Um Modelo para o Gerenciamento do Protocolo FTP baseado em Domínios

José Aparecido Carrilho

Um Modelo para o Gerenciamento do Protocolo FTP baseado em Domínios

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. José Aparecido Carrilho e aprovada pela Comissão Julgadora.

Campinas, 22 de dezembro de 1994.

Prof. Dr. Edmundo Roberto Mauro 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.

Um Modelo para o Gerenciamento do Protocolo FTP baseado em Domínios[1]

José Aparecido Carrilho[2]

Departamento de Ciência da Computação
IMECC – UNICAMP

Banca Examinadora:

- Edmundo Roberto Mauro Madeira[3] (Orientador)
- Edson dos Santos Moreira[4]
- Célio Cardoso Guimarães[3]
- Paulo Lício de Geus[3] (Suplente)

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

[2]O autor é Bacharel em Ciência da Computação pela Universidade Estadual de Campinas - UNICAMP.

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

[4]Professor do Departamento de Computação – Instituto de Ciências Matemáticas de São Carlos – USP.

Dedicatória

A Deus pai,
por seu amor, sabedoria e força.
A meus pais,
pela vida, amor e educação.
A meu irmão Airton,
cuja presença jamais será esquecida ...

Agradecimentos

A Deus, pela vida, saúde, família e amigos.

A meus pais, José Carrilho e Ana A. T. Carrilho, que nunca mediram esforços para educar todos seus filhos. Obrigado pelo amor, carinho, dedicação, ...

Ao meu irmão Airton, que zela por nós junto a Deus.

A meus irmãos Ivanildo, Rose, Neide, Lena e Alfeu pelo amor que nos une e pelo apoio durante toda a minha vida.

À minha sobrinha Muriana, fonte de alegria e esperança da família.

Ao Professor Edmundo Madeira, que durante esse período foi muito mais que um orientador, foi um amigo. Obrigado pela compreensão, pelo apoio e pelo carinho com que sempre me recebeu em sua sala. Espero poder contar sempre com sua amizade!

À Professora Ariadne, pelo carinho com que sempre me recebeu em sua sala e pela grande ajuda na versão em Inglês do artigo publicado no INET'94/JENC5.

À minha grande família de Campinas:

- Inês, pela amizade, carinho e incentivo durante todo esse período de convivência.
- Cláudio, grande amigo de todas as horas, pela alegria contagiante de sua presença.
- Fátima, pela amizade, carinho e momentos inesquecíveis.
- Kátia, minha irmãzinha querida, obrigado por tudo.
- Nilza, pessoa formidável que mesmo distância sempre nos transmite muita amor e paz.
- Paulo, amigo inesquecivel que sempre esteve muito presente com seu apoio.
- Carlos Cordeiro, pelos momentos agradáveis do início de nosso mestrado e pelo apoio de sempre.
- Nair e D. Tereza, pela amizade e carinho.

À família Vale da Silva: D. Inês, Seu Tatá, Ricardo e Nice, pelo carinho, fé e incentivo.

Ao Carlos Kelner, Elaine, Humberto, Nuccio, Karen, Pedro Rafael, Otávio e outros companheiros pela resolução de algumas dúvidas sobre a implementação e/ou formatação do texto da tese e pela amizade.

Aos colegas da minha turma de mestrado (Atta, Lincoln, Mário, Adauto, Marcus Vinícius, ...) e das outras turmas (Visoli, Maurício, Fileto, Rober, Bacarin, Tutumi, Guto, Victor, Anderson, Clevan, Daniel, Roseane, ...) pela amizade e momentos agradáveis que

passamos juntos.

Aos funcionários do DCC: Solange, Luiz, Alda, Euclides, Hélio, Monteiro, Emerson e Admilson, pelo carinho com que sempre me atenderam.

Ao Tadao Takahashi e todo o pessoal da RNP, pelo apoio a esse projeto.

Ao CNPq - Conselho Nacional de Pesquisa - e a FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo, pelo apoio financeiro a este projeto.

A todos os amigos que sempre acreditaram no meu trabalho e me incentivaram, meu carinho e gratidão.

Resumo

O gerenciamento de protocolos de aplicação é uma área de pesquisa muito recente e um tipo de serviço ainda não disponível aos usuários, mesmo sendo muito importante para a coleta de informações mais precisas sobre a origem do tráfego nas redes.

O objetivo deste trabalho é definir um modelo para o gerencimento de um protocolo de aplicação e implementá-lo. O procotolo escolhido inicialmente foi o FTP, embora o modelo definido possa ser adaptado para o gerenciamento de outros protocolos de aplicação.

O modelo proposto subdivide o gerenciamento dos protocolos de aplicação em domínios, compostos por uma série de agentes e subagentes, todos sob a coordenação de um ou mais gerentes. As informações do protocolo FTP são armazenadas em uma base de dados de gerenciamento, no grupo *ftp* e são mantidas por um subagente FTP. Um conjunto de serviços de gerenciamento para o protocolo FTP também foi definido.

O gerenciamento do protocolo FTP foi implementado no ambiente ISODE, mas o acesso às informações gerenciadas pode ser obtido a partir de outros sistemas de gerenciamento de rede, como por exemplo o *AIX NetView/6000* e o *SunNet*.

Abstract

The application protocol management is a research area very recent, and a service that is not available yet.

The goal of this work is to define a model for an application protocol management and to implement it. The application protocol selected to manage was the FTP, but this model can be easily modified to manage other application protocols.

The proposed model divides the network management in domains, where a domains is composed of several agents and subagents controled by one or more managers. The FTP management requires some management information stored in the *ftp* object group and maintained by the FTP subagent. Some FTP management services was also defined.

The FTP management was implemented using the ISODE, and can be accessed by other availabe network management systems, like AIX NetView/6000 and SunNet Manager.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

A última década foi marcada pela substituição dos "centros de computação", que continham computadores de grande porte e centralizavam o processamento de dados nas empresas e instituições de pesquisa, por uma série de computadores de menor porte, embora muitas vêzes com maior capacidade de processamento. Esses computadores eram distribuídos em diferentes locais e interligados por linhas de transmissão de dados, formando as redes de computadores [Tan89].

Nas redes de computadores, cada sistema opera isoladamente mas pode se comunicar com os outros sistemas para coletar dados necessários ao seu processamento ou para solicitar serviços remotos.

A necessidade de interligar sistemas cada vez mais distantes levou à evolução das redes locais e a criação de redes metropolitanas e de longa distância.

A exigência dos usuários de melhor qualidade na comunicação, ou seja, da redução nas taxas de erros e de velocidades de transmissão mais elevadas, tem impulsionado o desenvolvimento de novas tecnologias, como por exemplo, as redes FDDI - *Fiber Distributed Data Interface* [Tan89], ATM - *Asynchronous Transfer Mode* [BOU92], entre outras.

O interesse de interligar computadores de diferentes fabricantes e com as características mais diversas gerou a necessidade de padronização dos sistemas de comunicação. Dois conjuntos de protocolos para sistemas abertos, ou melhor, não proprietários, se destacam na área de redes de computadores:

- conjunto de protocolos OSI/ISO - *Open System Interconnection/ International Organization for Standardization*, definido em conjunto pelo CCITT - *Consultive Committe for International Telegraph and Telephone* - e pela ISO/IEC - *International Organization for Standardization / International Eletrotecnical Committe.* Esse conjunto de protocolos foi adotado como padrão internacional (padrão de direito); e

- conjunto de protocolos TCP/IP - *Transmission Control Protocol / Internet Protocol*, definido pelo Departamento de Defesa dos Estados Unidos, e utilizado na *Internet.* A

Internet é a rede com maior quantidade de sistemas inteligados [Kro92], conectando computadores de praticamente todos os países do planeta e, além disso, apresenta o maior índice de crescimento dentre as redes existentes (padrão de fato).

O rápido crescimento das redes de computadores, a interligação das redes locais em redes de longa distância, a exigência dos usuários por redes de melhor qualidade e o interesse de interligar computadores de diversos fabricantes e com diferentes características têm tornado cada vez mais complexa a tarefa de administrar as redes, isto é, o admistrador das redes cada vez tem mais dificuldades para mantê-las em funcionamento [Ros91a].

Na tentativa de simplificar o trabalho dos administradores de redes, em meados da década passada, começaram a ser definidos os primeiros sistemas de gerenciamento de redes.

O principal objetivo dos sistemas de gerenciamento de redes disponíveis tem sido manter o funcionamento das redes sob controle, identificando falhas e tomando as providências necessárias para solucioná-las. Muitas pesquisas estão sendo realizadas no sentido de desenvolver sistemas de gerenciamento inteligentes, que solucionam automaticamente a maioria dos problemas das redes, mas até os dias atuais o trabalho dos administradores das redes é indispensável, embora tenha sido simplificado.

Devido à própria evolução do gerenciamento de redes, os sistemas atuais trabalham principalmente com o gerenciamento dos protocolos das camadas inferiores das redes, preocupados em manter as redes operacionais. Nos últimos anos, tem aumentado o interesse pelo gerenciamento de protocolos de aplicação. A coleta de estatísticas sobre o fluxo de dados gerado por cada protocolo de aplicação fornece informações mais realistas sobre a origem do tráfego nas redes e permite uma melhor alocação de recursos como linhas de transmissão de dados e servidores de arquivo.

O objetivo desse trabalho é a definição de um esquema para o gerenciamento do protocolo FTP - *File Transfer Protocol* [PR85], que possa ser expandido posteriormente para gerenciar também outros protocolos de aplicação da redes *Internet*, como o SMTP - *Simple Mail Transfer Protocolo* [Pos82] - e o Telnet [PR83]. O interesse pelo gerenciamento dos protocolos de aplicação do conjunto de protocolos TCP/IP é justificado pela abrangência das redes *Internet* no mundo atual e pela instalação no país de uma rede acadêmica , a RNP - Rede Nacional de Pesquisa, que suporta o conjunto de protocolos TCP/IP, interliga as principais instituições de ensino e pesquisa do país, e faz parte da *Internet* fornecendo comunicação com o exterior.

A escolha do protocolo FTP, como ponto de partida para o gerenciamento dos protocolos de aplicação, deve-se, principalmente, às estatísticas coletadas na *Internet* que apontam a transferência de arquivos como a aplicação que gera o maior volume de tráfego [DJE92] [KJ94].

O esquema para o gerenciamento do protocolo FTP proposto nesse trabalho envolve:

- o gerenciamento em domínios: permitindo a classificação das transações e a identificação do tráfego em diferentes domínios de gerenciamento. O tráfego pode, por exemplo, ser classificado como local, regional, nacional ou internacional;

- a definição de um grupo de objetos *ftp* para a manutenção das informações de gerenciamento em cada sistema gerenciado;

- a implementação de um subagente FTP encarregado apenas da manutenção dos objetos gerenciados associados ao protocolo FTP, ou melhor, do grupo *ftp*. O subagente FTP deve se associar a um agente para receber as solicitações dos gerentes FTP e enviar as respostas ao mesmo; e

- a implementação de um gerente FTP que ofereça uma série de serviços de gerenciamento do protocolo FTP aos administradores das redes e aos usuários do protocolo FTP em geral.

As pesquisas sobre o gerenciamento de protocolos de aplicação são muito recentes e os primeiros resultados começaram a surgir este ano com a especificação de grupos de objetos e sistemas para o gerenciamento de correio eletrônico [FK94a] [ST94] e diretórios distribuídos [FK94b]. Com relação ao gerenciamento do protocolo FTP, levando em consideração as publicações na área que puderam ser analisadas, os primeiros artigos são resultados deste trabalho [CM94b] [CM94a].

O Capítulo 2 apresenta os conceitos básicos do gerenciamento de redes e a descrição de dois modelos: o Modelo de Gerenciamento de Redes OSI/ISO e o Modelo de Gerenciamento de Redes *Internet*. No final do capítulo, é feita uma comparação entre os dois modelos apresentados.

No Capítulo 3 são descritos e comparados três sistemas de gerenciamento de redes disponíveis no mercado: o Sistema de Gerenciamento de Redes do ISODE, o *AIX NetView/6000* e o *SunNet Manager*. O objetivo desse capítulo é verificar a viabilidade de gerência do protocolo FTP a partir de cada um dos sistemas analisados.

O Capítulo 4 contém a definição do Modelo para o Gerenciamento do Protocolo FTP. Neste capítulo, são apresentados os conceitos de gerenciamento em domínios, a definição do grupo de objetos *ftp*, e a especificação de um subagente FTP e um conjunto de serviços para o gerenciamento do protocolo FTP.

No Capítulo 5 é descrita a implementação de um protótipo para o gerenciamento do protocolo FTP, desenvolvida na plataforma ISODE. Este protótipo foi contruído de acordo com o modelo definido no Capítulo 4 e pode ser utilizado em outros sistemas de gerenciamento de redes disponíveis, como o *AIX NetView/6000* e o *SunNet Manager*.

A conclusão deste trabalho e suas possíveis extensões são apresentadas no Capítulo 6.

Capítulo 2

Gerenciamento de Redes

2.1 Introdução

Gerenciar uma rede significa manter seu funcionamento sob controle, o que envolve entre outras atividades:

- controlar a configuração da rede;
- administrar o acesso aos recursos disponíveis;
- controlar o desempenho da rede, mantendo o tráfego e a utilização de dados em níveis aceitáveis; e
- identificar e solucionar falhas.

O gerenciamento de redes assumiu posição de destaque nas pesquisas em redes de computadores a partir da metade da década de 80. Até então, os administradores das redes controlavam o funcionamento das mesmas através de testes periódicos em diferentes níveis (físico, de conexão, entre outros) com o objetivo de detectar problemas e providenciar a solução dos mesmos.

A administração das redes passou a se tornar bastante complexa a partir do fim da década de 70, principalmente devido:

- ao crescimento acelerado das redes, gerado pela adição de novos equipamentos e pela interligação das redes locais em redes de longa distância;
- às alterações freqüentes nas configurações das redes; e
- ao aumento da heterogeneidade dos equipamentos ligados à rede.

A complexidade do problema deixou bem clara a necessidade de ferramentas automáticas para o gerenciamento de redes. No início da década de 80, alguns fabricantes, como a IBM - *International Busines Machine*, DEC - *Digital Equipments Corporation* e AT&T *Americam Telegraph & Telephone*, lançaram produtos para o gerenciamento de redes baseados em arquiteturas proprietárias [BRI93]. Estes produtos não obtiveram sucesso muito expressivo, pois o mercado necessitava de produtos abertos, isto é, produtos que pudessem gerenciar equipamentos heterogêneos e de diferentes fabricantes.

A partir da metade da década de 80, grupos de pesquisadores se unem em duas frentes de trabalho distintas com objetivos similares: a especificação de plataformas de gerenciamento de redes abertas. Como resultado desses trabalhos são definidos dois modelos de gerenciamento de redes abertos:

- Modelo de Gerenciamento de Redes OSI/ISO, baseado no protocolo CMIP - *Common Management Information Protocol*; e
- Modelo de Gerenciamento de Redes *Internet*, baseado no protocolo SNMP - *Simple Network Management Protocol.*

2.2 Gerenciamento de Redes OSI/ISO

O modelo para o gerenciamento de redes OSI/ISO foi especificado em 1989 por um grupo de pesquisadores do ISO/IEC e do CCITT.

Devido à complexidade e variedade de atividades envolvidas no gerenciamento de uma rede, o modelo OSI/ISO divide e classifica estas atividades em cinco áreas funcionais [Pro87] [BRI93]:

1. Gerenciamento de Falhas: abrange as funções necessárias para investigar a ocorrência de falhas, identificá-las, diagnosticar suas causas e se possível corrigí-las. Os administradores da rede sempre devem ser informados da ocorrência de falhas, mesmo que estas sejam corrigidas automaticamente.

2. Gerenciamento de Configuração: tem como função controlar as condições do ambiente de comunicação do sistema aberto, identificando mudanças significativas e modelando a configuração dos recursos físicos e lógicos da rede.

3. Gerenciamento de Desempenho: oferece as funções para medir, monitorar, avaliar e relatar os níveis de desempenho alcançados pela rede. Essas informações podem ser usadas para fins de planejamento e controle de qualidade dos serviços da rede.

4. Gerenciamento de Segurança: busca garantir a política de segurança definida pela rede e assim controlar o acesso aos recursos disponíveis. Envolve o gerenciamento de

segurança do sistema, o gerenciamento dos serviços de segurança e o gerenciamento dos mecanismos de segurança.

5. Gerenciamento de Contabilização: permite determinar o custo associado à utilização dos recursos da rede, isto é, determinar quais recursos estão sendo utilizados e em que proporção.

Apesar de apresentarem objetivos distintos, as áreas funcionais se relacionam no sentido de que as informações geradas em uma área podem ser utilizadas como suporte para as decisões a serem tomadas nas outras áreas.

A plataforma de gerenciamento OSI/ISO [BRI93] é composta por quatro componentes básicos: os gerentes, os agentes, o protocolo de gerenciamento e os objetos gerenciados, conforme mostra a Figura 2.1.

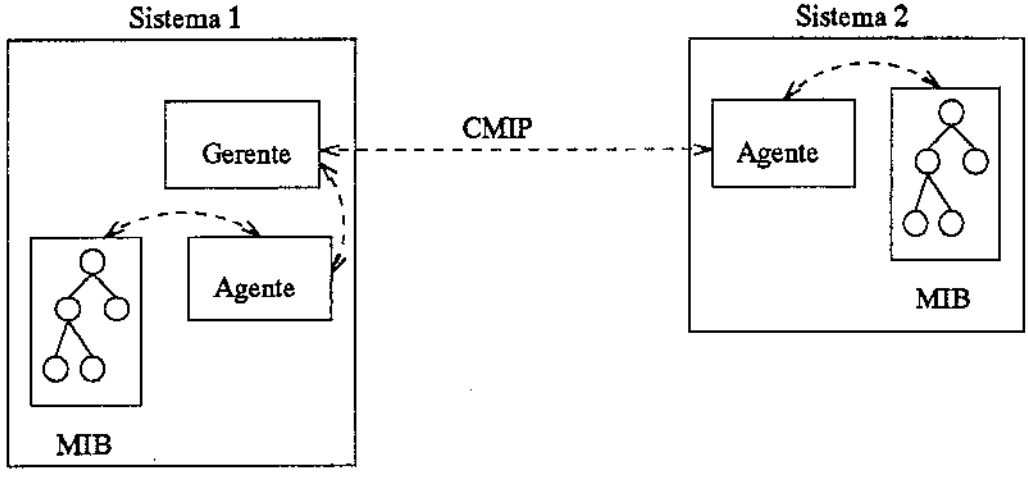


Figura 2.1: Componentes do Gerenciamento de Redes

Os gerentes são encarregados de controlar o funcionamento da rede, transmitindo as operações de gerenciamento aos agentes com o objetivo de coletar informações atualizadas dos objetos gerenciados e manipulá-los.

Os agentes fazem a manutenção dos objetos gerenciados e executam as ações requisitadas pelos gerentes. Além disso, os agentes podem transmitir aos gerentes responsáveis as notificações de falha geradas pelos objetos gerenciados ou as notificações da ocorrência de eventos. Um agente pode ser instalado num sistema onde há um gerente ou em um sistema diferente.

No modelo OSI/ISO foi definido como protocolo de gerenciamento de redes o CMIP, que define as regras de comunicação entre os gerentes e os agentes.

Os objetos gerenciados representam os recursos que estão sujeitos ao gerenciamento. Tais recursos podem ser entidades das camadas, conexões e dispositivos de comunicação, entre outros.

Os objetos gerenciados são definidos em termos de:

- seus atributos ou propriedades;
- operações às quais podem ser submetidos;
- notificações que podem emitir para informar a ocorrência de eventos extraordinários aos gerentes; e
- suas relações com outros objetos.

O conjunto dos objetos gerenciados e seus atributos, ou seja, o conjunto das informações de gerenciamento de um sistema, é organizado hierarquicamente na MIB - *Management Information Base*.

Os componentes da plataforma de gerenciamento de redes OSI/ISO foram especificados para serem executados sobre a pilha de protocolos OSI/ISO, como aplicações.

2.2.1 Arquitetura OSI de Gerenciamento

O modelo de gerenciamento de redes OSI/ISO foi definido com base na pilha de protocolos OSI/ISO [Bla89] [Tan89], formada pelas sete camadas conforme mostra a Figura 2.2.

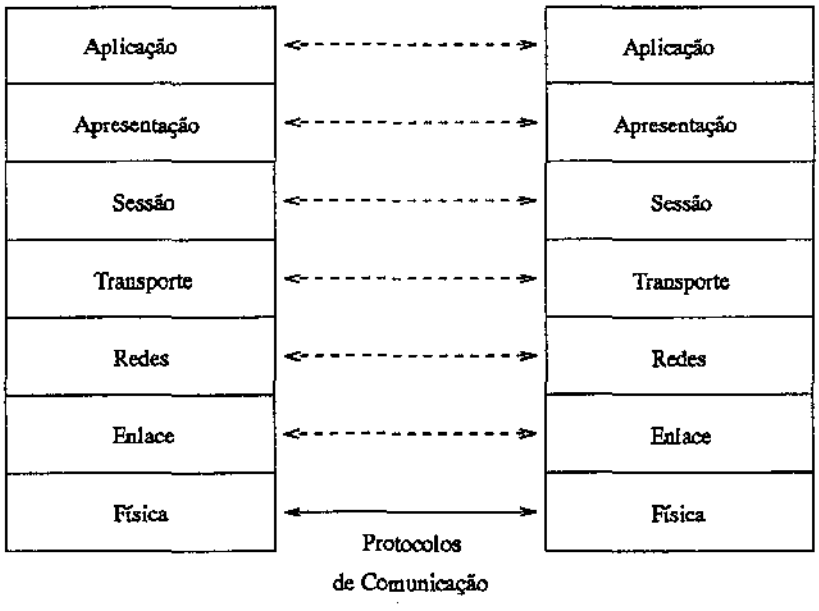


Figura 2.2: Pilha de Protocolos OSI/ISO

Para cada uma das sete camadas do modelo OSI é definida uma LME - *Layer Management Entity* conforme mostra a Figura 2.3, responsáveis por coletar as informações de gerenciamento de cada camada e oferecer os serviços de gerenciamento às mesmas. Cada LME conhece apenas o funcionamento do protocolo específico de sua camada.

