego devido ao NFS, enquanto que para o tráfego para fora do departamento, há um predomínio do *login* remoto.

A outra ferramenta desenvolvida foi denominada de **combine**. Esta ferramenta tem a finalidade de combinar os dados de domínios de um mesmo arquivo de *log* e gerar um outro arquivo de *log* contendo dados combinados. Por exemplo, para se obter os dados relativos à figura 5.11, monitoramos as aplicações citadas acima para os seguintes domínios:

DomIndice	FirstAddr1	LastAddr1	FirstAddr2	LastAddr2	DomStatus
1	143.106.7.0	143.106.7.159	0.0.0.0	143.106.6.255	1
2	143.106.7.0	143.106.7.159	143.106.7.160	255.255.255.255	1
3	143.106.7.0	143.106.7.159	143.106.23.0	143.106.23.31	1
4	143.106.23.0	143.106.23.31	0.0.0.0	143.106.22.255	1
5	143.106.23.0	143.106.23.31	143.106.23.32	255.255.255.255	1
6	143.106.23.0	143.106.23.31	143.106.7.0	143.106.7.159	1

apDomTable

Fazendo-se:

```
combine <arquivo-log1> +1 +2 -3 +4 +5 -6 <arquivo-log2>
```

obtém-se em **arquivo-log2** as informações presentes na figura 5.11.

Uma observação importante é que a ferramenta **combine** somente pode ser utilizada para combinar dados de domínios cujas aplicações monitoradas são iguais.

Através destes dados temos uma informação interessante a respeito do tráfego na rede do DCC-Unicamp, e podemos ter idéia das informações que o esquema

proposto pode oferecer. Pode-se verificar como é possível monitorar o tráfego entre grupos diversos de computadores, para aplicações diferentes, e com intervalos de monitoração adequados a cada caso.

Foi observado, após as monitorações iniciais, uma certa dificuldade em localizar quais as aplicações (caracterizadas pelo código da porta do servidor) utilizavam mais a rede. Para que se pudesse minorar tal problema, foi realizada uma modificação na MIB de forma a monitorar o tráfego, conjuntamente, de uma faixa de códigos de porta. Desta forma, foi possível localizar as faixas responsáveis pelo maior tráfego e, através de aproximações sucessivas, encontrar, de fato, as aplicações que mais carregavam a rede.

Capítulo 6

Conclusão

O gerenciamento de redes vem sendo cada vez mais utilizado e tornou-se uma parte fundamental da administração de redes, em especial, para as grandes e complexas. Neste contexto, o SNMP tem sido bastante usado e já é um padrão de mercado *de facto*.

Como visto, o SNMP fornece uma base de dados de gerenciamento padrão e possibilita a extensão do escopo de gerenciamento através da definição de novos objetos de gerenciamento por empresas ou grupos de pesquisa. Desta forma, à medida que os usuários de computadores solicitam aumentos de capacidade de gerenciamento os fornecedores de tecnologia podem atendê-los.

Dentre as áreas que se vem solicitando um aumento de capacidade de gerenciamento temos o gerenciamento de aplicações. As aplicações são a verdadeira finalidade da rede. As pessoas usam a rede para corresponder-se eletronicamente, navegar através do *mosaic*, *gopher*, entre outras aplicações. Assim, é natural indagar se a rede suporta o tráfego gerado por uma determinada aplicação ou se um dado servidor está, ou não, bem localizado, entre outras duvidas.

Este trabalho tenta contribuir com o gerenciamento de aplicações propondo um esquema para o gerenciamento de tráfego. Por ser o conjunto de protocolos mais difundido, o esquema foi idealizado especificamente para redes TCP-IP.

Foi proposto um esquema de gerenciamento em domínios, onde cada domínio é composto por um par de conjuntos de elementos de rede entre os quais se quer monitorar o tráfego. Propôs-se também uma base de dados de gerenciamento que possibilitou ao gerente de rede a inclusão de domínios e a definição das aplicações, para cada domínio, que se quisesse monitorar. O agente de gerenciamento coletaria todos os pacotes da rede à qual estivesse conectado, podendo, assim, monitorar o tráfego de qualquer domínio para o qual sua rede fosse passagem obrigatória.

Algumas vantagens do esquema de gerenciamento proposto são: seguir o modelo SNMP, que o torna compatível com qualquer aplicativo de gerenciamento que siga tal padrão, a possibilidade de monitorar qualquer aplicação em redes TCP-IP e, de acordo com a localização do agente, a possibilidade de se gerenciar o tráfego para uma variedade bastante grande de configurações de domínios.

O gerenciamento de aplicações permite uma discriminação de tráfego da rede, fornecendo inclusive a noção dos elementos originadores e destinatários de tráfego, fornecendo uma boa base para se fazer uma melhor alocação de servidores na rede, auxiliando também na decisão sobre a formação de sub-redes.

Com o intuito de validar o esquema de gerenciamento, um protótipo foi implementado. O agente de gerenciamento foi desenvolvido em um computador IBM-PC, na linguagem *Borland C*. O aplicativo de gerenciamento utilizado foi o SunNet Manager, onde compilou-se a nova MIB. Obtiveram-se resultados úteis demonstrando a capacidade do esquema.

Ao optar-se por implementar o esquema de gerenciamento de tráfego usando um PC levou-se em conta que é um hardware barato e amplamente difundido. Além disso, foram utilizados softwares de domínio público (*packet driver e waterloo TCP*) existentes para PCs que forneceram a base para a comunicação de rede. Por fim, ressalta-se que a implementação do agente em um PC não interfere com o tráfego da rede e nem no desempenho de qualquer computador que estiver sendo utilizado, deixando a coleta de dados de gerenciamento imperceptível aos usuários (o esquema gera o tráfego de dois datagramas - uma requisição e uma resposta - a cada intervalo de monitoração).

Uma das deficiências sentidas no esquema é o fato de ter de se especificar qual aplicação medir sem ter exata noção das aplicações mais consumidoras de tráfego da rede. Muitas vezes obteve-se um tráfego pequeno devido às aplicações que se especificou e grande devido a aplicações não especificadas (linha correspondente a **outras aplicações** gerada automaticamente para cada domínio na tabela **apDataTable**). Diante disso, uma das sugestões de trabalhos futuros seria o desenvolvimento de um modo de monitoração extra, no qual o tráfego devido a cada aplicação utilizada na rede fosse acumulado separadamente. Depois, usando essas informações, o esquema original poderia ser utilizado para monitorar algumas aplicações de interesse específico.

Uma outra sugestão é tornar o programa que calcula os percentuais de tráfego de aplicações em todo um período (todos os intervalos de medição de uma série) independente do SunNet Manager, já que do modo em que foi proposto o programa analisa um arquivo de *log* gerado pelo SunNet Manager.

Ainda no tocante ao programa que calcula os percentuais, seria interessante o desenvolvimento de uma interface gráfica que tornasse a inspeção dos dados mais

atrativa.

Bibliografia

[Alm92] P. Almquist. Type of service in the internet protocol suite. Internet Request for Comments 1349, julho de 1992.

[Car93] T. C. B. Carvalho, editor. *Gerenciamento de redes - uma abordagem de sistemas abertos.* Makron books, 1993.

[Car94] J. A. Carrilho. Um modelo para o gerenciamento do protocolo ftp baseado em domínios. Master's thesis, DCC-Unicamp, dezembro de 1994.

[CFSD90] J. D. Case, M. Fedor, M. L. Schoffstall, e C. Davin. Simple network management protocol (SNMP). Internet Request for Comments 1157, Maio de 1990. Obsoletes RFC 1098.

[Cic94] R. Cicilini. Desenvolvimento de um agente snmp para plataformas rodando dos. Master's thesis, ICMSC-USP, maio de 1994.

[CM94a] J. A. Carrilho e E. R. M. Madeira. A scheme for ftp management. *Proc. of INET'94/JENC5- The Annual Conference of Internet Society and Joint-European Network Conference*, junho de 1994.

[CM94b] J. A. Carrilho e E. R. M. Madeira. Um esquema para o gerenciamento do protocolo ftp baseado em domínios. *12o. Simpósio Brasileiro de Redes de Computadores*, maio de 1994.

[CM94c] J. Case e K. McCloghrie. Administrative model for version 2 of the simple network management protocol (snmpv2). Internet Request for Comments 1445, abril de 1994.

[CMRW93a] J. Case, K. McCloghrie, M. Rose, e S. Waldbusser. Coexistense between version 1 and version 2 of the internet standard network management framework. Internet Request for Comments 1452, 1993.

[CMRW93b] J. Case, K. McCloghrie, M. Rose, e S. Waldbusser. Introduction to version 2 of the simple network management protocol (snmpv2). Internet Request for Comments 1441, abril de 1993.

[CMRW93c] J. Case, K. McCloghrie, M. Rose, e S. Waldbusser. Management information base for version 2 of the simple network management protocol (snmpv2). Internet Request for Comments 1450, abril de 1993.