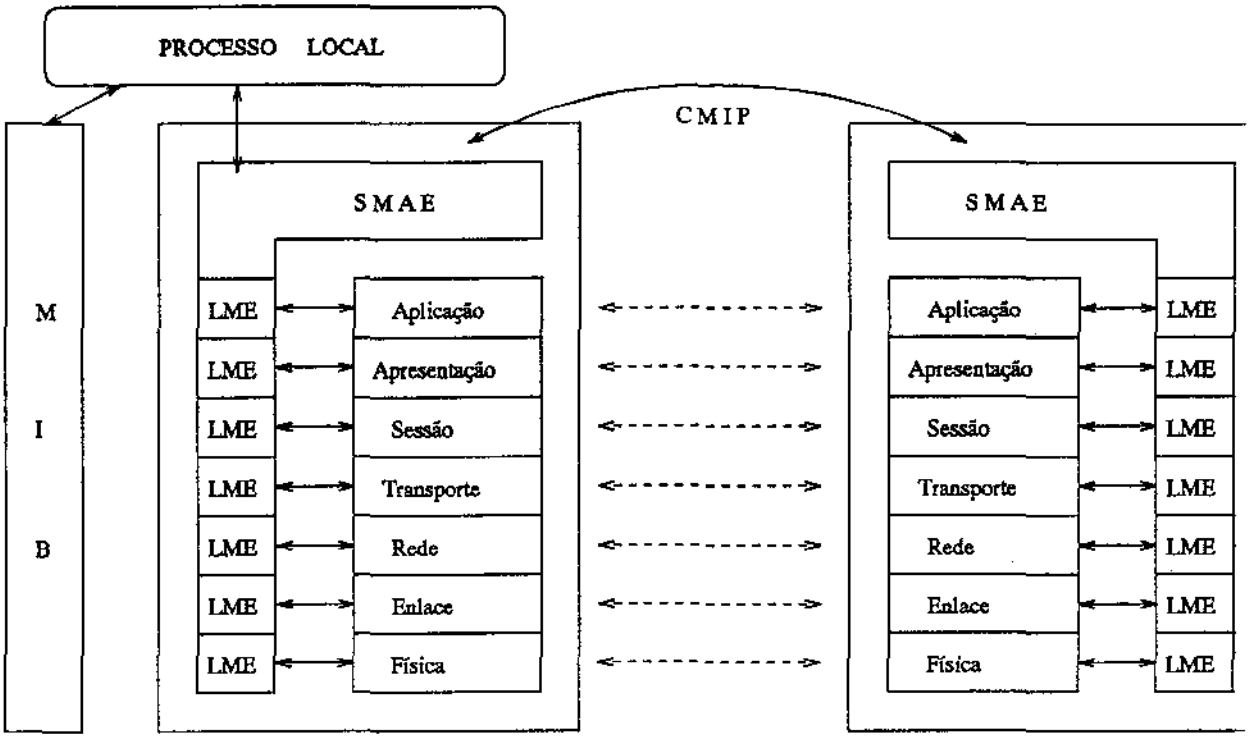


Figura 2.3: Arquitetura de Gerenciamento do Modelo OSI/ISO

A integração das LMEs e a interface com os gerentes e agentes é realizada através da SMAE - *System Management Application Entity*. As LMEs se comunicam com a SMAE através de mecanismos locais. Embora a SMAE tenha acesso à LME de cada camada, a SMAE pertence à camada de aplicação OSI e faz uso de outros serviços da camada de aplicação para comunicar-se com outras SMAEs remotas.

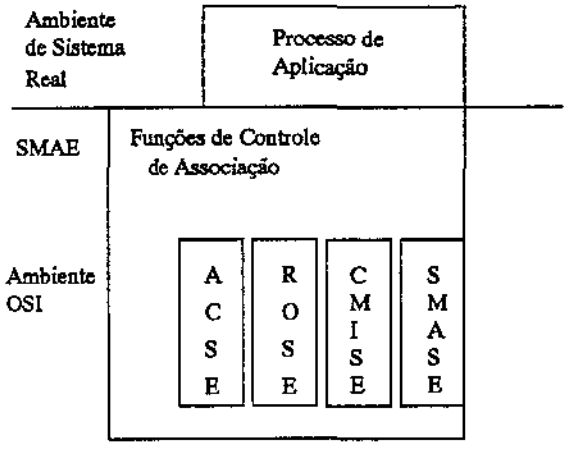
A comunicação entre as SMAEs local e remota é realizada através do protocolo CMIP, com o objetivo de trocar as informações de gerenciamento entre os nós da rede, ou seja, entre gerentes e agentes.

A Entidade que solicita a associação é denominada SMAE-iniciadora e a que recebe o pedido da associação é a SMAE-respondedora. Normalmente a SMAE-iniciadora é um gerente que está solicitando serviços a um agente, que é a SMAE-respondedora. A associação também pode ser solicitada pelo agente a fim de reportar a ocorrência de eventos extraordinários ao gerente. Neste caso os papéis são invertidos, o agente passa a ser a SMAE-iniciadora e o gerente a SMAE-respondedora.

A SMAE é composta pelos seguintes ASEs - *Application Service Elements*, como pode ser visto na Figura 2.4:

- SMASE - *System Management Application Service Element*,
- ACSE - *Association Control Service Element*,

- CMISE - *Common Management Information Service Element,*
- ROSE - *Remote Operation Service Element.*



Legenda:

ACSE - Association Control Service Element
CMISE - Common Management Information Service Element
ROSE - Remote Operations Service Element
SMAE - System Management Application Element
SMASE - System Management Application Service Element

Figura 2.4: Camada de Aplicação para o Gerenciamento de Redes

O SMASE define a semântica e sintaxe abstrata das MAPDUs - *Management Application Protocol Data Units.* As associações entre SMAEs remotas são estabelecidas e liberadas através do uso dos servicos do ACSE. Os serviços de comunicação usados pelo SMASE são prestados pelo CMISE. O uso do CMISE implica na presença do ROSE.

O CMISE define o serviço e os procedimentos usados para transferência das CMIPDUs - *Common Management Information Protocol Data Units* e provê um meio de troca de informações de gerenciamento.

2.2.2 CMIP

O protocolo de gerenciamento adotado no modelo OSI é o CMIP [ISOa]. O CMIP define as normas de comunicação entre os gerentes e agentes do modelo OSI.

O protocolo CMIP opera a nível de aplicação e faz uso de toda a pilha de protocolos OSI para a transferência de suas mensagens. Geralmente a nível de transporte o CMIP utiliza-se de um protocolo orientado a conexão.

O CMIP é um protocolo orientado a conexão, portanto antes da troca de PDUs - *Protocol Data Units* de gerenciamento é necessário o estabelecimento de uma associação entre as duas camadas de aplicação das entidades envolvidas.

As primitivas que compõem o protocolo CMIP são as seguintes:

- *M-Create*: cria uma instância de um objeto gerenciado.
- *M-Delete*: remove uma instância de um objeto gerenciado.
- *M-Get*: lê valores dos atributos de objetos gerenciados.
- *M-Cancel-Get*: cancela a execução de um pedido de *M-Get*.
- *M-Set*: atribui valores a atributos dos objetos gerenciados.
- *M-Action*: executa uma ação sobre um objeto gerenciado.
- *M-Event-Report*: aviso da ocorrência de algum evento anormal em algum objeto gerenciado.

Todas as primitivas do protocolo CMIP são confirmadas. Normalmente o gerente envia a primitiva na forma de um *request* e o agente responde com um *response*, com exceção da primitiva de *M-Event-Report* onde o agente envia o *request* e o gerente confirma seu recebimento com o *response*.

Uma única mensagem CMIP pode especificar a aplicação de uma operação de gerenciamento em vários objetos, conforme descrito na seção a seguir. A execução dessa operação de gerenciamento poderá ser realizada de forma atômica ou não, de acordo com o uso do processo de sincronização. Se a operação for realizada de forma atômica, esta deverá se completar com sucesso para todas as variáveis selecionadas da MIB ou não se completará para nenhuma.

2.2.3 MIB

Os conceitos básicos do modelo de informação [ISOb] usado para o gerenciamento OSI são definidos através da SMI - *Structure of Management Information*.

A SMI estabelece as regras para a definição dos objetos gerenciados da MIB como, por exemplo, os tipos de objetos possíveis, as operações que podem ser realizadas em cada um, o comportamento dos mesmos quando uma operação é efetuada, entre outras. Além disso, a SMI configura o cenário no qual a MIB será definida, ou seja, determina onde os objetos serão armazenados, em uma base de dados ou em arquivos, e o esquema para o armazenamento desses objetos.

A definição dos objetos gerenciados que irão compor a MIB seguindo as normas SMI é realizada através de um subconjunto da notação ASN.1 - *Abstract Syntax Notation-One* [Ros90].

A MIB é definida como um conjunto de objetos gerenciados em um sistema aberto, no qual cada objeto gerenciado é uma visão abstrata de um recurso real do sistema, conforme mostra a Figura 2.5.

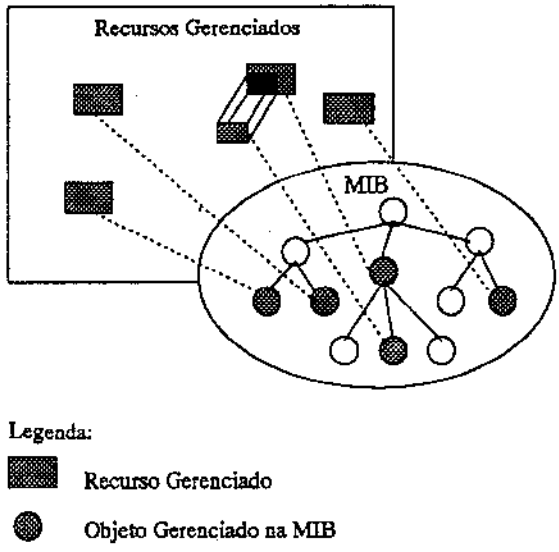


Figura 2.5: Mapeamento de Objetos do Mundo Real em Objetos da MIB

Os objetos da MIB devem refletir o estado dos recursos do mundo real, através de seus atributros. Acessando esses objetos, os gerentes podem identificar problemas, controlar o funcionamento dos recursos, enfim, gerenciar a rede.

No modelo OSI/ISO, os objetos que devem ser gerenciados em uma rede não foram padronizados, isto é, a definição da MIB depende das necessidades de cada administração. Entretanto, há alguns objetos básicos que compõem as MIBs dos sistemas em geral, entre outros pode se destacar objetos para o controle do funcionamento e do fluxo de dados de cada camada OSI, do funcionamento do sistema como um todo e da configuração da rede.

Hierarquias de Gerenciamento

Os objetos gerenciados podem ser organizados na MIB em três tipos de hierarquias: de herança, de nomeação e de registro.

Na Hierarquia de Herança ou de Classe, mostrada na Figura 2.6, os objetos são agrupados em superclasses por possuírem atributos, comportamento, pacotes condicionais, operações e notificações comuns. Nesse tipo de hierarquia podem existir também subclasses que herdam as propriedades das superclasses de maneira irrestrita e possuem algumas

propriedades particulares. Uma subclasse pode ser derivada de mais de uma superclasse, o que é definido como herança múltipla.

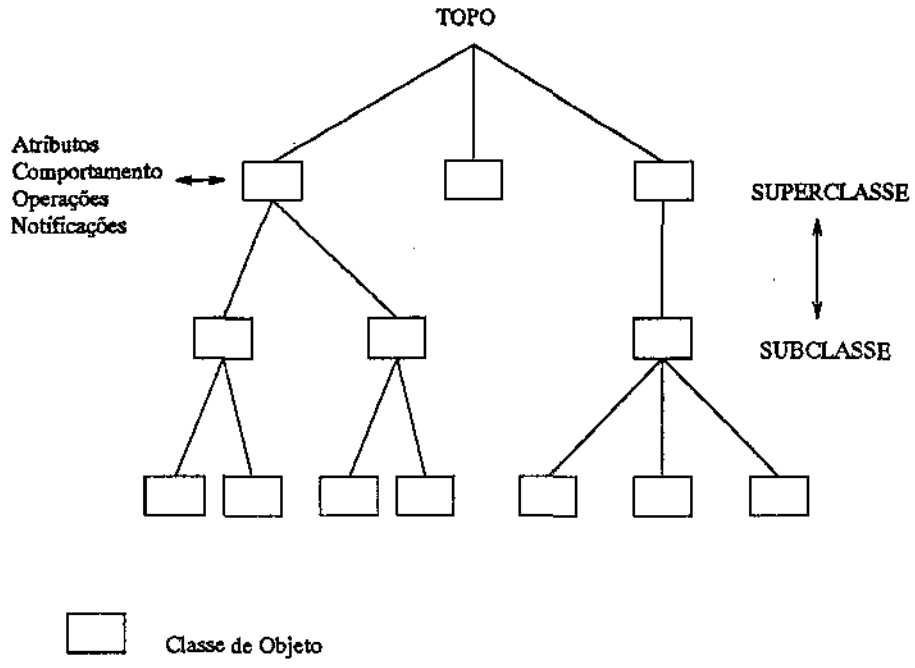


Figura 2.6: Árvore de Classes de Objetos Gerenciados

A Hierarquia de Nomeação relaciona as instâncias dos objetos com seus respectivos nomes. Nessa hieraquia o nível mais alto é chamado de raiz e é um objeto nulo. Os subordinados são identificados por nomes únicos, RDNs - *Relative Distinguished Names*. Os RDNs são formados pela combinação de um atributo e um valor, e devem ser únicos para cada uma das instâncias de objetos que possuam o mesmo superior na hierarquia. O nome completo de uma instância, DN - *Distinguished Name*, consiste de uma sequência de RDNs começando pela raiz e incluindo o RDN da própria instância, conforme mostra a Figura 2.7.

A Hierarquia de Nomeação permite modelar hierarquias do mundo real na MIB, por exemplo, módulos, submódulos e componentes eletrônicos ou modelar hierarquias organizacionais como diretórios, arquivos, registros e campos.

A Hierarquia de Registros é usada para identificar de forma universal os objetos, independente das outras hierarquias. Esta hierarquia é especificada segundo as regras estabelecidas pela notação ASN.1 para a árvore de registros usada na atribuição de identificadores aos objetos. Cada nó da árvore está associado a uma autoridade de registro, por exemplo à ISO, que determina como são atribuídos os seus números. Dessa forma, cada objeto é identificado por uma sequência de números, onde cada número corresponde a um nó da árvore, conforme mostra o exemplo na figura 2.8.

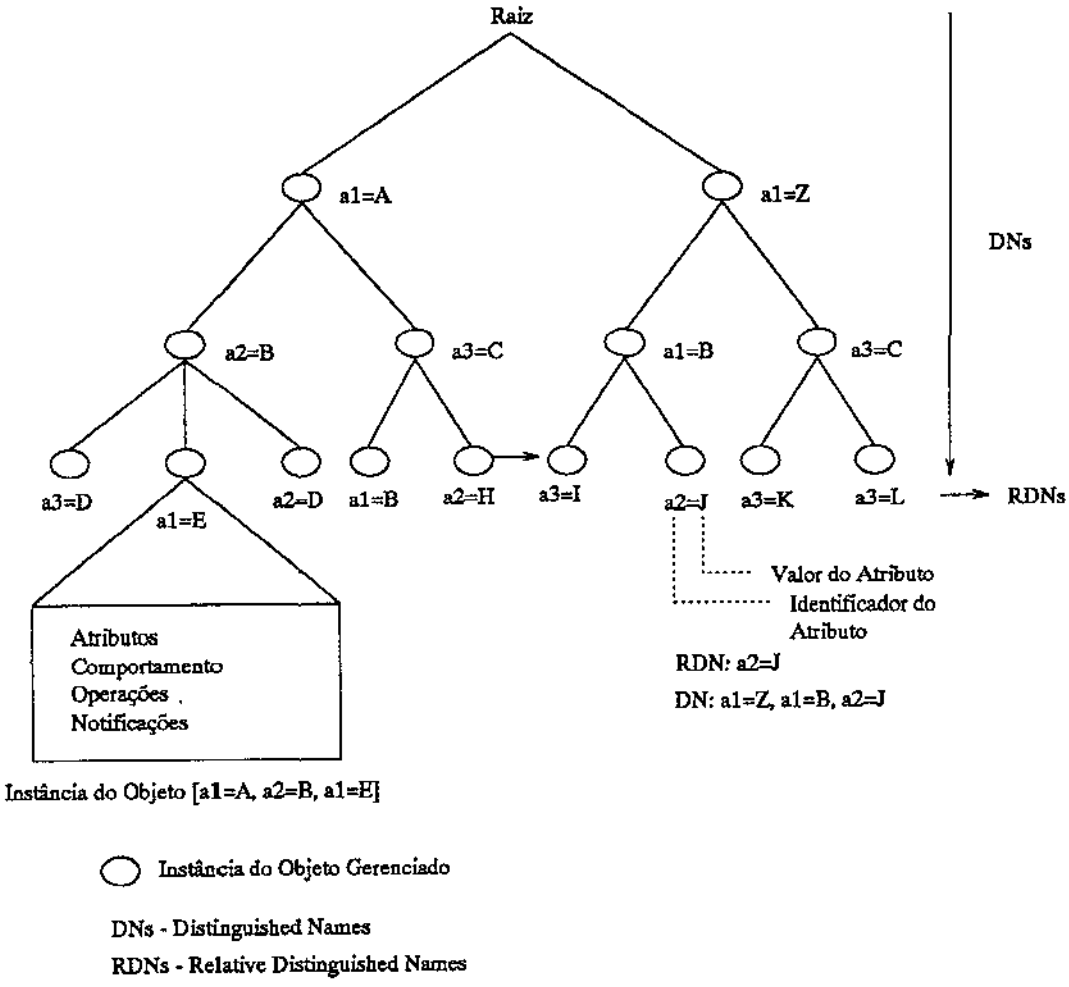


Figura 2.7: Exemplo de Hierarquia de Nomeação

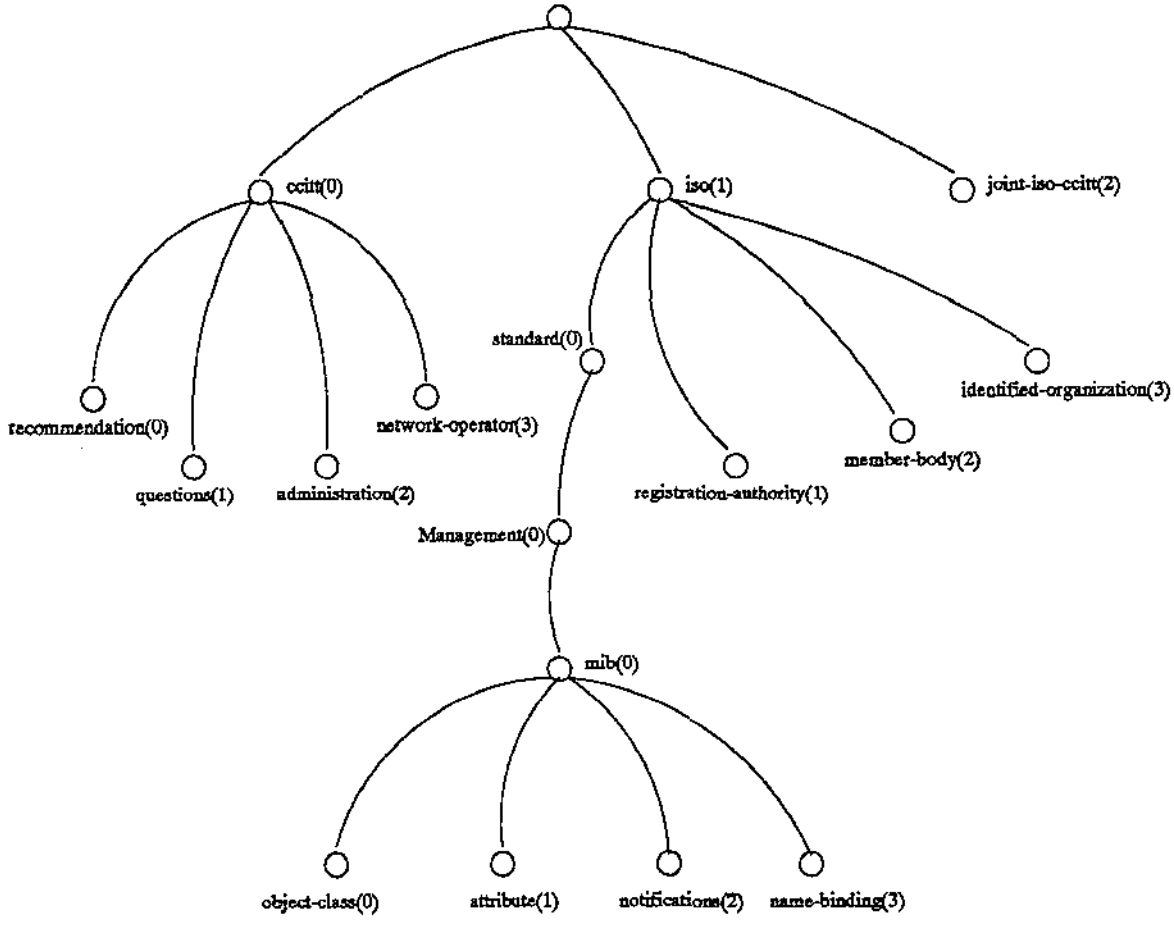


Figura 2.8: Exemplo de Hierarquia de Registro

Identificação dos Objetos

A identificação dos objetos gerenciados de interesse para a realização de uma operação de gerenciamento pode ser feita em uma das seguintes formas:

1. os objetos gerenciados de interesse podem ser especificados um a um seguindo a hierarquia de nomeação ou de registro na solicitação de uma operação, como no exemplo mostrado na Figura 2.9;

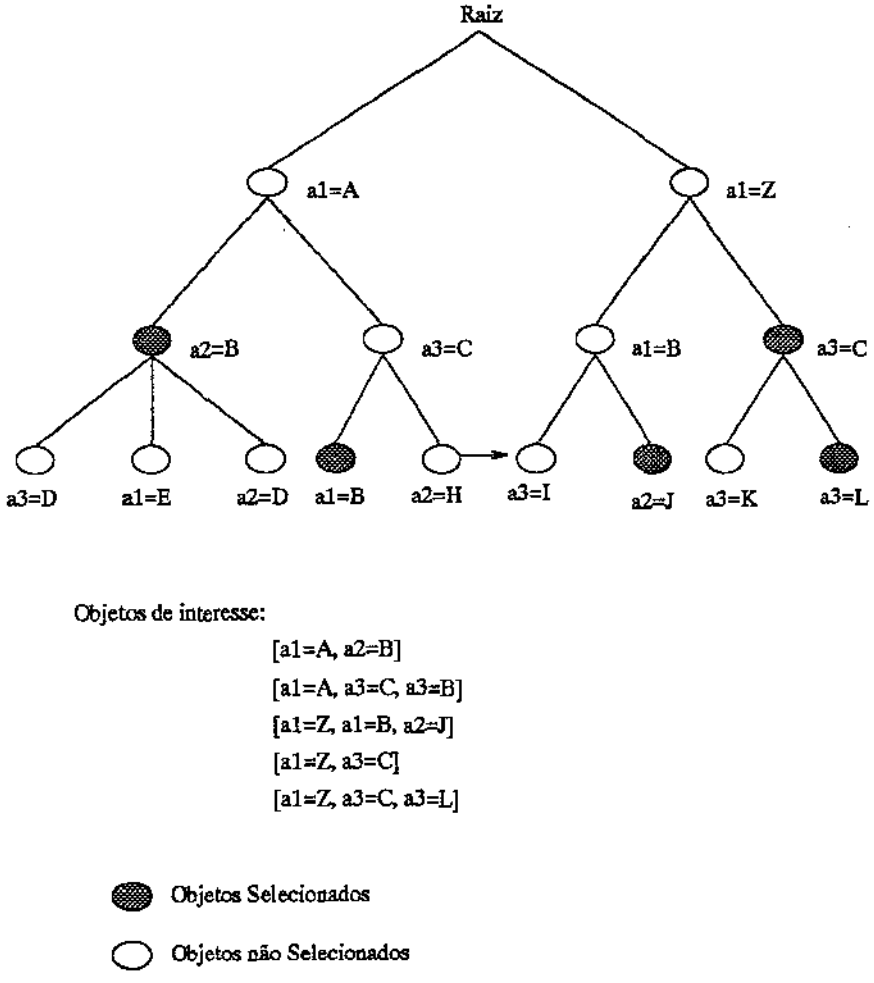


Figura 2.9: Exemplo de Identificação de Objetos Um a Um

2. *scoping*: um nó intermediário da árvore é especificado e a operação de gerenciamento solicitada é aplicada em todos os objetos da subárvore com raiz neste nó, como pode ser visto na Figura 2.10; e

3. filtros: alguns agentes permitem definir critérios que os objetos gerenciados devem satisfazer para que a operação de gerenciamento seja executada. O filtro normalmente é utilizado em conjunto com o *scoping*, aplicando-se uma expressão condi-

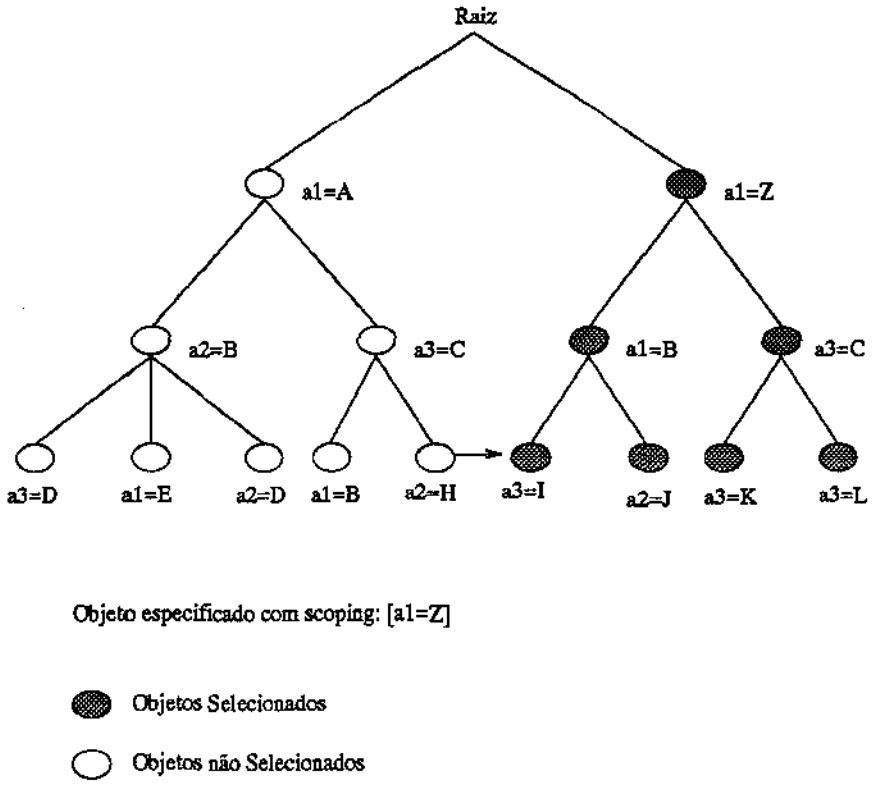


Figura 2.10: Exemplo de Identificação de Objetos por *Scoping*

cional arbitrária nos atributos dos objetos da subárvore definida no *scoping* e a operação de gerenciamento é realizada nos objetos que satisfazem a expressão em questão, como mostra a Figura 2.11.

Proteção das Informações de Gerenciamento

A possibilidade de uma rede ser controlada por mais de um gerente, gerou a necessidade da ISO definir algumas normas de controle de acesso que protejam a MIB contra acessos não autorizados. Dessa maneira, a ISO definiu mecanismos de autenticação com vários níveis de controle de acesso ou de autorização. Seguindo essa filosofia, alguns gerentes podem ter acesso apenas a parte dos objetos gerenciados enquanto outros podem acessar diferentes conjuntos de objetos. Além disso, os usuários podem ter permissões distintas para acessar os objetos gerenciados, ou seja, alguns usuários podem acessar um objeto para leitura e escrita de seus atributos, outros podem ter apenas permissão para ler os valores dos atributos desse objeto, e outros podem não ter permissão sequer para ler esses valores.

Foram também definidos controles para o envio de notificações de falhas somente para determinados gerentes.

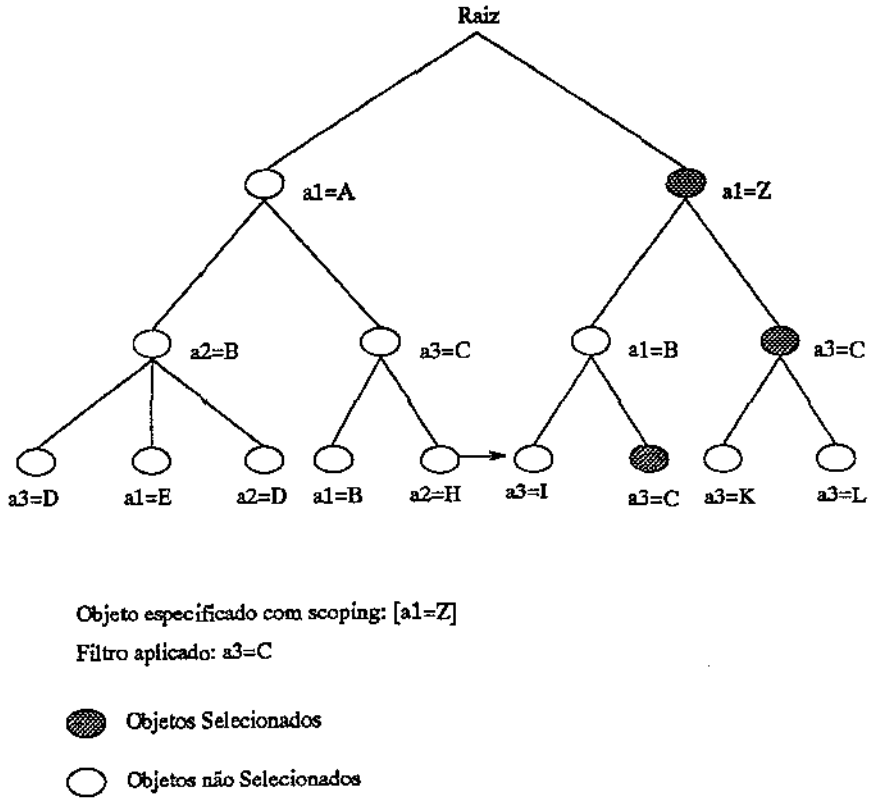


Figura 2.11: Exemplo de Identificação de Objetos usando Filtros

2.3 Gerenciamento de Redes *Internet*

O gerenciamento de redes *Internet* tem por objetivo possibilitar aos administradores da rede analisar os problemas das mesmas, controlar as tabelas de roteamentos, localizar os computadores que violam os padrões dos protocolos, contabilizar o uso de recursos, enfim, manter sob controle todo o funcionamento das redes.

Assim como no modelo OSI, os componentes envolvidos no processo de gerenciamento de redes *Internet* são: gerentes, agentes, objetos gerenciados e o protocolo de gerenciamento.

Os gerentes têm a função de controlar o funcionamento da rede e para isso se comunicam com os agentes via o protocolo de gerenciamento e acessam as informações de gerenciamento para consultá-las ou alterar seus valores.

Os agentes são encarregados de manter as informações de gerenciamento atualizadas e de responder as solicitações emitidas pelos gerentes.

Os objetos gerenciados, assim como no modelo OSI, representam os recursos do mundo real que estão sujeitos ao gerenciamento, como por exemplo protocolos de comunicação e tabelas de roteamento. As informações de gerenciamento são armazenadas em uma MIB mantendo uma estrutura hierárquica.

O protocolo de gerenciamento de redes define as normas de comunicação entre os gerentes e agentes, possibilitando a troca das informações de gerenciamento entre entidades remotas. No modelo Internet é adotado o protocolo SNMP [CFSD90].

2.3.1 Modelo de Gerenciamento *Internet*

O modelo *Internet* de gerenciamento de redes, apresentado na Figura 2.12, é baseado no paradigma Cliente/Servidor [Com91]. Cada *host* ou *gateway* sendo gerenciado mantém um programa servidor, que é o agente de gerenciamento, e sua MIB com as informações necessárias. O agente permanece sempre a espera de uma comunicação do gerente.

Um gerente é implementado como um *software* cliente e é ativado em um *host* local. Ao se ativar o gerente, é especificado o agente com o qual deseja estabelecer uma comunicação. O gerente, então, se encarrega de estabelecer um contato com o agente e a partir desse momento se dá inicio ao processo de gerenciamento, onde o gerente envia comandos para recuperar as informações da MIB ou alterar as condições de funcionamento do *gateway* ou host via seus objetos de gerenciamento.

Sempre é o gerente quem inicia a comunicação com o agente e envia-lhe os comandos. A função do agente é prestar serviços ao gerente executando os comandos enviados por este e enviando-lhe as respostas, através do protocolo SNMP.

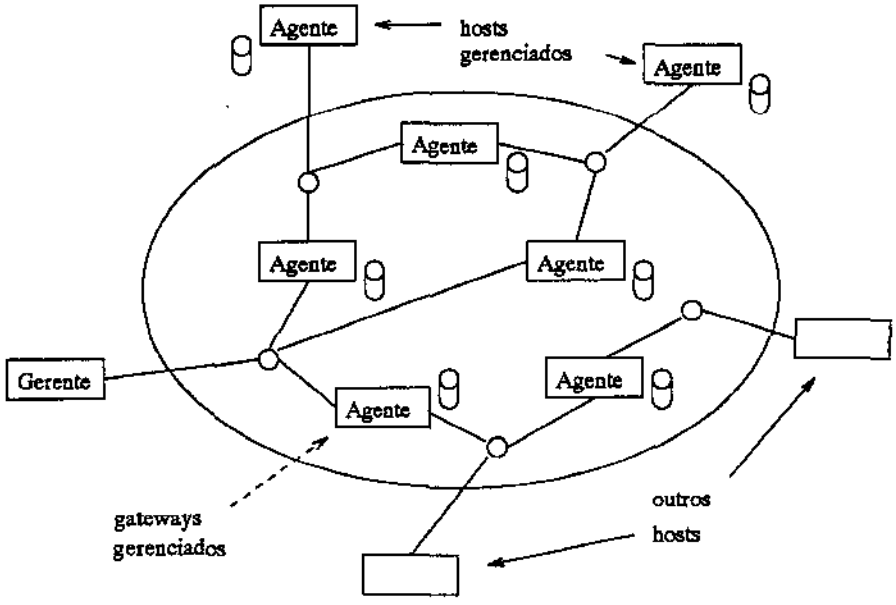


Figura 2.12: Modelo Internet para Gerenciamento de Redes

2.3.2 SNMP

O SNMP [Ros91a] define as normas de comunicação entre o gerente e o agente, especifica a forma e o significado das mensagens de gerenciamento que circulam na rede e a representação de nomes e valores nessas mensagens. Além disso, através do Protocolo SNMP, pode-se definir a relação administrativa entre os *host* e *gateways*, via os mecanismos de autenticação que este provê.

O SNMP opera a nível de aplicação, mantendo assim uma certa independência com relação ao *hardware* e aos *softwares* básicos, ou seja, o SNMP pode ser usado para gerenciar os diferentes tipos de equipamentos da rede. Dessa maneira, o gerente pode controlar todos os *hosts* e *gateways* da rede de forma uniforme.

A única desvantagem do protocolo SNMP ser do nível de aplicação é sua dependência com relação ao funcionamento dos protocolos de transporte e rede e do sistema operacional. Se uma tabela de roteamento se deteriorar, por exemplo, é impossível corrigí-la ou reinicializar o sistema de uma máquina remota. Da mesma maneira, se ocorrer alguma falha no sistema operacional de uma máquina, não é possível alcançar o programa de aplicação que implementa o gerenciamento de redes.

O protocolo SNMP é dito como independente do protocolo da camada de transporte, isto porque hoje existem vários mapeamentos do protocolo SNMP para diferentes protocolos do nível de transporte. Originalmente o protocolo SNMP foi definido para usar os protocolos TCP ou UDP - *User Datagram Protocol* para o transporte de suas mensagens. Hoje pode-se encontrar mapeamentos do SNMP diretamente sobre a *Ethernet* [SDFC89] e sobre os serviços de transporte do modelo OSI orientados ou não a conexão [Ros93].

Todos os mapeamentos do protocolo SNMP para o nível de transporte têm em comum a transmissão das mensagens SNMP através de um processo de serialização. Dessa forma qualquer estrutura de dados é codificada como uma seqüência de octetos e enviada pela rede, no destino a estrutura é decodificada readquirindo uma forma idêntica à original. As estruturas de dados são traduzidas de ASN.1, seu formato padrão, para uma sequência de bytes em um mapeamento de um para um.

Embora sejam definidos vários mapeamentos a nível de transporte, o protocolo mais utilizado é o UDP, principalmente por ser este um protocolo da família dos protocolos *Internet* e por não ser um protocolo orientado a conexão. A preferência por um protocolo não orientado a conexão se deve à necessidade de reduzir o impacto da inserção do sistema de gerenciamento em redes que já estão em funcionamento. Desse modo a pilha dos protocolos *Internet* adquire a estrutura mostrada na Figura 2.13:

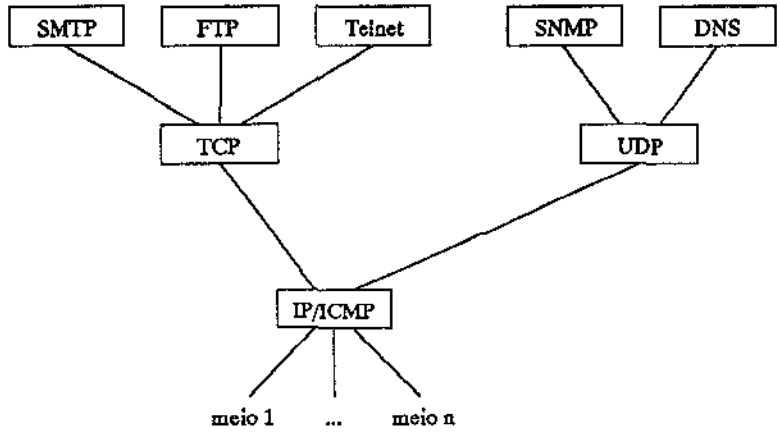


Figura 2.13: Protocolos Internet

O SNMP se baseia no paradigma de *fetch-and-store* [Ros91a], isto é, possui dois comandos básicos: um para buscar os valores de itens e outro para armazenar valores de itens. As outras operações são definidas como *side-effects*, ou seja, todos os objetos do mundo real que podem ser gerenciados e as operações ou ações que podem ser realizadas são mapeadas em objetos da MIB. Quando o gerente deseja que o agente execute uma ação ele deve alterar o valor do objeto da MIB do agente que corresponda a ação desejada de acordo com regras preestabelecidas. O agente identifica o que foi alterado e executa a ação predefinida. Um exemplo clássico é a reinicialização de uma máquina, onde é mantido um objeto na MIB que representa o tempo para a próxima reinicialização e quando o gerente desejar que a máquina seja reinicializada ele deve alterar o valor desse objeto para zero. O agente, ao identificar que o tempo para a próxima reinicialização tem o valor zero, aciona o processo de reinicialização da máquina.

As vantagens de se ter um protocolo de gerenciamento baseado no paradigma de *fetch-and-store* são: estabilidade, simplicidade e flexibilidade.

O SNMP é estável no sentido que permanece fixo e permite a inserção de novos objetos na MIB, podendo dessa forma serem inseridos novos comandos no sistema como *side-effects* da alteração dos valores dos novos obejtos da MIB, sem a necessidade de modificação do protocolo.

O SNMP é simples de implementar, entender e usar porque não há a complexidade de casos especiais para cada comando SNMP, e é flexível porque permite acomodar novos comandos de forma elegante.

Do ponto de vista do usuário, a não existência de comandos imperativos pode ser invisível, sendo necessário apenas uma interface que mapeie os comandos dos usuários para as primitivas SNMP adequadas.

Portanto, o conjunto de primitivas do protocolo SNMP é bem reduzido:

- *get-request*: busca o valor de uma ou mais variáveis.
- *get-next-request*: busca o valor da primeira variável a partir da última consultada ou da variável especificada, sem saber seu nome exato.
- *set-request*: altera o valor de uma ou mais variáveis.
- *get-response*: informa o resultado de uma operação de busca ou de alteração .
- *trap*: reporta ocorrência de um evento.

As primitivas de *get-request* e *set-request* provêem as operações básicas de *fetch-and-store*, recuperando e alterando os valores das variáveis.

A primitiva de *get-next-request* é usada para localizar a instância de uma variável sem a necessidade de especificá-la. Esta primitiva é bastante útil, pois permite ao gerente identificar os objetos que compõem a MIB de um agente sem conhecimentos prévios.

A primitiva *get-response* é usada pelo agente para responder aos pedidos de consulta e alteração de suas variáveis emitidos pelos gerentes.

A primitiva *trap* é enviada pelo agente para informar ao gerente responsável a ocorrência de algum evento extraordinário. É função do gerente programar os agentes para o envio de *traps*, sendo também sua função definir quais os eventos que serão informados por cada agente. Quando um gerente recebe um *trap*, é sua função decidir se deve conectar-se com o agente que emitiu o aviso no mesmo instante ou se espera que chegue a hora de conectá-lo pelo mecanismo de *polling*.

Uma única mensagem SNMP pode especificar uma operação para múltiplas variáveis. O agente executa a operação de forma atômica, isto é, ou a operação se completa para todas as variáveis especificadas ou não é efetivada para nenhuma. Por exemplo, quando o gerente envia um *set-request* para alterar o valor de três variáveis, se os novos valores podem ser atribuídos por este gerente às três variáveis, a operação se completa com sucesso, mas caso o gerente não tenha permissão para alterar o valor de uma das variáveis, a operação não é processada para nenhuma das três.

2.3.3 MIB

Os objetos gerenciados são organizados hierarquicamente na MIB [MR91b].

Como no modelo OSI, as leis para definição dos objetos gerenciados são estabelecidas pela SMI [RM90]. A SMI determina os tipos de objetos possíveis, as formas de acesso (leitura e escrita, somente leitura ou não acessível) e o seu estado (obrigatório, opcional ou obsoleto).

A SMI estabelece o uso de um subconjunto da notação ASN.1 para definição dos objetos da MIB, como no modelo OSI/ISO.

Os objetos que compõem a MIB são identificados por uma cadeia de nomes ou *Object Identifiers* que determinam o caminho a ser seguido na árvore para localizar um objeto. O *Object Identifier* é uma sequência de números naturais, onde cada número representa uma coordenada na árvore.

O primeiro nível da árvore contém os seguintes grupos:

- ccitt (0): administrado pelo CCITT.
- iso (1): administrado pela ISO/IEC.
- joint-iso-ccitt (2): administrado em conjunto pelas duas instituições, ISO/IEC e CCITT.

A *Internet* possui uma subárvore sob sua administração, com o seguinte prefixo: iso(1) org(3) dod(6), como pode ser observado na Figura 2.14.

A árvore da *Internet* é dividida em 4 subárvores: *directory, mgmt, experimental* e *private*, onde as três últimas são de particular interesse para o gerenciamento de redes. A Figura 2.15 mostra a subarvore Internet.

A subárvore *mgmt* contém a definição da MIB. A subárvore *experimental* mantém os dados sobre experiências de gerenciamento cadastradas pela *Internet* antes da padronização das mesmas. A subárvore *private* contém um único grupo chamado *enterprises* onde são registrados objetos privados e de vendedores específicos.

A *Internet* definiu sua primeira MIB em meados de 1988 [MR88], contendo os seguintes grupos de objetos:

- *system*: grupo que mantém informações genéricas sobre a entidade sendo gerenciada;
- *interfaces*: grupo que mantém informações de cada interface por onde o sistema pode enviar ou receber datagramas IP;
- *address translation*: grupo composto por uma tabela de tradução de endereços de rede para endereços físicos;

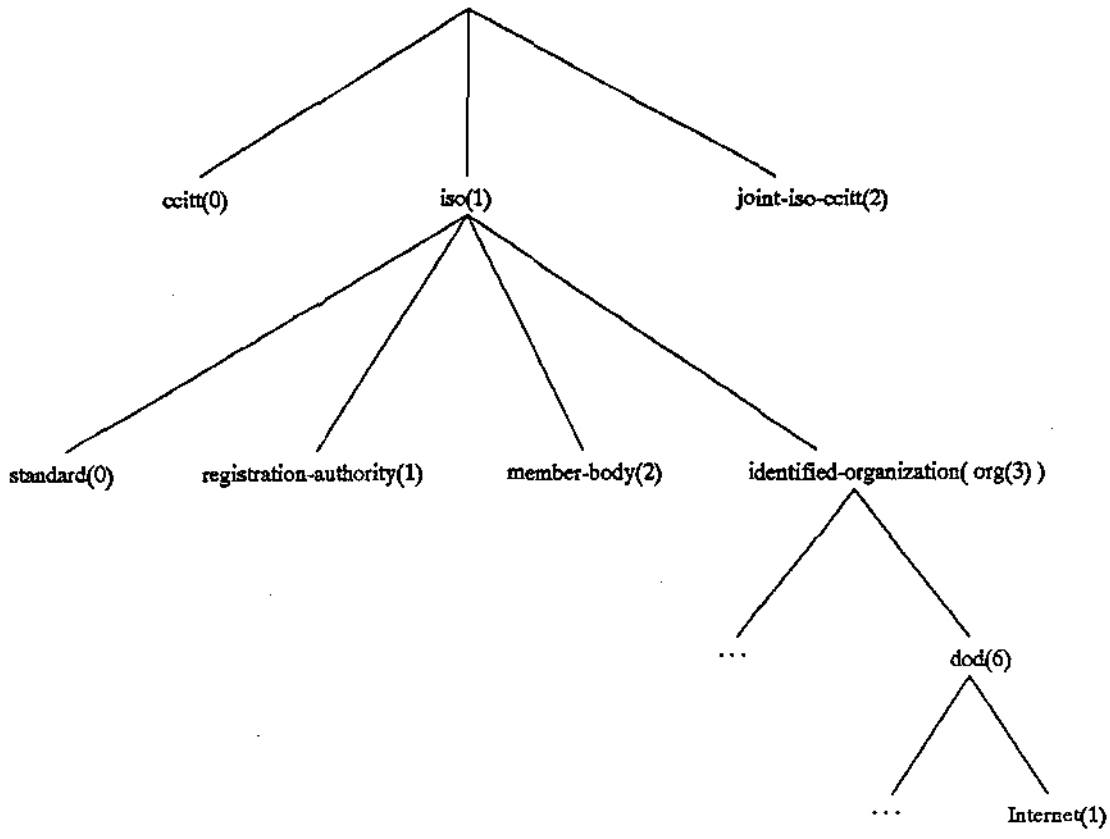


Figura 2.14: Cadeia de Nomes para Internet

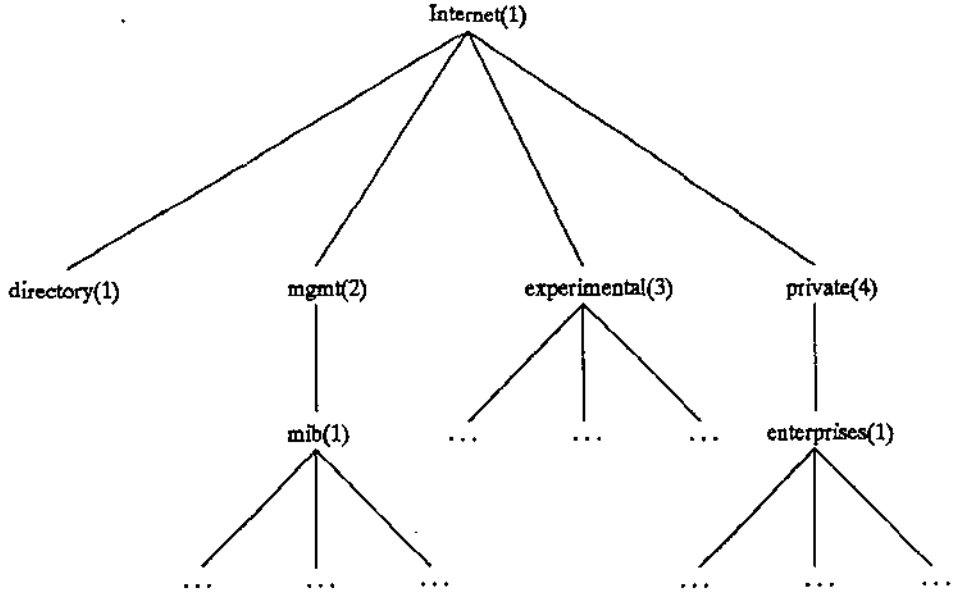


Figura 2.15: Subárvore Internet

- *ip*: grupo para gerenciamento do protocolo IP, contém estatísticas de datagramas enviados e recebidos, tabelas de endereços e de roteamento;
- *icmp*: grupo para gerenciamento do protocolo ICMP - *Internet Control Message Protocol*, contém estatísticas das mensagens enviadas e recebidas.
- *tcp*: grupo para gerenciamento do protocolo TCP, contém informações sobre os algoritmos de retransmissão, sobre cada conexão e estatísticas sobre os segmentos enviados e recebidos.
- *udp*: grupo para o gerenciamento do protocolo UDP, contém estatísticas dos datagramas enviados e recebidos.
- *egp*: grupo para o gerenciamento do protocolo EGP - *Exterior Gateway Protocol*, contém estatísticas das mensagens EGP enviadas e recebidas e uma tabela das entidades EGP vizinhas.