[CMRW93d] J. Case, K. McCloghrie, M. Rose, e S. Waldbusser. Manager to manager management information base. Internet Request for Comments 1451, abril de 1993.

[CMRW93e] J. Case, K. McCloghrie, M. Rose, e S. Waldbusser. Protocol operations for version 2 of the simple network management protocol (snmpv2). Internet Request for Comments 1448, abril de 1993.

[Com91] D. E. Comer. *Internetworking with TCP-IP, vol. 1.* Prentice Hall International, segunda ediç ao, 1991.

[Cro91a] R. Crosson. A simple network management protocol function library for ibm-pc compatible computers, 1991. National Institute of Standards and Technology.

[Cro91b] R. Crosson. Snmplib management information base functions, 1991. National Institute of Standards and Technology.

[Eng91] Erick Engelke. *Waterloo TCP.* University of Waterloo, 1991. Disponível por ftp em ftp.unicamp.br.

[Kil94a] S. Kille. Mail monitoring mib. Internet Request for Comments 1566, janeiro de 1994. Editor N. Freed.

[Kil94b] S. Kille. Network services monitoring mib. Internet Request for Comments 1565, janeiro de 1994. Editor N. Freed.

[LR93] D. Lynch e M. Rose, editores. *Internet system handbook.* Addisson-Wesley, 1993.

[MR90a] K. McCloghrie e M. T. Rose. Management information base for network management of TCP/IP-based internets. Internet Request for Comments 1156, Maio de 1990. Obsoletes RFC 1066.

[MR90b] K. McCloghrie e M. T. Rose. Structure and identification of management information for TCP-IP-based internets. Internet Request for Comments 1155, Maio de 1990. Obsoletes RFC 1065.

[MRe91] K. McCloghrie, M. T. Rose, e eds. Management information base for network management of TCP-IP-based internets: MIB-II. Internet Request for Comments 1213, Maio de 1991.

[Pos80] J. Postel. User datagram protocol. Internet Request for Comments 768, agosto de 1980.

[Pos81a] J. B. Postel. Internet protocol. Internet Request for Comments 791, Setembro de 1981. Obsoletes RFC 760.

[Pos81b] J. B. Postel. Transmition control protocol. Internet Request for Comments 793, Setembro de 1981.

[Pos82] J. B. Postel. Simple mail transfer protocol. Internet Request for Comments 821, Agosto de 1982. Obsoletes RFC 788.

[PR83] J. B. Postel e J. K. Reynolds. Telnet protocol specification. Internet Request for Comments 854, Maio de 1983. Obsoletes RFC 764.

[PR85] J. B. Postel e J. K. Reynolds. File transfer protocol. Internet Request for Comments 959, Outubro de 1985. Obsoletes RFC 765.

[Rei91] Peter Reid. A data model for network monitoring and management. *Telecommunications*, 25(8):85–89, agosto de 1991.

[Rom89] John Romkey. *PC/TCP Packet Driver Specification.* FTP Software Inc., 1.09 ediç ao, setembro de 1989.

[Ros91] M. T. Rose. *The simple book - an introduction to management of TCP-IP internets.* Prentice-Hall, 1991.

[RW94] M. A. Rocha e C. B. Westphall. Gerencia de redes de computadores através de novos agentes. *Anais do 12o. Simpósio de Redes de Computadores*, maio de 1994.

[SC87] S. Spanenberg e R. G. A. Cote. Views on a network analyser. *BYTE*, julho de 1987.

[SM] C. K. Silveira e E. M. R. Madeira. Um esquema para o gerenciamento de aplicações em redes tcp-ip. Artigo submetido ao 13o. Simpósio Brasileiro de Redes de Computadores.

[Sta93] W. Stallings. *Local and Metropolitan Area Networks.* MacMillan Publisher Company, quarta ediç ao, 1993.

[Sun93a] Sun Microsystems Inc. *SunNet Manager Reference Guide*, 1993. Vers ao 2.2.

[Sun93b] Sun Microsystems Inc. *SunNet Manager User Guide*, 1993. Vers ao 2.2.

[Tan91] A. S. Tanenbaum. *Computer networks.* Prentice-Hall International, 1991. Segunda Ediç ao.

[TdS94] L. Tarouco e A.C. Benso da Silva. Uma proposta para gerencia de correio eletrônico. *12o. Simpósio Brasileiro de Redes de Computadores*, maio de 1994.