A necessidade de gerenciar novos objetos levou a definição da MIB II no final do ano de 1989 [MR91b]. A definição da MIB II teve como filosofia básica manter a compatibilidade com a SMI e com MIB I, dessa forma foram mantidos os objetos definidos na MIB I e adicionados novos objetos, tanto nos grupos existentes como através da criação de um novo grupo, o grupo *snmp*. O grupo *snmp* foi definido para o gerenciamento do protocolo SNMP, e mantém estatísticas das mensagens SNMP enviadas e recebidas.

A SMI prevê três mecanismos para a extensão da MIB:

- adição de novos objetos padrões, definindo-se uma nova versão da MIB padrão da *Internet*.
- adição dos objetos como não-padrão na subárvore *experimental*.
- adição de objetos privados através da subárvore *enterprises*.

Dessa forma, pode-se extender a MIB de forma a suprir as diferentes necessidades de gerenciamento. Por exemplo, novos grupos de objetos foram definidos para o gerenciamento de redes FDDI [CR93], de pontes [DLRM91], de *gateways* [MR91a, WB91], e vários outros grupos.

Identificação das Instâncias dos Obejtos

As instâncias dos objetos são manipuladas pelo protocolo de gerenciamento. A SMI não define como identificar as instâncias dos objetos, essa é uma função do protocolo SNMP. A identificação de um objeto é feita através de seu *Object Identifier* (sequência numérica que descreve o percurso na árvore de gerenciamento desde a raiz até o objeto) ou pela

composição de nomes dos objetos desde a raiz até o objeto desejado. A identificação da instância de um objeto, por sua vez, é feita pela concatenação da identificação do objeto com um sufixo. O formato do sufixo depende do tipo do objeto. Existem dois tipos de objetos:

- simples: objetos que não são colunas de tabelas; e
- tabulares: objetos que compõem uma tabela.

As seguintes regras são utilizadas para a formação dos sufixos dos objetos:

- somente instâncias de objetos que são folhas na árvore de gerenciamento podem ser identificadas;
- objetos simples recebem um sufixo único, igual a zero. Por exemplo, o objeto que descreve um sistema, *sysDescr*, no grupo *system*, tem como instância: *sysDescr.0* ou 1.3.6.1.2.1.1.1.0;
- objetos que são colunas de tabelas recebem um sufixo formado pela seleção dos valores de algumas colunas da tabela, de acordo com uma descrição que aparece na definição da tabela. São selecionadas colunas de forma a gerar sufixos únicos. Por exemplo, instâncias das colunas da tabela *ifTable*, que descreve as características das interfaces de um *host*, são identificadas pelo uso dos valores da coluna *ifIndex* como sufixo. Dessa forma, a instância do objeto *ifDescr* associada a primeira interface é: *ifDescr.1* ou 1.3.6.1.2.1.2.2.1.2.1 .

Nas operações de gerenciamento envolvendo múltiplas variáveis, a identificação das instâncias envolvidas deve ser feita uma a uma. A operação *get-next* não exige a identificação completa da instância do objeto, será devolvida como resposta o valor da primeira instância a partir da identificação passada como parâmetro. Exemplos:

- *get-next (sysDescr.0)* - retorna como resposta o nome e valor da próxima instância da árvore, que é *sysObjectID.0*, uma vez que o próximo objeto no grupo *system* é o sysObjectID;
- *get-next (sysDescr)* - retorna como resposta o nome e o valor da próxima instância da árvore, que é *sysDescr.0* .

Proteção dos Objetos Gerenciados

Uma rede pode ser controlada por mais do que um gerente, isto é, os *gateways* e *hosts* podem responder a diferentes gerentes, o que gera a necessidade de mecanismos de autenticação.

Múltiplos níveis de autorização também são suportados pelo protocolo SNMP, permitindo diferentes privilégios para gerentes específicos. Por exemplo, gerentes podem ter permissão somente para recuperar informações (*read-only*), outros podem ter permissão para recuperar e alterar as informações de gerenciamento (*read-write*).

As políticas de autenticação e autorização de acesso aos objetos descritas a seguir são usadas entre as entidades de aplicação SNMP com o objetivo de proteger os objetos gerenciados contra acessos não desejados.

O processo de autenticação é baseado na definição de *community*, onde o protocolo SNMP define uma *community* como uma relação entre um agente SNMP e um ou mais gerentes SNMP. Quando uma mensagem é enviada, esta deve conter duas partes:

- o nome da *community*, podendo incluir informações extras que certifiquem que a entidade SNMP que esta enviando essa mensagem realmente pertence àquela *community*.
- dados, contendo a operação SNMP desejada e os operandos necessários.

Se o nome da *community* corresponde a um nome conhecido pela entidade SNMP que esta recebendo a mensagem, esta é aceita, caso contrario é descartada.

Desde que a entidade SNMP que enviou mensagem é identificada como membro de uma *community* conhecida, o nó gerenciado determina o nível de acesso permitido para esta entidade.

Para cada *community* é definido um subconjunto de objetos arbitrários visíveis, que é chamado de visão. Para cada objeto em uma visão, a *community* terá um modo de acesso: *read-write* ou *read-only*.

A permissão de acesso aos objetos da MIB é definida como a combinação da permissão de acesso do gerente ao objeto via *community* à qual este pertence, com a permissão de acesso especificada na definição do objeto na MIB. Dessa maneira, pode-se obter as seguintes combinações:

permissão do gerente	**permissão de acesso dos objetos na MIB**			
	read-only	*read-write*	*write-only*	*not-accessible*
read-only	3	3	1	1
read-write	3	2	4	1

onde:

Classe	**operações permitidas**
1	nenhuma
2	*get, get-next, set*
3	*get, get-next*
4	*set*

A relação entre os agentes e gerentes definida via *community*, juntamente com o conjunto de objetos acessíveis para uma *community* e as permissões de acesso a esses garantem a proteção dos objetos contra acessos indevidos.

Os agentes e gerentes podem pertencer a várias *communitys* e com diferentes permissões. Dessa maneira, um agente pode ser gerenciado por vários gerentes e um gerente pode possuir uma série de agentes sob sua administração.

Proxy Agent

A necessidade do gerenciamento de dispositivos que não implementam o conjunto de protocolos *Internet* levou a definição de um agente especial, o *proxy agent*, que age em favor dos dispositivos externos à rede *Internet*.

Quando um gerente deseja informações de um dispositivo externo, ele contacta o *proxy agent* encarregado pelo gerenciamento deste dispositivo e envia-lhe as operações desejadas como se os objetos gerenciados do dispositivo pertencessem a MIB do *proxy agent*. O *proxy agent* identifica os objetos como do dispositivo externo e traduz a mensagem SNMP enviada pelo gerente para o tipo de interação suportada pelo dispositivo externo, como por exemplo para RPC - *Remote Procedure Call*. O dispositivo externo recebe a solicitação do gerente e responde ao *proxy agent* seguindo o mesmo protocolo, este por sua vez traduz a resposta para uma mensagem SNMP e envia-a ao gerente.

Dessa maneira o *proxy agent* funciona como um intermediário na comunicação do gerente com os dispositivos externos, que não implementam o conjunto de protocolos *Internet*. A Figura 2.16 mostra o gerenciamento de um dispositivo externo que usa RPCs para se comunicar com o *proxy agent*.

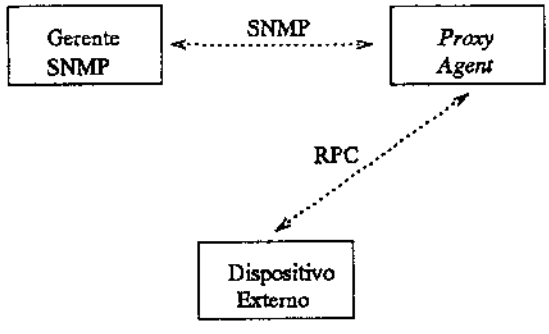


Figura 2.16: Gerenciamento via *Proxy Agent*

2.3.4 SNMPv2 - Simple Network Management Protocol version 2

O SNMPv2, mais que uma nova versão do protocolo SNMP, é uma nova versão do modelo de gerenciamento de redes *Internet*, onde foram resolvidas várias deficiências do modelo original.

O SNMPv2 não sofreu alteração com relação aos componentes envolvidos no processo de gerenciamento, que são: o gerente, os agentes, os objetos gerenciados e o protocolo de comunicação [JCW93a].

O ambiente de gerenciamento também é definido pela SMI, que especifica o subconjunto da notação ASN.1 usado na definição dos objetos gerenciados, módulos de gerenciamento e notificações disponíveis. A SMI foi expandida na segunda versão com a especificação de novos módulos, objetos e *traps* [JCW93b].

No SNMPv2, o modelo administrativo foi alterado para oferecer maior segurança [GM93a, GM93c]. Nesse sentido, foi definido o conceito de *party*, que é a especificação do conjunto de operações disponível para cada entidade envolvida no gerenciamento, ou seja, o conjunto de primitivas SNMPv2 que a entidade pode receber e enviar. Associados à cada *party* há dois protocolos, um para autenticação e outro para privacidade. O protocolo de autenticação permite que a entidade remota reconheça que a mensagem realmente foi enviada pela entidade descrita na *party*. A autenticação não é mais feita através de comunidades, mas cada mensagem SNMPv2 transmitida na rede deve conter a especificação da entidade fonte e destino, a fim de aprimorar os mecanismos de segurança. O protocolo de privacidade usa mecanismos de criptografia de dados para não permitir que outras entidades acessem o conteúdo das mensagens transmitidas na rede sem autorização.

Além desses mecanismos de segurança, pode-se definir diferentes visões da MIB na *party* de cada entidade, permitindo que cada gerente tenha acesso apenas à parte da MIB que realmente é de seu interesse.

A MIB, no SNMPv2, foi acrescida com a definição de novos ramos na subárvore *internet*: o grupo *snmpv2* e o grupo *security* [JGW93a]. Sob esses novos ramos foram definidos novos grupos de objetos para o gerenciamento de entidades da segunda versão do modelo *Internet* e para o controle de segurança. Entre outros grupos de objetos se destacam o grupo *party* [GM93b] que descreve o comportamento de uma entidade SNMPv2 e o grupo *manager-to-manager* [JGW93c] que descreve o comportamento dos gerentes SNMPv2.

O protocolo SNMPv2 [JGW93d] possui duas primitivas adicionais a sua primeira versão:

- *inform-request*: primitiva utilizada na comunicação entre dois gerentes. Essa primitiva resolve a falta de comunicação entre os gerentes na primeira versão do protocolo SNMP e facilita a implementação de diferentes níveis de gerenciamento, ou seja, hie-

rarquias de gerentes; e

- *get-bulk-request*: permite a solicitação de grupos de dados especificando o nome de uma única variável, por exemplo, pode ser utilizada para a transmissão de todas as entradas de uma tabela. Essa primitiva resolve a deficiência da primeira versão do protocolo SNMP onde todas as variáveis deveriam ser especificadas uma a uma na consulta.

Com relação ao mapeamento do protocolo SNMPv2 nas camadas inferiores, é previsto o uso de diferentes protocolos de transporte [JGW93e], mas é dada preferência ao uso do protocolo UDP, da mesma forma que na primeira versão.

A coexistência entre as duas versões do protocolo SNMP é prevista na definição da segunda versão [JGW93b]. Como o SNMPv2 oferece uma série de facilidades adicionais às oferecidas na primeira versão, pode-se configurar um gerente SNMPv2 para gerenciar também agentes SNMP. Antes do gerente enviar uma mensagem, este deve verificar em sua base de dados se o agente destino suporta o SNMP ou o SNMPv2, e enviar a mensagem na versão apropriada. Esse controle e uso das diferentes versões do protocolo SNMP pode ser feito internamente pelo gerente, sendo transparente para as aplicações de gerenciamento a versão que está sendo utilizada na comunicação com o agente. Uma outra alternativa para a coexistência das diferentes versões de gerenciamento é programar um agente SNMPv2 para agir como um *proxy agent* na comunicação entre o gerente *SNMPv2* e o agente SNMP, convertendo as primitivas do protocolo SNMPv2 para o protocolo SNMP e vice-versa. Dessa maneira, é possível conciliar o uso das duas versões do protocolo SNMP em uma rede.

Como o SNMPv2 é um modelo definido recentemente, durante o período de desenvolvimento desse projeto, não foi possível acessar nenhum sistema de gerenciamento de redes que o implementasse.

2.4 Comparação dos Modelos de Gerenciamento

Basicamente os dois modelos de gerenciamento, *Internet* e OSI, provêem o mesmo ambiente de gerenciamento de redes. Ambos são compostos por gerentes, agentes, MIB e um protocolo de gerenciamento [Nan91].

As aplicações de gerenciamento de redes são executadas em uma estação de gerenciamento (gerente). Os recursos gerenciados são representados como objetos na MIB e mantidos pelo agente. Os gerentes podem recuperar ou alterar os valores dos objetos gerenciados na MIB dos agentes e estes podem enviar notificações aos gerentes. Todas essas transações entre os agentes e os gerentes são realizadas via um protocolo de gerenciamento.

Analisando com um pouco mais de detalhe os dois modelos pode-se perceber os pontos principais que diferem um modelo do outro.

O agente do modelo *Internet* é bem mais simples que o agente OSI. A função do agente no modelo *Internet* se restringe a prestação de serviços e ao envio de *traps* preestabelecidos pelo gerente, ficando a cargo do gerente a maior parte das tarefas de gerenciamento. No modelo OSI, a maior parte das tarefas de gerenciamento é transferida para o agente, de forma que o processamento se torna mais distribuído e o agente mais complexo.

No modelo *Internet* há o conceito do *proxy agent*, agindo como intermediário no gerenciamento de dispositivos externos à rede *Internet*, enquanto que este conceito não é encontrado no modelo OSI.

2.4.1 Arquitetura dos Modelos

No que se refere à arquitetura, os dois modelos são bastante distintos [BRI93]. O modelo OSI requer a existência de uma entidade de gerenciamento em cada camada de sua pilha de protocolos, as LMEs, sendo essas entidades encarregadas da coleta de informações e do gerenciamento das camadas. Além das entidades de gerenciamento das camadas, há no modelo OSI uma entidade a nível de aplicação, a SMAE, que se comunica com as LMEs localmente e com as SMAEs remotas via o protocolo de gerenciamento, o CMIP. Os agentes e gerentes são implementados como processos de aplicação e usuários dos serviços oferecidos pelas SMAEs. O modelo *Internet* não contém entidades para o gerenciamento das camadas. A obtenção das informações de gerenciamento e seu armazenamento na MIB é responsabilidade de algum elemento de *software* qualquer não padronizado. Uma vez que as informações de gerenciamento estão armazenadas na MIB dos agentes, os gerentes podem consultá-las e alterá-las via o protocolo de gerenciamento, o SNMP. Os agentes e gerentes também são implementados como processos de aplicação no modelo *Internet*.

No que se refere a filosofia de funcionamento, o modelo *Internet* é baseado na política de *polling* e notificações, isto é, periodicamente o gerente consulta o estado de cada agente para verificar seu funcionamento, essa ordem pode ser alterada devido ao envio de uma notificação (*trap*) por parte do agente, ficando a critério do gerente a decisão de verificar o problema no momento do recebimento de um *trap*, ou de esperar o *polling* se completar e o agente ser consultado. Por outro lado, o modelo OSI é baseado na política de *reporting*, isto é, o agente estabelece uma conexão com o gerente sempre que algum evento inesperado ocorre, além disso, a qualquer momento o gerente pode estabelecer uma conexão para consultar o estado dos agentes.

2.4.2 Protocolos de Gerenciamento

O protocolo CMIP, no modelo OSI/ISO, é um protocolo orientado a conexão e utiliza toda a pilha de protocolos OSI para a transferência das primitivas de gerenciamento. Normalmente é utilizado um protocolo orientado a conexão a nível de transporte garantindo assim maior confiabilidade na transferência das mensagens de gerenciamento.

Por outro lado, o protocolo SNMP do modelo *Internet* não é orientado a conexão e geralmente utiliza-se dos serviços do UDP a nível de transporte. Embora seja um protocolo menos confiável por não ser orientado a conexão, o protocolo UDP é bastante rápido e simples, reduzindo dessa forma o impacto do sistema de gerenciamento nas redes.

Os dois protocolos, CMIP e SNMP, são definidos usando a notação ASN.1.

No que se refere às primitivas, os dois protocolos apresentam um cojunto bastante reduzido de primitivas, embora o protocolo SNMP seja um pouco mais compacto que o CMIP. O protocolo CMIP apresenta primitivas adicionais para criação e remoção de instâncias de variáveis e para o envio de comandos.

2.4.3 MIB

Os conceitos básicos dos modelos de informação usados pelos sistemas de gerenciamento OSI e *Internet* são definidos através da SMI.

A SMI descreve o cenário no qual as MIBs serão definidas e possibilita uma abordagem orientada a objetos nas MIBs dos dois modelos. Os recursos do mundo real que serão gerenciados são representados por objetos nas MIBs, seguindo a filosofia de orientação a objetos.

Em se tratando de objetos gerenciados, a *Internet* definiu um conjunto de objetos básicos que compõem a MIB, enquanto que no modelo OSI não há padronização dos objetos da MIB, ficando como função de cada administração definir os objetos que se deseja gerenciar. Os dois modelos permitem alteração dos objetos da MIB com inclusão e remoção dos objetos gerenciados a qualquer momento, dessa forma pode-se moldar as MIBs às necessidades de cada sistema gerenciado.

Os objetos da MIB do modelo *Internet* são bem mais simples, podendo ser basicamente variáveis escalares e tabelas bidimensionais de variáveis escalares. Já a representação dos objetos no modelo OSI é bem mais rica, cada objeto pode conter:

- lista de atributos;
- operações que podem ser aplicadas no objeto;
- comportamento do objeto em resposta as operações de gerenciamento;
- notificações que podem ser emitidas pelo objeto;
- pacotes condicionais que podem ser encapsulados no objeto.

Cada atributo dos objetos gerenciados no modelo OSI é uma variável simples correspondendo a um objeto da MIB do modelo *Internet*.

No que se refere às hierarquias de gerenciamento, os dois modelos se diferem. No modelo OSI são definidos três tipos de hierarquias: de herança (ou de classes), de nomeação

e de registro. No modelo *Internet*, no entanto, só é utilizada a hierarquia de registro, para identificação dos objetos de forma universal, como no modelo OSI. Nos dois modelos, a hierarquia de registro é especificada de acordo com as regras definidas pela notação ASN.1 para a atribuição de identificadores de objetos.

Com relação à identificação dos objetos, o modelo OSI apresenta facilidades adicionais que permitem a identificação de grupos de objetos para a aplicação de uma operação de gerenciamento (*scoping*) e, além disso, pode-se usar filtros restringindo os objetos selecionados pelo *scoping* apenas àqueles que satisfazem algumas condições. O modelo *Internet*, em sua primeira versão, não apresentava essas facilidades para seleção de múltiplos objetos, essa deficiência foi solucionada no SNMPv2 com a inclusão de uma nova operação que permite a recuperação de grupos de variáveis (*get-bulk*) [OSN93]. Além disso, o protocolo SNMP possui a primitiva *get-next* que permite aos gerentes identificar os objetos da MIB sem conhecimento prévio.

Nos dois modelos, as operações de gerenciamento são aplicadas aos objetos selecionados de forma atômica, isto é, ou são aplicadas em todos os objetos ou em nenhum. As operações aplicadas nos objetos selecionados com *scoping* e filtro no modelo OSI podem ser atômicas ou não, dependendo do uso de sincronização.

2.4.4 Implementações

As implementações de sistemas seguindo a primeira versão do modelo *Internet* se encontram em número bem maior que as dos sistemas no modelo OSI, principalmente porque o modelo *Internet* é bem mais simples. As implementações do modelo *Internet* são menores, mais baratas e geram sistemas mais rápidos devido ao uso de datagramas para a comunicação. Implementações do SNMPV2 ainda não estão disponíveis no mercado devido à sua recente definição.

As implementações dos sistemas de gerenciamento *Internet* podem ter problemas de comunicação devido ao tráfego gerado pelo processo de *polling* quando um gerente está encarregado do gerenciamento de muitos agentes em redes de longa distância. Em compensação no modelo OSI, se um gerente for responsável por muitos agentes, o número de *event reports* gerados também pode causar problemas de comunicação.

2.4.5 Integração dos dois Modelos

Os modelos de gerenciamento de redes OSI/ISO e *Internet* serão utilizados, simultaneamente, ainda por um bom período de tempo [OSN92]. A integração destes modelos tem sido o foco de várias pesquisas e as soluções apresentadas enfatizam o uso de um protocolo de gerenciamento de um modelo sobre a camada de transporte da pilha de protocolos do outro modelo [UWH90] [AsS93]. A combinação mais utilizada é do protocolo CMIP sobre os protocolos TCP/IP, conhecida como conjunto de protocolos CMOT - *CMIP*

over TCP/IP. O CMOT permite o gerenciamento de redes TCP/IP através do protocolo CMIP. Além dessa combinação, pode-se utilizar o protocolo SNMP sobre a camada de transporte de redes OSI/ISO [Ros93].

2.5 Considerações Finais

Comparando-se os dois modelos de gerenciamento de redes, OSI/ISO e *Internet*, pode-se observar que, de forma geral, estes possuem mais semelhanças do que diferenças, os dois modelos possuem os mesmos componentes e são utilizados para resolver os mesmos problemas, embora em redes distintas.

Em uma análise mais detalhada, pode-se notar que o modelo *Internet* é bem mais simples e prático, o que, em conjunto com a grande abrangência das redes *Internet*, impulsionou diversos fabricantes a desenvolverem sistemas de gerenciamento seguindo este modelo. Isso explica o fato de a maioria das implementações de sistemas de gerenciamento de redes encontrados no mercado seguir o modelo *Internet*.

O modelo *Internet* para o gerenciamento de redes será adotado a partir do próximo capítulo para o desenvolvimento desse trabalho, devido a sua abrangência no mundo atual e no Brasil, e principalmente porque o objetivo deste trabalho é definir um modelo para o gerenciamento do protocolo FTP, que é o protocolo de transferência de arquivos do conjunto de protocolos *Internet*.

No próximo Capítulo serão analisadas algumas implementações de sistemas de gerenciamento de redes da primeira versão modelo *Internet* que estiveram diponíveis para testes durante o desenvolvimento desse trabalho.