Apêndice A

MIB apTraffic

```
apTraffic-MIB DEFINITIONS ::= BEGIN

IMPORTS
  enterprises, OBJECT-TYPE, NetworkAddress, IpAddress, Counter
  FROM RFC1155-SMI;

apTraffic OBJECT IDENTIFIER ::= {enterprises 45}

apMode OBJECT-TYPE
       SYNTAX Counter
       ACCESS read-write
       STATUS mandatory
       ::= {apTraffic 1}

apPeriod     OBJECT-TYPE
             SYNTAX Counter
             ACCESS read-write
             STATUS mandatory
             ::= {apTraffic 2}

apDomTable   OBJECT-TYPE
             SYNTAX SEQUENCE OF ApDomEntry
             ACCESS read-write
             STATUS mandatory
```

```
              ::= {apTraffic 3}

apDomEntry    OBJECT-TYPE
              SYNTAX ApDomEntry
              ACCESS read-write
              STATUS mandatory
              ::= {apDomTable 1}

ApDomEntry ::= SEQUENCE {
                 apDomIndex
                   Counter,
                 apFirstAddr1
                   IpAddress,
                 apLastAddr1
                   IpAddress,
                 apFirstAddr2
                   IpAddress,
                 apLastAddr2
                   IpAddress
               }

apDomIndex    OBJECT-TYPE
              SYNTAX Counter
              ACCESS read-write
              STATUS mandatory
              ::= {apDomEntry 1}

apFirstAddr1 OBJECT-TYPE
              SYNTAX IpAddress
              ACCESS read-write
              STATUS mandatory
              ::= {apDomEntry 2}

apLastAddr1   OBJECT-TYPE
              SYNTAX IpAddress
              ACCESS read-write
              STATUS mandatory
              ::= {apDomEntry 3}
```

```
apFirstAddr2 OBJECT-TYPE
             SYNTAX IpAddress
             ACCESS read-write
             STATUS mandatory
             ::= {apDomEntry 4}

apLastAddr2  OBJECT-TYPE
             SYNTAX IpAddress
             ACCESS read-write
             STATUS mandatory
             ::= {apDomEntry 5}

apDataTable  OBJECT-TYPE
             SYNTAX SEQUENCE OF ApDataEntry
             ACCESS read-write
             STATUS mandatory
             ::= {apTraffic 4}

apDataEntry  OBJECT-TYPE
             SYNTAX ApDataEntry
             ACCESS read-write
             STATUS mandatory
             ::= {apDataTable 1}


ApDataEntry  ::=
             SEQUENCE {
               apDataIndex
                 Counter,
               apDataProt
                 Counter,
               apDataAplic
                 Counter,
               apDataBytesEnv
                 Counter,
               apDataDatEnv
                 Counter,
               apDataBytesRec
                 Counter,
```

```
                apDataDatRec
                  Counter,
                apDataStatus
                  Counter
               }

apDataIndex    OBJECT-TYPE
               SYNTAX Counter
               ACCESS read-write
               STATUS mandatory
               ::= {apDataEntry 1}

apDataProt     OBJECT-TYPE
               SYNTAX Counter
               ACCESS read-write
               STATUS mandatory
               ::= {apDataEntry 2}

apDataAplic    OBJECT-TYPE
               SYNTAX Counter
               ACCESS read-write
               STATUS mandatory
               ::= {apDataEntry 3}

apDataBytesEnv OBJECT-TYPE
               SYNTAX Counter
               ACCESS read-write
               STATUS mandatory
               ::= {apDataEntry 4}

apDataDatEnv   OBJECT-TYPE
               SYNTAX Counter
               ACCESS read-write
               STATUS mandatory
               ::= {apDataEntry 5}

apDataBytesRec OBJECT-TYPE
               SYNTAX Counter
               ACCESS read-write
```

```
                STATUS mandatory
                ::= {apDataEntry 6}

apDataDatRec    OBJECT-TYPE
                SYNTAX Counter
                ACCESS read-write
                STATUS mandatory
                ::= {apDataEntry 7}

apDataStatus    OBJECT-TYPE
                SYNTAX Counter
                ACCESS read-write
                STATUS mandatory
                ::= {apDataEntry 8}

apFinal         OBJECT-TYPE
                SYNTAX Counter
                ACCESS read-write
                STATUS mandatory
                ::= {apTraffic 5}


END
```

Apêndice B

Projeto do Software Agente

Módulo Principal:

- agente.c

Módulos SNMPLIB [Cro91a]

- fndlenfl.c
- fnddatfl.c
- getobjln.c
- getnxtob.c
- getobjpt.c

Módulos WATTCP [Eng91]

- pcpkt.c
- pcsed.c
- asmpkt.asm
- outch.asm
- outhex.asm
- outs.asm
- qcmp.asm