Capítulo 3

Sistemas de Gerenciamento de Redes

3.1 Introdução

O desenvolvimento de ferramentas para o gerenciamento de redes tem concentrado os esforços de vários fabricantes e pesquisadores nos últimos anos. Como fruto das pesquisas nessa área é possível encontrar uma série de produtos de gerenciamento de redes com características diversas, entre outros produtos comerciais podem ser destacados:

- *AIX NetView/6000* desenvolvido pela IBM para o gerenciamento de redes SNA - *System Network Architeture*, OSI e TCP/IP. Esse pacote contém a implementação dos seguintes protocolos de gerenciamento: SNA, CMIP e SNMP.
- *SunNet Manager* desenvolvido pela SUN para o gerenciamento de redes em geral e redes locais. Esse pacote implementa o gerenciamento através dos protocolos DECnet - *DEC Network Protocol* - e SNMP.
- *OpenView Network Management* desenvolvido pela HP - *Hewlett Packard, Inc* - para o gerenciamento de redes HP, OSI e TCP/IP. Esse pacote implementa os protocolos SNMP e CMIP.
- EMA - *Enterprise Management Architecture* desenvolvido pela DEC para o gerenciamento de sistemas distribuídos, redes DEC e OSI. Esse pacote contém a implementação dos seguintes protocolos para o gerenciamento de redes: CMIP, SNMP, DECnet e SNA.

Além de sistemas comerciais, pode se encontrar algumas implementações desenvolvidas por pesquisadores para fins acadêmicos e de pesquisa, cujos fontes podem ser acessados para os mesmos fins. Dentre essas implementações pode-se destacar o pacote de

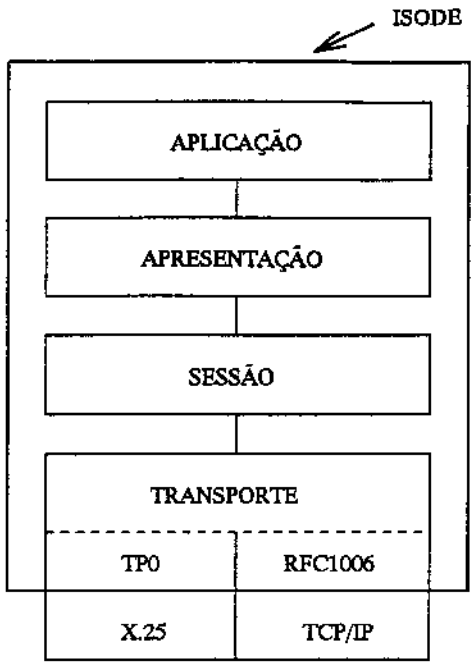


Figura 3.1: Pilha de Protocolos do ISODE

gerenciamento do ISODE - *ISO Development Environment* [Ros91a], baseado no protocolo SNMP, e o *SNMP Development Kit* [Dav89] desenvolvido no MIT - *Massachusetts Institute of Technology* - também baseado no SNMP.

Neste capítulo é apresentada uma análise de três pacotes de gerenciamento de redes que podem ser utilizados em redes TCP/IP:

- *4BSD/ISODE SNMP* [Ros91a] desenvolvido no ISODE;
- *AIX NetView/6000* [IBM92] desenvolvido pela IBM;
- *SunNet Manager* [Sun91] desenvolvido pela SUN.

3.2 4BSD/ISODE SNMP

O ISODE é um ambiente para o desenvolvimento e uso de aplicações do conjunto de protocolos OSI/ISO [Per91]. Para tanto o ISODE contém a implementação das camadas de aplicação, apresentação e sessão do modelo OSI e de parte da camada de transporte que é utilizada como uma interface entre as camandas TCP da pilha de protocolos *Internet* e a camada de sessão do modelo OSI. O ISODE também pode ser instalado sobre X25 como mostra a Figura 3.1.

O ISODE contém a implementação de alguns serviços do modelo OSI, entre outros, pode-se destacar o sistema de diretórios distribuídos, terminais virtuais e um *gateway*

entre os protocolos de transferência de arquivos FTP e FTAM - *File Transfer, Access and Management.* Além desses serviços do modelo OSI, o ISODE contém a implementação de um pacote de gerenciamento de redes, o 4BSD/ISODE SNMP, que é baseado no protocolo SNMP da *Internet.* Embora o ISODE seja um ambiente para o desenvolvimento de aplicações OSI/ISO, o pacote de gerência de redes implementa o modelo da *Internet*, isto porque o ISODE pode ser instalado sobre os protocolos TCP/IP, devido à abrangência das redes *Internet* no mundo atual e devido à expectativa de se manter, por um bom período, a convivência harmônica entre as redes *Internet* e OSI/ISO.

O 4BSD/ISODE SNMP é composto pela definição de algumas bases de informações para a MIB, pela implementação do protocolo SNMP, do protocolo SMUX - *SNMP Multiplexing*, do agente SNMP e por ferramentas para rápida prototipagem de aplicações de gerenciamento. No que se refere ao gerente, o pacote possui apenas uma interface que possibilita o envio de consultas via primitivas SNMP. As bases de informações são definidas em ASN.1.

O pacote de gerenciamento de redes do ISODE não contém a implementação de nenhum *proxy agent* para o gerenciamento de sistemas que não suportam o protocolo SNMP.

3.2.1 MIB

O 4BSD/ISODE SNMP contém uma série de arquivos com a definição de objetos para a MIB em ASN.1. Entre outros, há a definição da MIB I e da MIB II, de grupos de objetos para o gerenciamento de redes FDDI e Ethernet.

Os objetos da MIB II, em sua maioria, têm seus valores armazenados em variáveis do *Kernel*, principalmente porque os protocolos *Internet* são implementados no *Kernel*, com exceção dos protocolos de aplicação. Desse modo, o *Kernel* mantém variáveis com estatísticas sobre o fluxo de mensagens de cada um de seus protocolos. Os objetos, que não são mantidos pelo *Kernel*, têm seus valores armazenados em arquivos de configuração e são atualizados pelo administador do sistema ou por procedimentos de coleta de dados.

A ligação entre os objetos da MIB e suas respectivas instâncias no *Kernel* ou em arquivos de configuração é realizada pelos agentes SNMP.

3.2.2 Agente SNMP

O agente SNMP do ISODE, o *snmpd*, é um *daemon* que foi implementado como um processo do usuário, fazendo uso de *system calls* para acessar as variáveis do *Kernel* e atender os comandos de gerenciamento, o que reduz um pouco o desempenho do sistema. Contudo, pelo fato do agente ser um processo do usuário, há uma maior liberdade para configurá-lo e adaptá-lo às novas necessidades. Além disso, falhas no agente SNMP, não resultam em falhas do sistema UNIX ou das entidades dos protocolos.

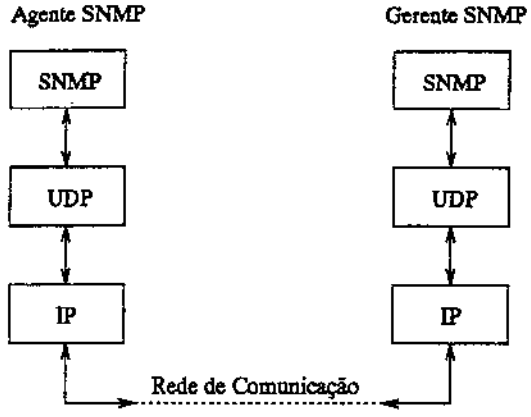


Figura 3.2: Comunicação Gerente-Agente SNMP

O processo agente é encarregado de montar a MIB do sistema, ligar cada objeto da MIB a sua instância via procedimentos de acesso e atender às solicitações enviadas pelos gerentes.

Quando é iniciado, o agente SNMP envia um *trap, o Cold Start Trap,* para o gerente, informando-o que seu processo está executando e pronto para ser contactado. Então o processo agente entra em um *looping*, a espera de mensagens do gerente. O agente SNMP espera as mensagens do gerente em uma porta UDP, como mostra a Figura 3.2.

Ao receber um datagrama UDP, o agente SNMP converte-o para uma estrutura ASN.1 e esta é interpretada como uma mensagem SNMP. Se qualquer problema é detectado nesse processo, a mensagem é descartada. Caso contrário, o campo de *community name* é conferido para verificar se o gerente possui permissão para acessar as informações do agente e só então seu pedido será analisado:

1. o agente identifica a função a ser executada;

2. todos os objetos envolvidos na requisição são localizados pelo agente e as permissões de acesso são verificadas;

3. se o gerente possui permissão para executar a função desejada em todos os objetos especificados, a operação se completa com sucesso e o agente monta uma mensagem SNMP de *get-response* para enviar ao gerente. Caso o gerente não possua permissão para acessar pelo menos um dos objetos especificados, a operação solicitada não é executada para nenhum dos objetos requeridos e uma primitiva de *get-response* de erro é enviada informando o motivo do erro na execução da operação solicitada.

O agente pode enviar mensagens de *trap* a alguns gerentes específicos quando ocorre alguma falha ou quando os objetos gerenciados adquirem um determinado valor, desde que tenham sido previamente programados para isso pelos gerentes responsáveis.

As mensagens SNMP enviadas pelo agente são codificadas em ASN.1 e inseridas em datagramas UDP para serem transmitidas via rede aos gerentes.

As funções executadas pelo agente SNMP são bastante simples, resumindo-se basicamente a atender as solicitações dos gerentes e enviar *traps* predefinidos. Isso facilita sua compreensão e implementação. Em compensação, o código do gerente concentra a maioria das funções, tornando-se mais pesado, pois centraliza o gerenciamento da rede.

3.2.3 Gerente

O pacote 4BSD/ISODE SNMP não contém a implementação de um gerente SNMP, possui apenas um programa, snmpi - *snmp initiator*, que é uma interface que possibilita o envio de comandos SNMP para os agentes, permitindo, desse modo, o teste das primitivas SNMP e a verificação da MIB dos agentes.

O 4BSD/ISODE SNMP possui uma série de ferramentas para a rápida prototipagem de aplicações de gerenciamento, como o mosy - *Managed Object Syntax Compiler (YACC based)* e o *SNMP-capable gawk* [1].

O *mosy* é um compilador que lê definições da MIB em ASN.1 e produz um arquivo usado pela biblioteca em tempo de execução (*runtime library*) para o acesso aos objetos.

O *SNMP-capable gawk* é um interpretador de comandos que permite a escrita de aplicações em um *script* bastante simples. Cada linha do *script* é comparada com padrões e caso haja casamento dos padrões uma ação é executada, as ações são escritas em uma linguagem semelhante a linguagem *C*. O gawk permite o teste de idéias e a obtenção de resultados de uma forma bastante rápida.

3.2.4 Exportando Módulos da MIB

O pacote 4BSD/ISODE SNMP provê um mecanismo baseado no protocolo SMUX [Ros91b] que permite aos processos de usuários, chamados de *SMUX Peer* ou subagentes, exportar módulos da MIB para agentes SNMP.

Desse modo, é possível que o agente SNMP mantenha as informações da MIB padrão da Internet (MIB I ou MIB II) e os outros grupos de objetos controlados por outros processos, os subagentes. A Figura 3.3 mostra um subagente que exporta seus objetos para um agente SNMP através do protocolo SMUX.

Quando um subagente é ativado, este inicia uma associação SMUX com um agente SNMP local ou remoto e registra seu módulo da MIB na base da dados do agente. O agente vai manter os objetos definidos pelo subagente na sua MIB até quando o subagente decidir removê-los.

[1] SNMP-capable gawk: adaptação do *gawk* para interpretar uma linguagem usada na construção de novas aplicações de gerenciamento.

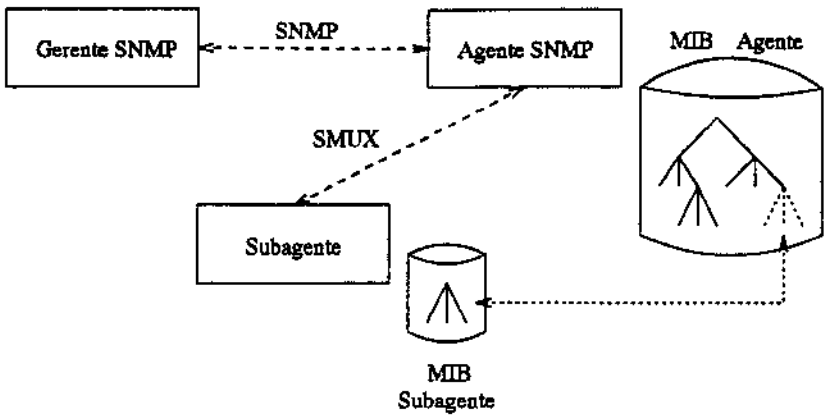


Figura 3.3: Comunicação Gerente-Subagente

Quando o agente recebe uma mensagem SNMP de *get*, *get-next* ou *set* de um gerente e a mensagem envolve uma ou mais variáveis de uma subárvore registrada por um subagente, o agente converte a mensagem SNMP para a equivalente em SMUX contendo apenas as variáveis da subárvore do subagente e envia a mensagem SMUX para ser processada pelo subagente. O subagente realiza a operação desejada e responde com uma mensagem SMUX para o agente SNMP. O agente une a resposta do subagente com o resultado do processamento local e envia a mensagem SNMP de *get-response* resultante para o gerente.

Tanto o agente como o subagente podem encerrar uma associação SMUX em qualquer instante.

As primitivas do protocolo SMUX envolvidas na comunicação entre um agente SNMP e um subagente são as seguintes:

- *open*: abertura de uma conexão SMUX;
- *register-request*: registrar um grupo de objetos na MIB do agente;
- *register-response*: confirma o registro de um grupo de objetos;
- *close*: fechar uma conexão SMUX;
- *get-request, get-next-request, set-request* e *get-response*: equivalentes às primitivas do protocolo SNMP.

3.2.5 Análise do 4BSD/ISODE SNMP

O pacote 4BSD/ISODE SNMP não está completo, apresentando falhas na implementação das primitivas de *get-next* e *set*, isso no que se refere à implementação do protocolo SNMP. Além disso, o pacote não contém a implementação de um gerente.

Para o gerenciamento de novos objetos via o 4BSD/ISODE SNMP é necessário:

1. a expansão da MIB, com o objetivo de anexar os novos objetos a serem gerenciados. Pode-se escrever um arquivo em ASN.1 com a definição dos novos objetos e executar o compilador *cmosy* para anexá-lo à MIB;

2. a definição e implementação de um novo agente ou subagente para manter os novos objetos. No caso de uso de um subagente será necessario exportar os novos objetos para um agente SNMP via SMUX; e

3. a definição e implementação de novas aplicações de gerenciamento através do *SNMP-capable gawk* para o gerenciamento dos novos objetos.

A implementação do gerenciamento de novos objetos via o 4BSD/ISODE SNMP é facilitada devido à possibilidade de acesso aos códigos fontes do pacote para fins acadêmicos.

3.3 AIX NetView/6000

O AIX NetView/6000 é uma aplicação para o gerenciamento de redes baseada nos protocolos TCP/IP [IBM92]. O AIX NetView/6000 usa o protocolo SNMP para transferir as informações sobre problemas, configuração e desempenho da rede entre os nós gerenciados e a aplicação de gerenciamento, ou seja, o gerente. Os nós gerenciados executam uma aplicação servidora, ou melhor, um agente SNMP. As informações de gerenciamento são armazenadas na MIB. O AIX NetView/6000 permite a extensão da MIB para o gerenciamento de novas informações.

3.3.1 MIB

O AIX NetView/6000 contém a definição completa da MIB I e da MIB II em ASN.1, e a declaração das MIBs para dispositivos *SNMP-capable OEM*, como roteadores CISCO. A IBM possui uma subárvore declarada sob o subgrupo *enterprise* do grupo *private* da hierarquia de gerenciamento para o controle de seus produtos.

A maioria dos objetos da MIB I e da MIB II é instanciado com variáveis do *Kernel* e mantida pelo *Kernel* do AIX, como na implementação analisada na seção anterior. As informações que não são obtidas do *Kernel* são obtidas a partir de arquivos de configuração mantidos pelo administrador da rede ou por processos de coleta de dados. As informações da MIB são acessadas pelos agentes através de procedimentos de acesso.

O AIX NetView/6000 inclui funções para carregar extensões da MIB declaradas em ASN.1, assim pode-se definir novos grupos de objetos em ASN.1 para o controle de novas informações. A forma de armazenamento e manutenção dos novos objetos deve ser especificada e implementada pelo usuário, bem como os novos subagentes que deverão se associar a um agente SNMP local ou remoto para responder as solicitações dos gerentes.

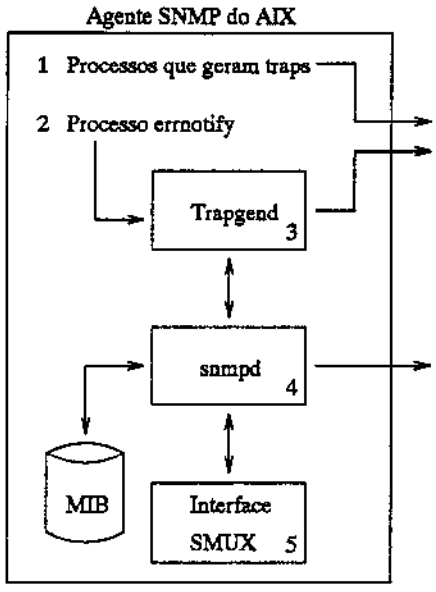


Figura 3.4: Componentes Funcionais do Agente SNMP do AIX

3.3.2 Agente SNMP

O código do agente SNMP é parte dos produtos básicos para redes do AIX sendo, portanto, fornecido junto com o sistema operacional.

Como o gerente AIX NetView/6000 usa o protocolo SNMP para comunicar-se com os agentes, qualquer implementação de agente SNMP pode ser utilizada para o gerenciamento de um sistema, o que possibilita o controle de outros sistemas diferentes do AIX, mas que suportam agentes SNMP.

A Figura 3.4 mostra os componentes funcionais do agente SNMP do AIX, que são os seguintes:

1. Processos que geram *traps*: processos que identificam a ocorrência de falhas e montam mensagens de *trap* para serem enviadas ao gerente.

2. Processo *errnotify*: processo que envia uma notificação ao *Daemon Trapgend* quando algum alerta é enviado ao arquivo *errlog*.

3. *Daemon Trapgend*: processo que converte alertas de notificação de erros armazenados no arquivo *errlog* para *traps* que são enviados ao gerente.

4. *Daemon snmpd*: processo que oferece os serviços do protocolo SNMP e contém a definição da MIB. Faz a função principal do agente, ou seja, atende às solicitações dos gerentes SNMP.

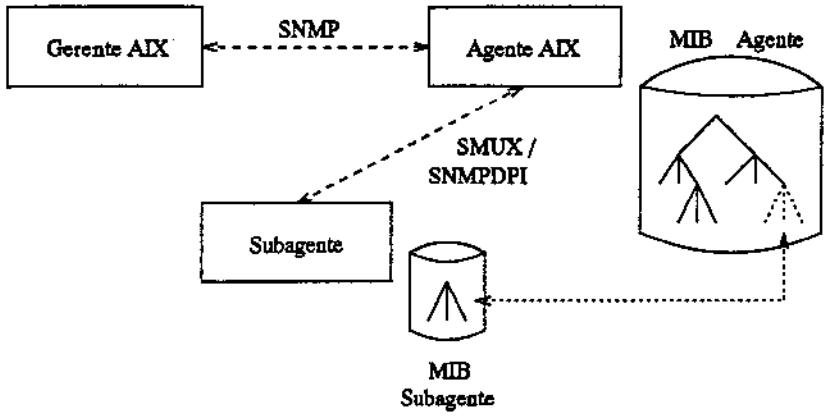


Figura 3.5: Comunicação Gerente-Subagente

5. Interface SMUX-subagente: interface com o *snmpd* que permite ao usuário escrever códigos para extensão da MIB e gerenciamento de novos objetos. Os processos escritos pelos usuários para o controle dos novos objetos são chamados de subagentes.

3.3.3 Subagentes

Os subagentes são os processos implementados pelos usuários encarregados de fazer a manutenção de novos grupos de objetos cujo gerenciamento não é oferecido pelo AIX NetView/6000. Assim, o usuário pode expandir a MIB e inserir o gerenciamento de novos objetos de seu interesse, mas necessita implementar os subagentes para manter os novos objetos e responder às solicitações dos gerentes.

A comunicação entre os subagentes e os gerentes SNMP deve ser feita através de um agente SNMP. Um subagente está sempre associado a um agente SNMP local ou remoto, para o qual exporta seus objetos e do qual recebe as solicitações enviadas pelos gerentes SNMP.Os agentes SNMP traduzem os comandos SNMP enviados pelo gerente para o protocolo utilizado pelos subagentes e vice-versa.

Os subagentes, na plataforma IBM, podem usar um dos seguintes protocolos para se comunicarem com os agentes SNMP:

- SMUX: utilizado em sistema operacional UNIX.
- SNMPDPI: utilizado em um dos seguintes sistemas: VM, MVS e OS/2.

A Figura 3.5 mostra a comunicação entre um subagente que exporta seus objetos para um agente AIX e o gerente responsável pelo controle desses objetos.

Para o gerente SNMP é transparente o protocolo utilizado pelos subagentes, o gerente envia suas requisições e recebe as respostas em SNMP. É função do agente traduzir essas solicitações e respostas para o protocolo SMUX ou SNMPDPI, conforme a necessidade.

3.3.4 Gerente SNMP

O AIX NetView/6000 oferece ao administrador da rede a habilidade de gerenciar o desempenho e a configuração da rede, além de resolver possíveis problemas na rede, utilizando o protocolo SNMP.

O gerenciamento de problemas da rede é feito a partir da identificação do problema pelo agente SNMP e sua notificação à aplicação de gerenciamento via o protocolo SNMP. As notificações são feitas a partir do envio de primitivas de *trap*. Os seguintes *traps* definidos pelo protocolo SNMP são implementados no AIX NetView/6000:

- *Cold Start, Warm Start* - informam o início ou reinício da execução de um processo agente SNMP remoto.
- *Link up, Link down* - identificam mudanças de estado de um *link* IP local de um agente SNMP.
- *Authentication error* - identifica a rejeição de uma solicitação SNMP pelo agente devido a problemas de autenticação.
- *Peer gateway failure* - identifica uma falha de um *gateway* EGP vizinho.
- *Enterprise specific* - permite a criação de traps pelos agentes SNMP. Usando a implementação do agente AIX 3.2, é possível escrever programas que podem gerar *enterprises specific traps* para gerenciar aplicações e subsistemas separados do subsistema IP.

O próprio AIX NetView/6000 implementa vários *enterprise specific traps* que refletem mudanças descobertas na configuração da rede.

No que se refere ao gerenciamento de configuração, o AIX NetView/6000 é responsável por verificar e alterar os parâmetros de configuração da rede. Analisando as informações de desempenho e configuração da rede, a aplicação de gerenciamento pode decidir por alterar a configuração da mesma com o intuito de melhorar seu desempenho. Para realizar essa função, são necessárias as primitivas *get, get-next* e *set* do protocolo SNMP. O protocolo ICMP também é utilizado para o gerenciamento de configuração, identificando e consultando nós da rede que não executam o protocolo SNMP.

O gerenciamento de desempenho no AIX NetView/6000 é utilizado para o monitoramento de estatísticas de desempenho da rede. Estas estatísticas são mantidas nas MIBs dos agentes e obtidas pela aplicação de gerenciamento através das primitivas *get* e *get-next* do protocolo SNMP.

O gerente analisa o estado dos agentes em um processo de *polling*, ou seja, periodicamente o gerente percorre todos os agentes dos quais ele é reponsável e verifica o estado, desempenho e configuração de cada um. Quando o gerente recebe uma primitiva de *trap*,

este pode optar por verificar o estado do agente que enviou a mensagem imediatamente ou por esperar que chegue a vez do agente no processo de *polling*, essa decisão depende da gravidade do problema que foi informado.

Os principais processos em execução no nó gerente são:

1. *daemon trapd*: processo que recebe os *traps* gerados pelos nós agentes ou gerados internamente pelo próprio AIX NetView/6000. Os *traps* são enviados internamente para o componente de tratamento de eventos do *xnm*;

2. *xnm*: coleção de funções que provê a interface gráfica do usuário;

3. *daemon netmon*: responsável por descobrir novos nós e periodicamente verificar o estados dos nós da rede, ou seja, executar o *polling* para verificar a configuração da rede. Usa os protocolos ICMP e SNMP quando suportado pelo nó remoto. Quando ocorre mudanças na rede, o *netmon* gera *enterprise-specific traps*, que são enviados para o *trapd* processar;

4. *daemon snmp collect*: executa o *polling* dos sistemas agentes para verificar os valores dos objetos da MIB, monta um histórico desses valores ou repassa os mesmos para o *xnm* apresentar na forma de gráficos. O *snmp collect* coleta apenas os dados especificados pelo usuário na MIB dos agentes.

5. *daemon tralertd*: converte *os traps* recebidos do *trapd* para alertas emitidos ao usuário do NetView/6000.

6. *daemon apapplid*: recebe os comandos do usuário, executa-os e retorna as respostas.

A Figura 3.6 mostra os componentes do gerente AIX NetView/6000 e ilustra a comunicação entre os processos de um nó gerenciado (agente) e um nó gerente.

Os gerentes podem passar a gerenciar novas informações, sendo necessário, para isso, informá-los da existência de um novo subagente, carregando no nó gerente o arquivo de descrição dos novos objetos inseridos na MIB em ASN.1. Dessa forma, as informações dos novos objetos da MIB tornam-se acessíveis no AIX NetView/6000 e o gerente pode acessá-las do mesmo modo que as informações padrões da MIB. Pode-se, inclusive, definir novos objetos a serem analisados no processo de *polling* e gerar gráficos a partir da coleta de dados desses objetos. A versão do AIX NetView/6000 descrita neste trabalho não prevê mecanismos para a implementação de novas funções de gerenciamento, apenas meios para coletar e analisar os valores dos novos objetos da MIB.

3.3.5 Análise do AIX NetView/6000

O pacote para o gerenciamento de redes da IBM, formado pelo gerente AIX NetView/6000 e pelo agente AIX, apresenta uma interface gráfica para o usuário bastante amigável e

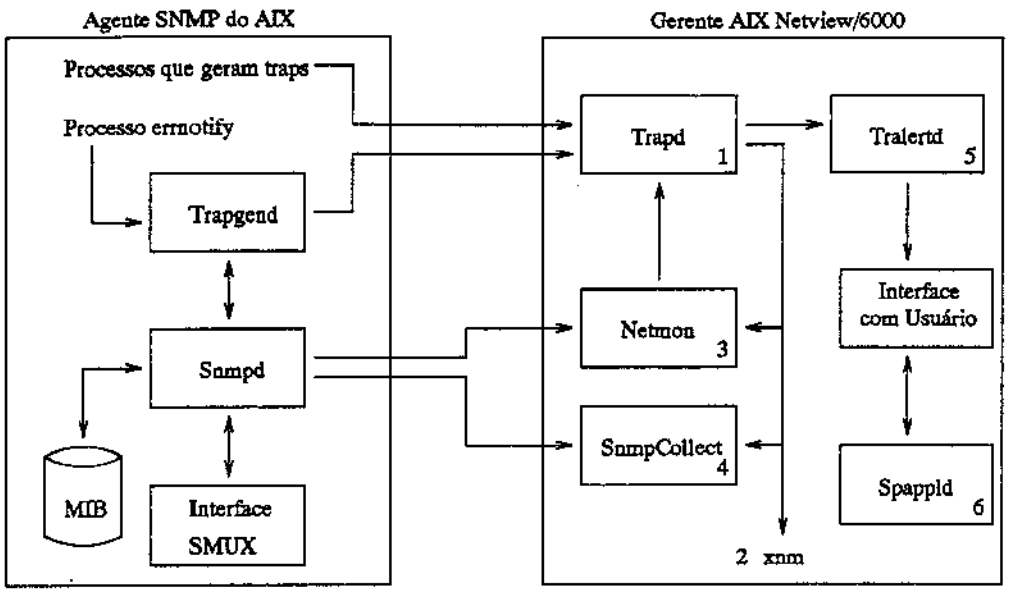


Figura 3.6: Comunicação entre os Processos do Agente e do Gerente no AIX NetView/6000

contém a implementação completa do protocolo SNMP. O gerente pode ser utilizado para o gerenciamento de agentes de outros fabricantes, desde que esses agentes utilizem o protocolo SNMP. Do mesmo modo, os agentes AIX podem ser gerenciados por outros gerentes que se comuniquem via SNMP.

O AIX NetView/6000, como o 4BSD/ISODE SNMP, permite a extensão da MIB e assim possibilita o gerenciamento de novas informações. Para gerenciar novos objetos via o AIX NetView/6000 o usuário necessita:

1. definir os novos objetos da MIB em ASN.1;

2. implementar subagentes, ou seja, processos para manter os novos objetos da MIB e fazer a comunicação com um agente local ou remoto via o protocolo SMUX ou SNMPDPI, a fim de transmitir as informações dos novos objetos aos gerentes; e

3. carregar o arquivo que contém a definição dos novos objetos em ASN.1 no nó do gerente, para que este tome conhecimento da existência de novos objetos que podem ser gerenciados, e coloque essas informações a disposição dos usuários para consulta ou geração de gráficos.

3.4 SunNet Manager

O *SunNet Manager* é uma plataforma para o gerenciamento de redes implementada de acordo com o modelo gerente-agente definido pela ISO e adaptado sobre redes TCP/IP

pela SUN [Sun91].

O objetivo do *SunNet Manager* não é resolver todas as questões de gerenciamento, mas sim definir uma plataforma para o desenvolvimento de sistemas de gerencimento de redes locais e sistemas departamentais.

O *SunNet Manager* é composto por uma base de dados de gerenciamento, a MDB - *Management DataBase*, por agentes e aplicações de gerenciamento. A comunicação entre os agentes e gerentes é realizada através de RPCs.

3.4.1 Management DataBase

A MDB é formada pela composição de uma série de arquivos de objetos gerenciados. Esses arquivos estão no formato ASCII, o que facilita alterações e expansão da base de dados.

Dois tipos de arquivos compõem a MBD: arquivos de estrutura e arquivos de instância. Os arquivos de estrutura contêm as definições dos esquemas dos agentes, com os atributos que cada agente controla, os tipos de elementos que podem ser gerenciados e os comandos que podem ser executados para cada tipo de elemento. Os arquivos de instância contêm as definições das instâncias dos objetos gerenciados.

Os gerentes e agentes SunNet usam as mesmas definições de dados, o que torna a comunicação entre eles viável.

Na plataforma *SunNet* há o mapeamento das MIB I e MIB II para arquivos ASCII, assim os objetos padrões da *Internet* podem ser gerenciados pelo *SunNet Manager*.

A MDB pode ser expandida para o gerenciamento de novos objetos, sendo necessário definir novos arquivos de objetos em ASCII, com a estrutura e instância dos novos objetos. Com a criação de novos arquivos de estrutura, pode-se definir novos tipos de objetos para atender necessidades particulares. O gerenciamento de novos objetos envolve a implementação de um novo agente para manter os novos objetos e de novas aplicações de gerenciamento para controlá-los.

Cada arquivo de objetos gerenciados é mantido e controlado por um agente. Desse modo, há vários agentes na plataforma *SunNet*, cada um responsável por um grupo de objetos da MDB.

3.4.2 Agentes SunNet

Os agentes na plataforma *SunNet* podem ser classificados em dois tipos: os que acessam diretamente os objetos gerenciados e os que acessam os objetos gerenciados de forma indireta, também conhecidos como *proxy agents*.

Os agentes que agem diretamente sobre os objetos gerenciados constituem a maioria dos agentes existentes. Estes agentes acessam os arquivos que compõem a MDB e normalmente são responsáveis por apenas um grupo de objetos da MDB.

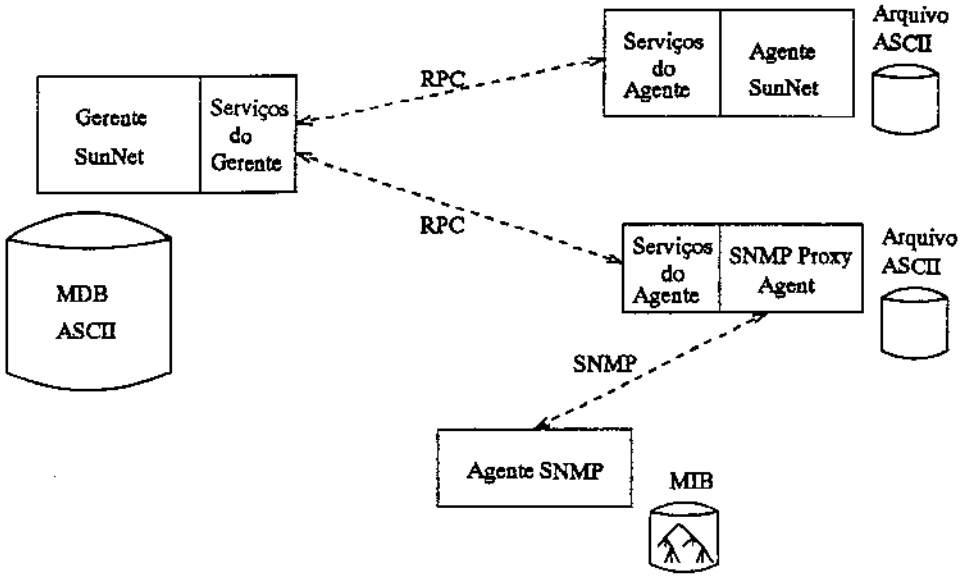


Figura 3.7: Comunicação entre um Gerente SunNet e seus Agentes

Os *proxy agents* possibilitam o gerenciamento de objetos que não são acessíveis diretamente. Esse tipo de agente funciona como um conversor de protocolos, isto é, os *proxy agents* traduzem o protocolo usado pelo gerente para um protocolo que é usado pelos sistemas que mantêm os objetos gerenciados. Os *proxy agents* permitem ao *SunNet Manager* ser estentido virtualmente para qualquer domínio. Por exemplo existe o *SNMP proxy agent* que permite ao *SunNet Manager* gerenciar qualquer dispositivo que suporte o protocolo SNMP.

O *SNMP proxy agent* pode ser instalado na mesma máquina que o gerente SunNet ou em uma máquina remota.

O *SNMP proxy agent* traduz as RPCs recebidas das aplicações de gerenciamento para comandos em SNMP que são enviados para os agentes SNMP, e vice-versa. A Figura 3.7 mostra os protocolos envolvidos na comunicação do gerente *SunNet* com dois tipos de agentes, um agente SNMP e um agente SunNet. O mapeamento da MIB I e da MIB II para arquivos ASCII permite ao *SunNet Manager* identificar os objetos e os agentes SNMP disponíveis e, através do *SNMP proxy agent*, o gerente pode gerenciá-los.

A comunicação entre os agentes e gerentes do SunNet é feita através de uma biblioteca de Serviços gerente-agente, que provê a infra-estrutura de gerenciamento e trata os serviços de comunicação. Os gerentes e agentes acessam os serviços de comunicação através de APIs - *Application Programming Interfaces*, que por sua vez usam RPC/XDR - *Remote Procedure Call/External Data Representation.*

Os agentes e gerentes não precisam conhecer os processos usados para comunicação entre estes, as APIs se encarregam desses serviços.

3.4.3 Gerente SunNet

O gerente SunNet é composto por um conjunto de processos que permitem ao usuário iniciar as tarefas de gerenciamento e coletar as informações necessárias da MDB.

A aplicação de gerenciamento central no pacote SunNet é conhecida como *SunNet Manager Console*. A Console é uma interface orientado a objeto que permite a criação de uma representação da rede. A console pode ser usada para iniciar as tarefas de gerenciamento e mostrar as informações coletadas.

Através da console pode-se solicitar os dados de gerenciamento aos agentes periodicamente e especificar os eventos que devem ser reportados por cada agente.

O gerente SunNet possui também a definição de todos os arquivos ASCII que compõem a MDB. Através destes arquivos o gerente identifica os agentes responsáveis por cada grupo de objetos e gerencia os mesmos.

3.4.4 Análise do SunNet Manager

O *SunNet Manager* é um pacote que pretende oferecer ao usuário uma plataforma para o desenvolvimento de novas aplicações de gerenciamento. Ele possui uma série de agentes e provê os mecanismos necessários para que novos agentes sejam definidos e novas funções de gerenciamento sejam implementadas [SW93] [RW94] .

Arquivos ASCII podem ser escritos pelos usuários para o gerenciamento de novos objetos e devem ser incorporados à MDB para que o gerente tome conhecimento dos novos objetos gerenciados.

A plataforma SunNet possui um compilador que traduz os arquivos de definição de objetos em ASN.1 para arquivos ASCII. Dessa forma, qualquer grupo de objetos definidos para compor a MIB pode ser facilmente mapeado para arquivos ASCII a fim de compor a MDB.

A plataforma SunNet foi desenvolvida, portanto, prevendo o gerenciamento de novos objetos. Para tanto é necessário:

1. definir os novos objetos em arquivos ASCII, que deverão compor a MDB. Esses arquivos também podem ser gerados pelo compilador que converte arquivos em ASN.1 para arquivos ASCII;

2. implementar um novo agente com duas funções básicas:

   - manter os novos objetos e acessá-los para responder as requisições do gerente;
   - fazer a comunicação com a aplicação de gerenciamento através da biblioteca de serviços do agente da plataforma SunNet. Essa função do agente inclui sua inicialização, tratamento das requisições do gerente e relatório de erros; e

3. implementar as novas funções de gerenciamento para o controle dos novos objetos gerenciados. Os arquivos ASCII dos novos objetos devem ser inseridos na MDB do gerente e novas funções devem ser implementadas para acessar os novos agentes e gerenciar os novos objetos.

3.5 Comparação dos Sistemas Analisados

Os três sistemas analisados, 4BSD/ISODE SNMP, AIX NetView/6000 e SunNet Manager, podem ser utilizados para o gerenciamento de redes *Internet*. O ISODE e o NetView implementam o modelo *Internet* de gerenciamento de rede completo, isto é, com protocolo SNMP, MIB, gerente e agentes SNMP. A principal diferença entre os dois se encontra no gerente, sendo o gerente do NetView bastante completo e com uma interface gráfica bastante amigável, enquanto o ISODE possui apenas uma interface para emissão de comandos.

O SunNet se difere dos outros sistemas, principalmente por não seguir o modelo *Internet* para o gerenciamento de redes. Apesar disso ele possui os mesmos componentes, gerente, agentes, protocolo de comunicação e base de dados de gerenciamento. As diferenças principais se encontram na base de dados - MDB - que é formada por uma série de arquivos ASCII sendo cada um destes controlado por um agente, e no que se refere a comunicação entre os agentes e gerentes que no SunNet é feita através de RPCs e não via o protocolo SNMP. Apesar dessas diferenças o SunNet Manager pode ser utilizado para o gerenciamento de agentes SNMP, através do *SNMP proxy agent* e do mapeamento das MIBs para arquivos ASCII.

Os três sistemas permitem a expansão das bases de dados e o gerenciamento de novos objetos. No ISODE e no NetView isso pode ser feito definindo os novos objetos em ASN.1 e inserindo-os na MIB, enquanto no SunNet é necessário definir os novos objetos em arquivos ASCII ou gerar esses arquivos a partir da compilação de arquivos ASN.1. Esses arquivos ASCII devem ser incluidos na MDB. Além da definição dos novos objetos, é necessário implementar os agentes ou subagentes que irão manter esses objetos e implementar as novas funções de gerenciamento requeridas.

Um esquema geral [CM94b] [CM94a] foi definido a fim de possibilitar o gerenciamento de novos objetos a partir dos três sistemas analisados neste capítulo. Esse esquema geral é mostrado na Figura 3.8 e envolve:

1. definição dos novos objetos em ASN.1 e mapeamento desses objetos para arquivos ASCII;

2. inserção dos novos objetos na MIB e dos arquivos ASCII na MDB;

3. implementação de um subagente que mantenha esses objetos e faça a comunicacação

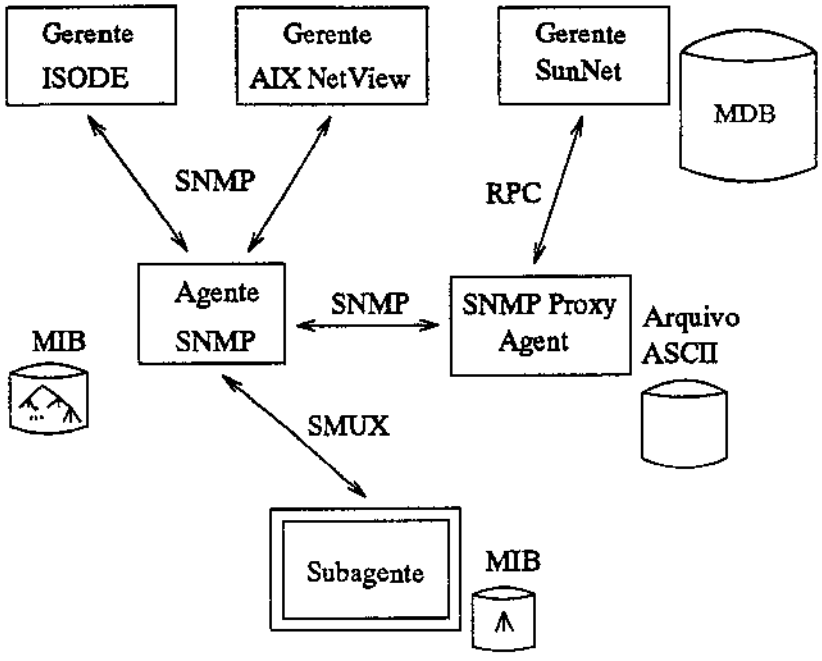


Figura 3.8: Comunicação entre um Subagente e Diferentes Gerentes

com os agentes SNMP através do protocolo SMUX para atender as solicitações dos gerentes;

4. programar os agentes SNMP para se comunicar com os subagentes; e

5. programar o *SNMP proxy agent* para se comunicar com os agentes SNMP adequados.

Os protocolos envolvidos no gerenciamento de um subagente através dos três sistemas analisados nesse capítulo são os seguintes:

- RPC : usado na comunicação entre o *SunNet Manager* e o *SNMP proxy agent*;
- SNMP : usado na comunicação do agente SNMP com o gerente do 4BSD/ISODE SNMP, o gerente do *AIX NetView/6000* e o *SNMP proxy agent*;
- SMUX : usado na comunicação do agente SNMP com o subagente.

3.6 Considerações Finais

Os sistemas de gerenciamento de redes 4BSD/ISODE SNMP, *AIX NetView/6000* e *SunNet Manager* foram analisados neste trabalho porque os três podem ser utilizados para o gerenciamento de redes *Internet* e, portanto, para o gerenciamento do protocolo FTP. Além disso, os três sistemas estiveram disponíveis para testes, mesmo que por curtos períodos de tempo.

Os seguintes testes foram realizados com os três sistemas, com o objetivo de verificar a interoperabilidade deles:

- gerente *Aix NetView/6000* - agente ISODE: verificou-se que um gerente *NetView* comunica-se com um agente do ISODE sem problemas, o que era esperado uma vez que as duas entidades suportam o protocolo SNMP.

- interface *snmpi* do ISODE - agente *AIX NetView/6000*: a interface de comandos do ISODE pode ser utilizada para acessar e gerenciar um agente AIX. O protocolo utilizado na comunicação entre os dois é o SNMP.

- *SunNet Manager* - *SNMP proxy agent* - Agente ISODE: o uso do *SNMP proxy agent* permite ao gerente *SunNet* acessar e gerenciar o agente SNMP do ISODE. Há dois protocolos envolvidos nessa comunicação: RPC entre gerente *SunNet* e o *proxy agent* e SNMP entre o *proxy agent* e o agente e

Não foi possível realizar testes entre o *SunNet Manager* e o *AIX NetView/6000*, uma vez que os dois sistemas estiveram disponíveis em períodos distintos. Porém, pode-se afirmar que o gerente *SunNet* gerencia um agente AIX, uma vez que o processo é semelhante ao utilizado para gerenciar um agente do ISODE.

Não é possível usar a interface de comandos do ISODE ou um gerente do AIX NetView/6000 para o gerenciamento de um agente do *SunNet* sem a construção de um *proxy agent* com funções inversas às do *SNMP proxy agent*. Isso significa que o novo *proxy agent*, teria a função de converter os comandos SNMP para RPCs e as respostas e *traps* de RPC para SNMP. Além disso, é necessário mapear os arquivos da MDB para os grupos de objetos da MIB.

Os testes realizados confirmam a viabilidade do gerenciamento um agente SNMP do ISODE através de outros gerentes. Consequentemente pode-se afirmar que o gerenciamento de novas informações inseridas na MIB de um agente ISODE via os gerentes do *NetView* e *SunNet* também é possível.

O 4BSD/ISODE SNMP foi selecionado para a implementação deste trabalho, devido a sua disponibilidade e possibilidade de adaptação de seus códigos. Embora a implementação tenha sido desenvolvida no ISODE, esta pode ser acoplada aos outros dois sistemas analisados neste capítulo, isto é, a implementação do gerenciamento do protocolo FTP foi baseada no esquema geral apresentado na seção anterior.

O esquema para o gerenciamento do protocolo FTP é apresentado no próximo capítulo.

Capítulo 4

Modelo para Gerenciamento do Protocolo FTP

4.1 Introdução

Analisando os grupos de objetos definidos como padrão no modelo *Internet* pode-se observar que estes não incluem objetos para o gerenciamento de protocolos de aplicação, como o SMTP, o FTP e o Telnet. A própria evolução do gerenciamento de redes explica a inexistência do gerenciamento de aplicações, uma vez que a necessidade de ferramentas para o gerenciamento de redes advem principalmente da dificuldade de gerenciar as configurações físicas das redes. O gerenciamento de redes vem evoluindo a partir das camadas inferiores das redes (física, enlace, rede e tranporte) e a segunda base de informações padronizada pelo *Internet*, isto é, a MIB II, inclui o gerenciamento de um protocolo de aplicação, o SNMP. Atualmente tem surgido a necessidade de se definir grupos de objetos para o gerenciamento de outros protocolos de aplicação.

O objetivo principal do gerenciamento dos protocolos de aplicação é a coleta de estátisticas sobre o fluxo de mensagens gerado por cada protocolo de aplicação. Desse modo, é possível obter informações mais realistas sobre a origem do tráfego nas redes e fazer uma melhor alocação de recursos como linhas de comunicação com melhores velocidades e servidores de arquivos.

Neste capítulo é definido um esquema para o gerenciamento do protocolo FTP. O protocolo FTP foi escolhido para o desenvolvimento desse trabalho por ser o principal gerador de tráfego nas redes *Internet*, segundo pesquisas recentes na área [DJE92] [KJ94]. Com o gerenciamento do protocolo FTP, informações mais precisas podem ser obtidas sobre os arquivos e servidores mais acessados, e, a partir desses dados, pode-se planejar melhor a organização dos arquivos e servidores na rede com a finalidade de redução do tráfego de dados.

O gerenciamento do protocolo FTP pode ser feito em domínios. A próxima seção

apresenta a definição de domínios de gerenciamento. A definição dos objetos que devem compor o grupo *ftp* na MIB é apresentada na seção 4.3.1, e as seções 4.3.2 e 4.3.3 apresentam, respectivamente, a definição do subagente FTP e de algumas aplicações de gerenciamento para o protocolo FTP.

4.2 Gerenciamento de Redes em Domínios Hierárquicos e Federativos

O gerenciamento de uma rede de longa distância, por ser uma tarefa bastante complexa, pode ser particionado em domínios [WT94]. Um domínio é definido como um conjunto de componentes com os mesmos objetivos e subordinados a um gerenciamento comum [VTH90]. Os componentes de um domínio podem ser, por exemplo, outros domínios, sistemas ou *softwares*.

Um domínio de gerenciamento [CM94b] é formado por um conjunto de agentes subordinados a um ou mais gerentes do domínio, onde cada gerente é responsável pela administração de diferentes aspectos da rede. Por exemplo, pode-se definir como um domínio de gerenciamento uma instituição com uma rede local que suporte a instalação dos agentes nos sistemas a serem gerenciados e de um ou mais gerentes para o controle desses sistemas.

Neste trabalho, os domínios são classificados em dois tipos: Domínios Hierárquicos e Federativos.

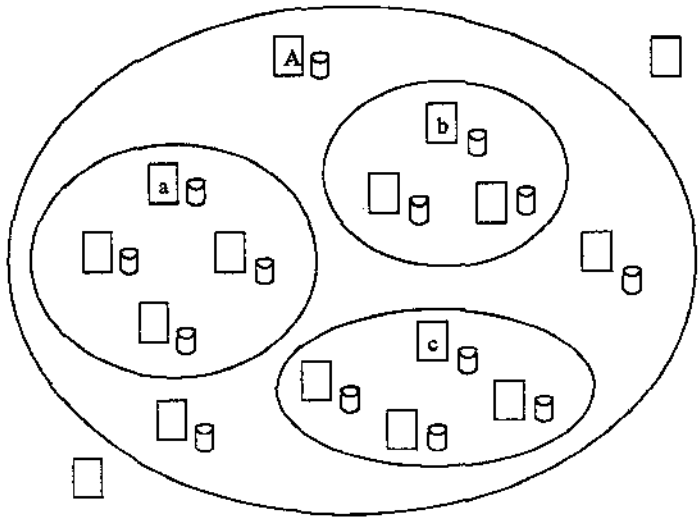
4.2.1 Domínios Hierárquicos

Domínios Hierárquicos refletem a hierarquia administrativa da rede, isto é, utiliza-se a divisão administrativa da rede para a definição dos domínios hierárquicos de gerenciamento.

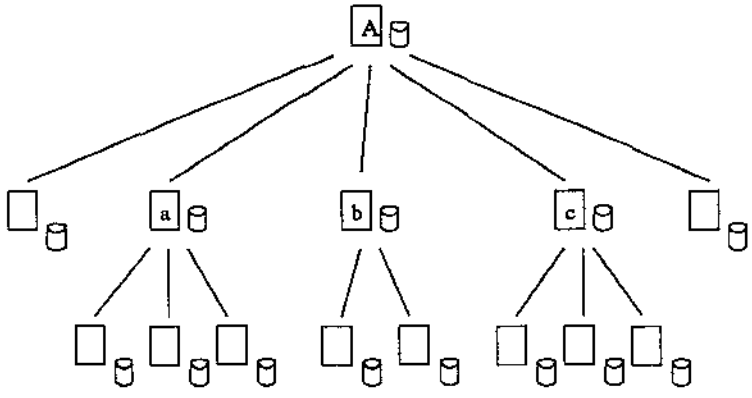
Um domínio hierárquico é formado por um conjunto de agentes subordinados a um ou mais gerentes do domínio em um nível. Um conjunto de gerentes de um determinado nível pode estar subordinado a um gerente de um nível superior, compondo um outro domínio hierárquico, e assim sucessivamente.

A Figura 4.1 ilustra um exemplo da organização de uma rede em domínios hierárquicos representados em conjuntos e árvores. Neste exemplo, há uma série de sistemas em uma rede onde alguns são gerenciados e outros não. Os sistemas a, b e c são gerentes de domínios e estão subordinados a um gerente de nível superior, A.

Um esquema geral para o gerenciamento em domínios hierárquicos pode ser definido da seguinte forma: cada sistema gerenciado deve manter sua MIB com todas as informações necessárias, um *software* agente e estar subordinado a pelo menos um gerente. Instituições de grande porte, isto é, com várias redes locais e grande número de sistemas, podem manter um gerente local. Os gerentes locais por sua vez devem manter uma MIB com as



a) Representação de Domínios Hieráquicos em Conjuntos



b) Representação de Domínios Hierárquicos em Árvores

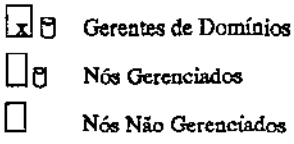


Figura 4.1: Domínios Hierárquicos

informações de seus subordinados que saem de seu domínio de gerenciamento e estarão sendo controlados por gerentes de um domínio hierárquico superior, por exemplo gerentes regionais. Os gerentes regionais da mesma maneira mantêm uma MIB com informações de seus subordinados que saem de seu escopo de gerenciamento, e estão sob o domínio de gerentes nacionais.

A subdivisão do gerenciamento de redes em domínios hierárquicos permite a obtenção de estatísticas sobre o fluxo de informação em cada domínio e entre os domínios. Por exemplo, o fluxo de informação em uma rede pode ser controlado nos seguintes níveis:

- regional: envolvendo as mensagens que são trocadas por sistemas que estão localizados em uma mesma região ou domínio;
- nacional: fluxo de mensagens geradas em um sistema e enviadas para sistemas de outro domínio regional, mas do mesmo domínio nacional, isto é, mensagens que são trocadas por sistemas de diferentes regiões de um país; e
- internacional: fluxo de mensagens trocadas por sistemas de diferentes domínios nacionais, isto é, mensagens enviadas para, ou recebidas de, sistemas que estão localizados em outros países.

A título de exemplificar a aplicação desse esquema de gerenciamento em domínios em uma rede acadêmica, pode-se definir a subdivisão da RNP em domínios e a instalação de agentes e gerentes conforme descrito a seguir.

Um gerente nacional poderia ser instalado em um dos nós do *backbone* da rede, por exemplo no NOC - *Network Operation Center*, onde esse gerente seria o responsável pelo gerenciamento de todo o *backbone* nacional. Em cada rede regional, poderia ser instalado um gerente regional que controlaria essa rede e manteria a base de informações para acesso do gerente nacional. As instituições de grande porte poderiam manter um gerente controlando suas redes locais e manteriam sua base de informações para o acesso dos gerentes regionais aos quais elas estariam subordinadas. Os sistemas gerenciados manteriam suas MIBs, além de um *software* agente encarregado de atualizar as informações de gerenciamento e de responder às consultas geradas pelos gerentes autorizados.

A Figura 4.2 mostra o gerenciamento do *backbone* da RNP, estando instalado o gerente nacional no NOC em São Paulo, e a Figura 4.3 ilustra o gerenciamento de uma das redes regionais da RNP, no caso a Rede Estadual do Rio de Janeiro, com a instalação do gerente regional no LNCC - Laboratório Nacional de Ciência da Computação. Instituições com várias redes locais e grande número de equipamentos como, por exemplo, a UFRJ - Universidade Federal do Rio de Janeiro - e a PUC/RJ - Pontifícia Universidade Católica do Rio de Janeiro - poderiam manter gerentes locais para controlar suas redes.

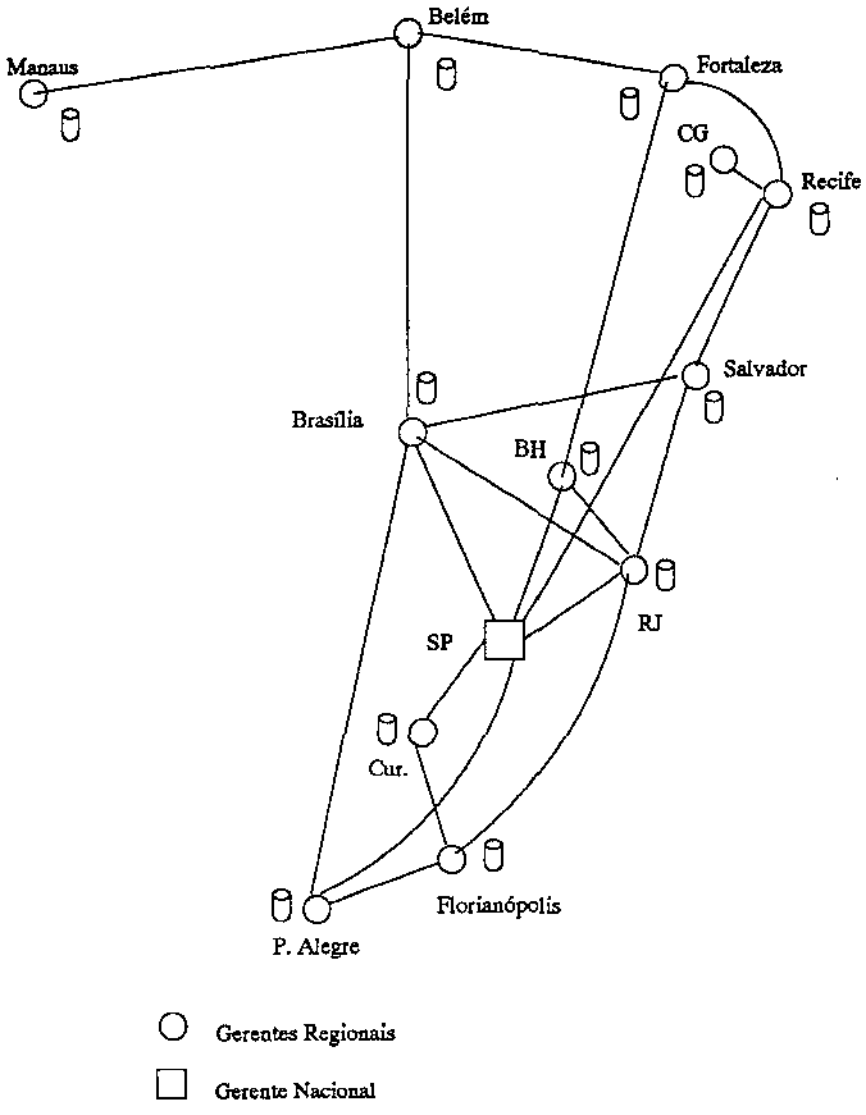


Figura 4.2: Exemplo do Gerenciamento da RNP a Nível Nacional

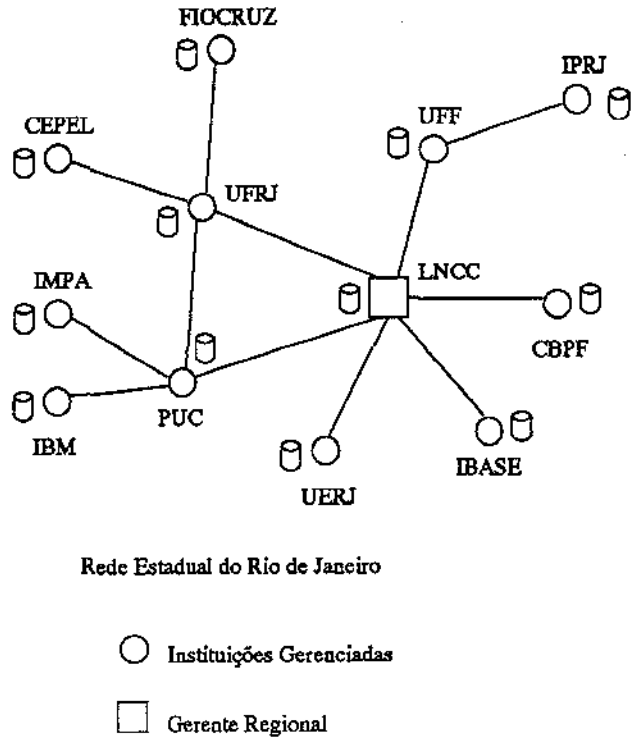


Figura 4.3: Exemplo do Gerenciamento da RNP a Nível Regional - Rede Rio

4.2.2 Domínios Federativos

A necessidade de se ter sistemas que estejam localizados em diferentes domínios hierárquicos, mas possuindo a mesma administração e os mesmos objetivos, levou à definição de domínios federativos. Um domínio federativo é formado por uma série de componentes localizados em diferentes domínios hieráquicos, mas com administração e objetivos comuns.

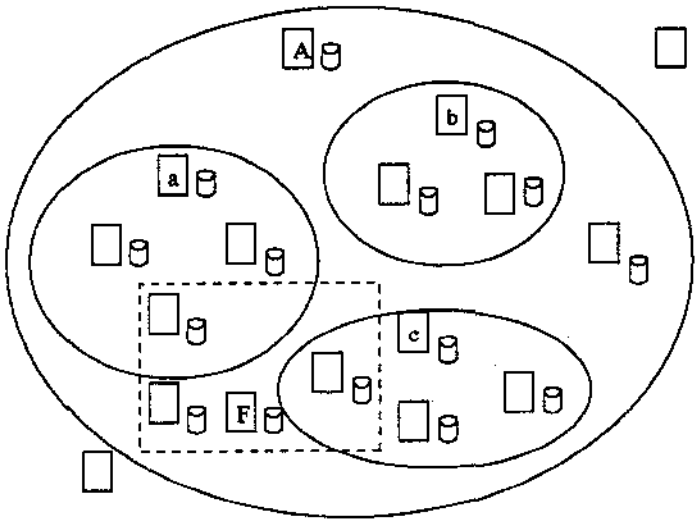
Os domínios de gerenciamento federativos são compostos por um conjunto de agentes que podem estar em diferentes domínios hierárquicos (administrativos) e por um ou mais gerentes do domínio que controlam o fluxo de informação entre esses agentes.

A Figura 4.4 mostra um exemplo de domínio federativo, onde F é um gerente de um domínio federativo responsável pela administração de sistemas que estão no escopo de gerenciamento dos domínios hierárquicos controlados pelos gerentes a, c e A.

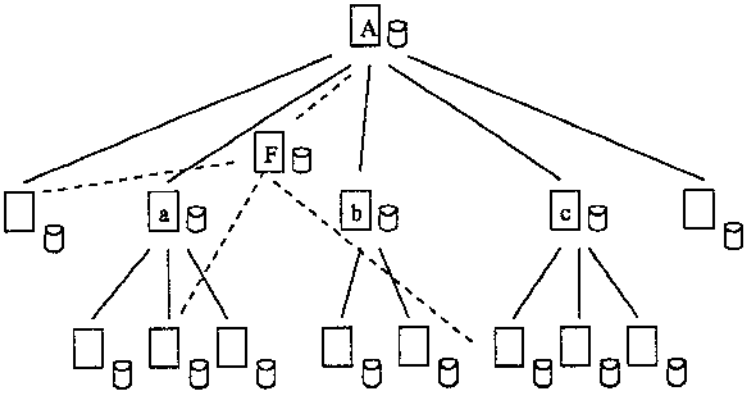
Uma aplicação natural para o gerenciamento em domínios federativos é o controle de instituições com sedes em diferentes regiões, como a Embrapa, a RNP e o CNPq. Essas instituições poderiam ser gerenciadas de duas maneiras:

- Gerenciamento Distribuído: onde cada sede da instituição responderia ao gerente regional do domínio onde ela está localizada, independentemente das demais.

- Gerenciamento Centralizado: onde as sedes, além de responderem aos gerentes re-



a) Representação de Domínios em Conjuntos



b) Representação de Domínios em Árvores

Figura 4.4: Domínio Federativo F

gionais de seus domínios, responderiam a um gerente da instituição encarregado de centralizar as informações de todas as sedes da mesma. Isso seria feito através da definição de domínios federativos para as instituições desse tipo.

Outra área de aplicação de domínios federativos surge quando pretende-se controlar o fluxo de informação entre dois sistemas específicos, como por exemplo, entre a Unicamp e a UFRJ, que estão subordinadas a diferentes gerentes regionais. Esse tipo de controle poderia ser realizado através da definição de um domínio federativo sob a administração de um novo gerente.

4.3 Gerenciamento do Protocolo FTP

O gerenciamento de um protocolo de aplicação, como no caso o FTP, pode ser feito em vários domínios, tanto hierárquicos como federativos.

Assim, cada sistema, que suporta FTP e que se deseja gerenciar, deve conter sua base de informações de gerenciamento e um agente que é contactado por pelo menos um gerente ao qual está subordinado. Este gerente deve manter, em sua base de informações, dados sobre as operações FTP dos clientes e servidores FTP do seu domínio que envolvem sistemas de outros domínios, ou seja, sistemas que não estão em seu escopo de gerenciamento. Deve-se instalar um agente, nos sistemas onde há gerentes, para fornecer informações para os gerentes de nível mais alto; e assim sucessivamente.

Portanto, para realizar o gerenciamento do protocolo FTP seguindo o esquema descrito acima foi necessária a definição de:

- um grupo *ftp* de objetos para ser inserido na MIB;
- um agente ou subagente para manter esses objetos do grupo *ftp*, permitindo aos gerentes acessá-los para leitura ou alteração; e
- um conjunto de serviços que podem ser oferecidos a partir do gerenciamento do protocolo FTP e um gerente que ofereça esses serviços automaticamente.

4.3.1 Grupo *ftp* de Objetos para a MIB

O grupo *ftp* de objetos foi definido a partir da especificação do protocolo FTP [PR85], apresentado no Apêndice A, e da definição dos outros grupos de objetos que compõe a MIB. A definião do grupo *ftp* de objetos foi feita visando manter as informações necessárias para suprir uma série de serviços importantes de gerenciamento envolvendo o protocolo FTP e servidores de arquivos acessíveis via esse protocolo, serviços esses que são apresentados na seção 4.3.3.

O grupo de objetos *ftp* foi dividido em quatro subgrupos ou bases e deve ser inserido na MIB dos sistemas gerenciados de acordo com as seguintes características:

1. *ftpcli*: subgrupo que mantém informações gerais sobre as transações de transferências de arquivos a partir de um cliente FTP. Esta base deve ser instalada nos sistemas gerenciados que suportem FTP. Os objetos que irão compor essa base devem ser selecionados pelo administrador da rede, de acordo com as necessidades de gerenciamento de cada sistema. A seguir, é apresentado um conjunto de objetos que estará disponível para os administradores:

   - *ftpcliDescr*: objeto que descreve o software de transferência de arquivos usado no sistema gerenciado;
   - *ftpcliInMsgs*: objeto para contabilizar o número de mensagens FTP recebidas pelo cliente FTP (confirmações ou mensagens de erros);
   - *ftpcliOutMsgs*: objeto para contabilizar o número de mensagens FTP enviadas;
   - *ftpcliInBytes*: objeto para contabilizar o número de *bytes* recebidos devido a mensagens FTP ou arquivos recebidos.
   - *ftpcliOutBytes*: objeto para contabilizar o número de *bytes* enviados devido a mensagens FTP ou transfêrencia de arquivos;
   - *ftpcliInFiles*: objeto que contabiliza o número de arquivos recebidos de entidades remotas;
   - *ftpcliOutFiles*: objeto que contabiliza o número de arquivos enviados para entidades remotas;
   - *ftpcliConns*: objeto que contabiliza o número de conexões abertas para transferência de arquivos, isto é, tanto conexões para transferência de comandos FTP como para transferência de dados;
   - *ftpcliDatConns*: objeto que contabiliza o número de conexões abertas exclusivamente para transferência de dados;
   - *ftpcliConnTime*: objeto que mantém o tempo médio em que as conexões permaneceram abertas;
   - *ftpcliBadAdds*: objeto que contabiliza o número de conexões recusadas pela entidade remota por erro de endereçamento;
   - *ftpcliConnErrors*: objeto que contabiliza o número de erros de conexão;
   - *ftpcliBadAuts*: objeto que contabiliza o número de acessos FTP recusados por falta de autorização. Refere-se ao número de vezes que o cliente tentou acessar algum servidor e não obteve exito por falta de autorização.

2. *ftpserv*: este subgrupo será instalado nos sistemas gerenciados que contenham um servidor de arquivos. Os objetos propostos neste trabalho para compor esta base são os seguintes:

   - *ftpservDescr*: objeto que descreve o servidor de arquivos;
   - *ftpservInMsgs*: objeto que contabiliza o número de mensagens FTP recebidas pelo servidor FTP;
   - *ftpservOutMsgs*: objeto que contabiliza o número de mensagens FTP enviadas pelo servidor FTP;
   - *ftpservInBytes*: objeto que contabiliza o número de *bytes* recebidos devido a mensagens FTP ou arquivos recebidos.
   - *ftpservOutBytes*: objeto que contabiliza o número de *bytes* enviados devido a mensagens FTP ou transfêrencia de arquivos;
   - *ftpservInFiles*: objeto que contabiliza o número de arquivos recebidos pelo servidor;
   - *ftpservOutFiles*: objeto que contabiliza o número de arquivos enviados pelo servidor;
   - *ftpservConns*: objeto que contabiliza o número de conexões abertas para transferência de arquivos, isto é, tanto conexões para transferência de comandos FTP como para transferência de dados;
   - *ftpservDatConns*: objeto que contabiliza o número de conexões abertas exclusivamente para transferência de dados;
   - *ftpservConnTime*: objeto que mantém o tempo médio em que as conexões permaneceram abertas;
   - *ftpservExcConns*: objeto que contabiliza o número de conexões recusadas por excesso de connexões ativas no servidor;
   - *ftpservConnErrors*: objeto que contabiliza o número de erros de conexão;
   - *ftpservBadAuts*: objeto que contabiliza o número de acessos FTP recusados pelo servidor por falta de autorização;
   - *ftpservFileTable*: tabela que contém informações sobre os arquivos disponíveis no servidor. Essa tabela contém as seguintes informações de cada arquivo: nome, tipo e tamanho do arquivo, permissões de acesso, quem gravou, data de gravação e seu estado. Além disso mantém uma lista dos usuários que acessaram este arquivo recentemente.

A inserção desses objetos do grupo *ftpserv* na MIB depende das necessidades de gerenciamento de cada sistema. Por exemplo, pode-se controlar apenas o fluxo de

dados de um servidor, sem se preocupar com o controle dos arquivos disponíveis, não sendo necessária a tabela de arquivos. Em outro sistema pode-se desejar apenas o controle dos arquivos disponíveis e transferidos via rede, sendo necessária a tabela de arquivos e os objetos *ftpservInFiles e ftpservOutFiles.*

3. *ftpman*: subgrupo que poderá ser instalado em todos os sistemas que possuem um *software* gerente de FTP. Essa base pode manter os seguintes objetos:

   - *ftpmanDescr*: objeto que descreve o gerente FTP;
   - *ftpmanInMsgs*: objeto que contabiliza o número de mensagens FTP recebidas de sistemas que estão fora do domínio do gerente;
   - *ftpmanOutMsgs*: objeto que contabiliza o número de mensagens FTP enviadas para sistemas que estão fora do domínio do gerente;
   - *ftpmanInBytes*: objeto que contabiliza o número de *bytes* recebidos devido a mensagens FTP ou arquivos recebidos de sistemas externos ao domínio do gerente.
   - *ftpmanOutBytes*: objeto para contabilizar o número de *bytes* enviados devido a mensagens FTP ou transfêrencia de arquivos para sistemas externos ao domínio do gerente;
   - *ftpmanInFiles*: objeto que contabiliza o número de arquivos recebidos de entidades externas ao domínio do gerente;
   - *ftpmanOutFiles:* objeto que contabiliza o número de arquivos enviados para entidades externas ao domínio do gerente;
   - *ftpmanConns*: objeto que contabiliza o número de conexões abertas para transferência de arquivos, isto é, tanto conexões para transferência de comandos FTP como para transferência de dados, envolvendo sistemas externos ao domínio do gerente;
   - *ftpmanDatConns*: objeto que contabiliza o número de conexões abertas exclusivamente para transferência de dados, envolvendo sistemas externos ao domínio do gerente;
   - *ftpmanConnTime*: objeto que mantém o tempo médio em que as conexões com sistemas externos ao domínio do gerente permaneceram abertas;
   - *ftpmanBadAdds*: objeto que contabiliza o número de conexões recusadas por erro de endereçamento, envolvendo sistemas externos ao domínio do gerente;
   - *ftpmanConnErrors*: objeto que contabiliza o número de erros de conexão, envolvendo sistemas externos ao domínio do gerente;

- *ftpmanBadAuts*: objeto que contabiliza o número de acessos FTP recusados por falta de autorização, envolvendo sistemas externos ao domínio do gerente;
- *ftpmanServTable*: tabela que contém as informações sobre os servidores FTP pertencentes ao domínio do gerente em questão. Esta tabela contém o endereço e o estado de cada servidor FTP;
- *ftpmanCliTable*: tabela que contém as informações sobre os clientes FTP pertencentes ao domínio do gerente em questão. Esta tabela contém o endereço e o estado de cada cliente FTP;
- *ftpmanManTable*: tabela que contém as informações sobre os gerentes FTP pertencentes ao domínio deste gerente. Esta tabela contém o endereço e o estado de cada gerente FTP.

4. *ftpext*: este subgrupo será instalado quando se deseja controlar o fluxo de informação entre dois sistemas específicos. Esta base deverá ser instalada em um dos dois sistemas envolvidos e pode conter qualquer subconjunto dos objetos definidos a seguir:

   - *ftpextDescr*: objeto que descreve o sistema remoto envolvido nesse controle;
   - *ftpextInMsgs*: objeto que contabiliza o número de mensagens FTP recebidas do sistema remoto;
   - *ftpextOutMsgs*: objeto que contabiliza o número de mensagens FTP enviadas para o sistema remoto;
   - *ftpextInBytes*: objeto que contabiliza o número de *bytes* recebidos devido a mensagens FTP ou arquivos recebidos do sistema remoto;
   - *ftpextOutBytes*: objeto para contabilizar o número de *bytes* enviados devido a mensagens FTP ou transfêrencia de arquivos para o sistema remoto;
   - *ftpextInFiles*: objeto que contabiliza o número de arquivos recebidos de entidades remotas;
   - *ftpextOutFiles*: objeto que contabiliza o número de arquivos enviados para entidades remotas;
   - *ftpextConns*: objeto que contabiliza o número de conexões abertas para transferência de arquivos, isto é, tanto conexões para transferência de comandos FTP como para transferência de dados, entre os dois sistemas em questão;
   - *ftpextDatConns*: objeto que contabiliza o número de conexões abertas exclusivamente para transferência de dados, envolvendo os dois sistemas em questão;
   - *ftpextConnTime*: objeto que mantém o tempo médio em que as conexões entre os dois sistemas permaneceram abertas;

- *ftpextConnErrors*: objeto que contabiliza o número de erros de conexão, envolvendo os dois sistemas em questão;
- *ftpextBadAuts*: objeto que contabiliza o número de acessos FTP recusados por falta de autorização, envolvendo os dois sistemas em questão.

Não se recomenda o controle do fluxo de informação entre várias máquinas ou mesmo entre duas máquinas por longos períodos. Esse serviço deve ser usado em casos especiais e por períodos limitados.

A definição do grupo de objetos *ftp* e seus subgrupos em ASN.1 é apresentada no Apêndice B.

A configuração do grupo *ftp* na MIB de cada sistema gerenciado depende dos serviços que se deseja oferecer e da configuração do sistema. A seguir são apresentadas algumas configurações do grupo *ftp* de acordo com os tipos de serviços:

1. Estimativa simples do fluxo de informação gerado em um sistema pelo protocolo FTP: para oferecer esse tipo de serviço é necessário instalar apenas o subgrupo *ftpcli* e/ou *ftpserv* no grupo *ftp* do sistema em questão, dependendo apenas dele ter um cliente e/ou servidor FTP.

2. Controle do fluxo de informação em um servidor de arquivos: esse tipo de serviço requer apenas a inserção do subgrupo *ftpserv* na MIB do sistema que possui o servidor de arquivos.

3. Controle do fluxo de informação em domínios hierárquicos e federativos: esse tipo de serviço requer a inserção dos subgrupos *ftpcli* e *ftpserv* nos sistemas gerenciados de um domínio que tenham, respectivamente, um cliente ou servidor FTP, e a instalação do subgrupo *ftpman* na MIB do sistema gerente do domínio, para ser consultada pelos gerentes de nível superior. A base dos gerentes contém, entre outras informações, a relação de todos os clientes, servidores e gerentes do domínio, possibilitando aos gerentes a classificação das transações FTP como internas ou externas aos seus domínios.

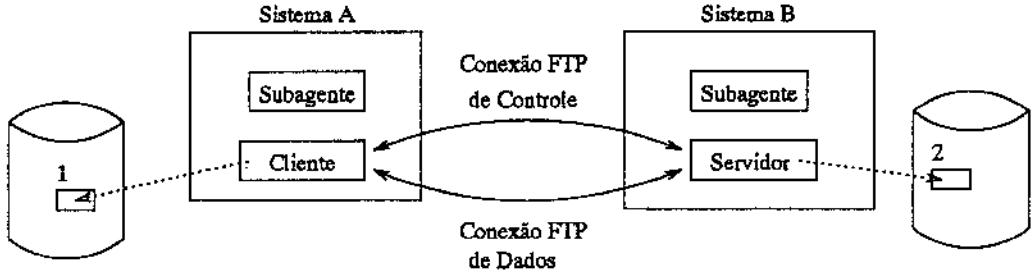
4. Controle do fluxo de informação entre dois sistemas específicos: esse tipo de serviço vai requerer, além dos subgrupos *ftpcli* e *ftpserv* nos sistemas gerenciados onde haja clientes e servidores FTP, a inserção do subgrupo *ftpext* na MIB de pelo menos um dos sistemas em questão. Caso seja necessário, pode-se definir domínios federativos para se analisar o fluxo de informação entre um conjunto de sistemas.

5. Controle do fluxo de informações de um sistema para domínios específicos: esse serviço requer, além dos subgrupos *ftpcli* e *ftpserv* na MIB do sistema que se deseja gerenciar, a inserção do subgrupo *ftpext* para manter as informações sobre as transações que fluem para os domínios especificados.

A coleta das informações de gerenciamento para o grupo *ftp* será realizada nos sistemas onde estão instalados os clientes e servidores FTP que estão sendo gerenciados. Como pode ser visto na descrição do protocolo FTP, no Apêndice A, há dois tipos de transferência de arquivos:

1. Entre um cliente e um servidor FTP: há duas conexões entre os sistemas do cliente e servidor FTP, uma para comandos e outra para transferência dos arquivos. Nesse caso, as informações de gerencimento são computadas nos subgrupos *ftpcli* e *ftpserv* dos sistemas do cliente e servidor FTP, respectivamente.

   A Figura 4.5 ilustra o processo de coleta de dados para esse tipo de transferência de arquivos.



Legenda:

Arquivo 1 - Dados fornecidos pelo Cliente FTP sobre as conexões de controle e de Dados.

Arquivo 2 - Dados fornecidos pelo Servidor FTP sobre as conexões de controle e de dados.

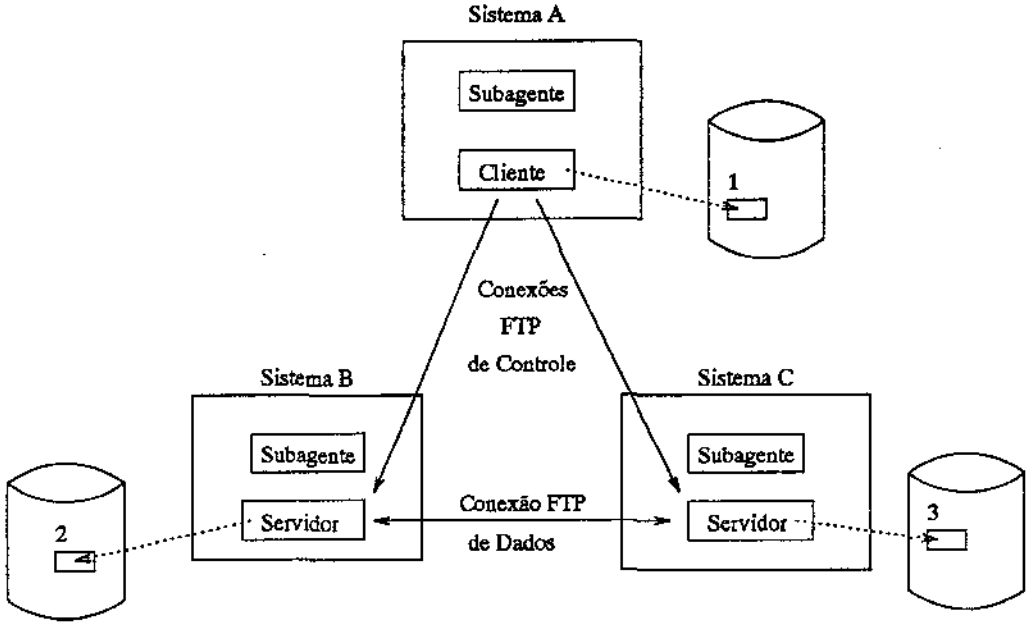
Figura 4.5: Tansferência de Arquivos Cliente-Servidor: Coleta de Dados

2. Entre dois servidores de arquivos: nesse tipo de transferência um cliente abre duas conexões para comandos FTP e programa a transferência de arquivos entre dois servidores. As informações das conexões de controle são computadas nas bases do cliente (*ftpcli*) e dos dois servidores envolvidos (*ftpserv*), enquanto a conexão de dados fornece informações apenas para as bases dos servidores.

   A Figura 4.6 ilustra o processo de coleta de dados para esse tipo de transferência de arquivos.

Os dados para os subgrupos *ftpman* e *ftpext* são fornecidos pelos sistemas onde estão os clientes e servidores FTP. Maiores detalhes sobre a coleta de dados são apresentados no próximo capítulo.

Uma vez definidos os objetos que compõe o grupo *ftp* na MIB, é necessário definir um agente ou subagente para manter esses objetos e atender as solicitações dos gerentes.



Legenda:

Arquivo 1 - Dados fornecidos pelo Cliente FTP sobre as conexões de controle.

Arquivos 2 e 3 - Dados fornecidos pelos Servidores FTP sobre as conexões de controle e de dados.

Figura 4.6: Tansferência de Arquivos Servidor-Servidor: Coleta de Dados

4.3.2 Subagente FTP

A fim de tornar a implementação do gerenciamento do protocolo FTP mais modular e possibilitar o acesso às informações gerenciadas por outros sistemas de gerenciamento, como visto no Capítulo 3, optou-se pela implementação de um subagente FTP. O subagente FTP é encarregado apenas da manutenção do grupo de objetos *ftp* definido na seção anterior.

O subagente FTP possui duas funções básicas:

- estabelecer a comunicação com o agente SNMP para exportar seus objetos e permitir que o gerente os acesse para consulta ou alteração; e
- instanciar os objetos do grupo *ftp* gerenciados em cada sistema. O subagente associa cada objeto definido em sua MIB com seus respectivos valores nos arquivos de dados.

O subagente FTP deverá ser instalado em todos os sistemas onde estejam gerenciando o protocolo FTP, ou seja, em todos os sistemas onde seja mantido pelo menos um dos subgrupos do grupo *ftp*. O agente SNMP, a quem o subagente exportará sua MIB, pode estar instalado no mesmo sistema que este ou em um sistema remoto. O subagente FTP deve ser instalado no sistema que contenha um gerente FTP, quando for necessário oferecer o gerenciamento em domínios. Neste caso, o subagente mantém as informações do subgrupo *ftpman* e de outros subgrupos, para o acesso de um gerente de um domímio superior.

O subagente FTP usa o protocolo SMUX para se comunicar com um agente SNMP. O subagente abre uma conexão SMUX com o agente SNMP e exporta os objetos pelo qual ele é responsável. Quando o agente SNMP recebe alguma solicitação envolvendo variáveis da subárvore exportada pelo subagente FTP, o agente envia uma mensagem SMUX para o subagente contendo apenas essas variáveis. O subagente processa a operação desejada e envia a respota para o agente SNMP. O agente SNMP compõe a resposta do subagente FTP com o resultado do processamento local e envia para o gerente responsável pela solicitação.

Como mostra a Figura 4.7, o agente SNMP funciona como um intermediário na comunicação entre o subagente FTP e um gerente que esteja oferencendo serviços de gerenciamento do protocolo FTP.

4.3.3 Serviços de Gerenciamento do Protocolo FTP

Esta seção apresenta os serviços de gerenciamento que podem ser oferecidos a apartir do grupo de objetos *ftp* definido para compor a MIB na seção 4.3.1 e mantido pelo subagente FTP definido na seção anterior.

O gerenciamento do protocolo FTP possibilita a obtenção de estatísticas sobre o fluxo de dados gerado por este protocolo em uma rede, isto é, este gerenciamento permite a

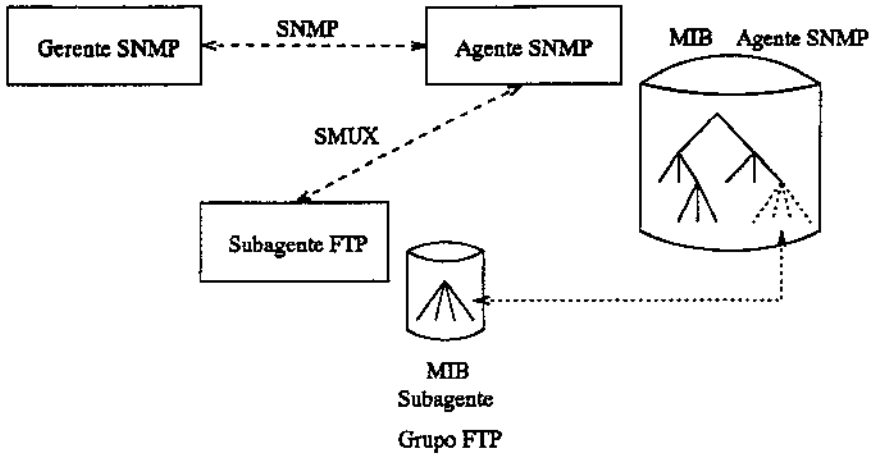


Figura 4.7: Comunicação Gerente - Subagente FTP

obtenção de uma estimativa real do consumo de recursos de rede devido à transferência de arquivos possibilitando assim uma melhor alocação de recursos como linhas e servidores.

Usando as informações definidas para o grupo *ftp* e armazenadas na MIB é possível obter uma série de estatísticas que são descritas a seguir por subgrupo.

1. A partir do subgrupo *ftpcli*, instalado em cada sistema gerenciado que suporte o protocolo FTP, pode-se obter estatísticas sobre:
   - número de conexões FTP abertas;
   - tempo médio de conexão aberta;
   - número de mensagens enviadas e recebidas pelo sistema, incluindo mensagens de erro;
   - número de *bytes* enviados e recebidos; e
   - número de arquivos enviados e recebidos.
2. A partir do subgrupo *ftpserv*, pode-se obter estatísticas dos servidores de arquivos disponíveis na rede, como:
   - número de conexões FTP abertas em cada servidor;
   - tempo médio de conexão aberta;
   - número de mensagens enviadas e recebidas pelos servidores, incluindo mensagens de erro;
   - número de *bytes* enviados e recebidos;
   - número de arquivos transmitidos;
   - arquivos mais acessados nos servidores; e

- lista dos usuários que acessaram recentemente um arquivo. Com essa informação é possível enviar mensagens automáticas aos usuários, que acessaram recentemente um arquivo, para informá-los da disponibilidade de uma nova versão do arquivo, por exemplo.

3. A partir do subgrupo *ftpman*, pode-se obter estatísticas do fluxo de mensagens entre os domínios, isto é, informacões sobre as conexões FTP que envolvem uma entidade interna e uma externa ao domínio do gerente FTP. Pode-se obter as seguintes informações sobre as conexões FTP:

   - número de conexões FTP abertas envolvendo um sistema externo ao domínio do gerente;
   - tempo médio de conexão aberta;
   - número de mensagens trocadas com os sistemas externos, incluindo mensagens de erro;
   - número de *bytes* enviados e recebidos; e
   - número de arquivos transferidos.

   Alem dessas informações, a partir do subgrupo *ftpman* pode-se obter informações sobre os servidores de arquivo pertencentes ao domínio do gerente FTP.

4. A partir do subgrupo *ftpext*, pode-se obter estatísticas sobre o fluxo de dados gerado pelo protocolo FTP entre dois sistemas específicos. Pode-se obter as seguintes informações:

   - número de conexões FTP abertas entre os dois sistemas;
   - tempo médio de conexão aberta;
   - número de mensagens trocadas entre os sistemas em questão, incluindo mensagens de erro;
   - número de *bytes* trocados entre os sistemas; e
   - número de arquivos trocados entre os sistemas.

Serviços mais sofisticados podem ser definidos a partir das estatísticas coletadas em cada subgrupo. Pode-se, por exemplo, compor relatórios com as seguintes informações:

1. melhores períodos para acessar um servidor de arquivos: com os horários que o servidor FTP está mais livre, ou seja, com menor número de conexões;

2. servidores mais acessados e arquivos mais requisitados em cada servidor;

3. fluxo de dados diário entre dois sistemas específicos, ou que ultrapassem os domínios de um gerente; e

4. clientes que geram o maior fluxo de dados na rede.

Muitos outros serviços podem ser oferecidos usando os objetos disponíveis ou definindo novos objetos e expandindo o grupo *ftp*.

4.4 Considerações Finais

A definição dos objetos que compõem o grupo *ftp* na MIB foi feita levando-se em consideração os objetos típicos que compõem os outros grupos da MIB [MR91b]. Além disso, buscou-se um conjunto mínimo de objetos que fornecesse uma estimativa mais realista sobre o fluxo de dados gerado pelo protocolo FTP. Alguns objetos foram definidos para que se pudesse oferecer alguns serviços de interesse geral, como por exemplo foi necessária a criação das tabelas de servidores de arquivos, clientes FTP e gerentes FTP no grupo *ftpman* para que se pudesse classificar as associações FTP como internas ou externas a um domínio.

O grupo de objetos *ftp* pode ser expandido caso se identifique a necessidade de oferecer novos serviços que não estão previstos nesse protótipo e que envolvam dados ainda não controlados.

Alterações no grupo *ftp* envolveram adaptações do subagente para que este controle os novos objetos que possam vir a ser inseridos na MIB, mas não irão afetar a parte de comunicação entre o subagente e o agente SNMP.

O gerenciamento de outros protocolos de aplicação, como SMTP e Telnet, pode ser feito seguindo o mesmo esquema de domínios e de controle do fluxo de dados entre os mesmos. Para gerenciar novos protocolos é necessário definir os grupos de objetos que serão inseridos na MIB, implementar um subagente para manter esses objetos disponíveis e novos serviços de gerenciamento. O grupo *ftp* e o subagente FTP podem ser utilizados como ponto de partida para a implementação do gerenciamento de novos protocolos de aplicação.

O subagente FTP pode exportar o grupo de objetos *ftp* para a MIB de qualquer agente SNMP que suporte o protocolo SMUX. Assim sendo, o subagente FTP pode ser utilizado com um agente do ISODE ou do *NetView*, analisados no capítulo anterior. O gerenciamento do subagente FTP, a partir do *SunNet Manager* envolve o mapeamento do grupo *ftp* da MIB para arquivos ASCII e o uso de um *SNMP proxy agent* e do agente SNMP do ISODE ou do *Netview*.

Caso se queira gerenciar o protocolo FTP a partir de qualquer um dos sistemas analisados no capítulo anterior, é necessário implementar os serviços de gerenciamento descritos na seção anterior nos três sistemas. Independente dos serviços automáticos, é possivel

acessar o grupo *ftp* da MIB para consultas a partir de qualquer um dos gerentes analisados.

O proximo capítulo apresenta a implementação de um protótipo para o gerencimento do protocolo FTP desenvolvida no ISODE de acordo com o esquema definido nesse capítulo.

Capítulo 5

Implementação do Modelo de Gerenciamento Proposto

5.1 Introdução

Neste capítulo é apresentado o protótipo para o gerenciamento do protocolo FTP que foi desenvolvido de acordo com o esquema proposto no capítulo anterior.

O ISODE foi utilizado como plataforma de desenvolvimento para o protótipo de gerenciamento do protocolo FTP. O ISODE foi escolhido dentre os sistemas disponíveis, principalmente, devido à possibilidade de acesso e modificação de seus fontes para fins acadêmicos, o que facilitou a implementação de módulos adicionais.

O ISODE contém a definição da MIB II em ASN.1, e a implementação de um agente SNMP, do protocolo SMUX, de um subagente SMUX *o unixd*, e de uma interface simples para emissão de comandos SNMP e recepção das respostas.

Para o gerenciamento do protocolo FTP foi necessário:

- definir o grupo *ftp* em ASN.1 e compilá-lo usando o compilador *mosy*;
- implementar um subagente FTP que associe os objetos do grupo *ftp* com os arquivos de dados, para instanciá-los. Além disso, o subagente é responsável pela comunicação com um agente SNMP; e
- implementar o gerente FTP que forneça ao administrador uma série de serviços automáticos e uma interface para requisição de estatísticas do protocolos FTP em cada sistema gerenciado.

O protótipo implementado contém a definição e instalação do grupo *ftp*, o subagente FTP e uma interface para emissão de consultas às informações de gerenciamento do protocolo FTP.

A próxima seção apresenta como foi implementado o grupo *ftp* de objetos para a MIB. As seções 5.3 e 5.4 mostram, respectivamente, o processo de implementação do subagente FTP e de alguns serviços de gerenciamento do protocolo FTP. Na seção 5.5 são apresentadas algumas considerações sobre a implementação do protótipo.

5.2 Grupo *ftp*

O grupo *ftp* foi dividido em 4 subgrupos (*ftpcli, ftpserv, ftpman e ftpext*) conforme especificado no capítulo anterior. Cada subgrupo do protocolo FTP foi definido em ASN.1 em um arquivo separado (*ftpcli.my, ftpserv.my, ftpman.my e ftpext.my*), de modo que possam ser instalados de acordo com as necessidades de gerenciamento de cada sistema. A definição do grupo *ftp* e seus subgrupos em ASN.1 é apresentada no Apêndice B.

Os arquivos ASN.1 do grupo *ftp* e de seus subgrupos foram compilados no compilador *mosy* do ISODE e inseridos na subárvore *enterprises* do grupo *private* da MIB, como mostra a Figura 5.1.

O grupo *ftp* foi inserido na subárvore *enterprises* porque seu gerenciamento é um experimento acadêmico não cadastrado pela *Internet*, ou seja uma iniciativa particular. Desse modo, o grupo *ftp* não pode ser inserido nas subárvores *mgmt* e *experimental*.

Os objetos do grupo *ftp* são instanciados com valores armazenados em arquivos de configuração e de dados.

A coleta das informações de gerenciamento, que serão armazenadas no grupo *ftp*, pode ser realizada de duas maneiras:

1. A partir dos códigos fontes de clientes e servidores FTP: nesse método os códigos responsáveis pela coleta de dados de gerenciamento são embutidos no próprio código dos clientes e servidores FTP.

   O gerenciamento do protocolo FTP é feito fim-a-fim, isto é, no esquema proposto deseja-se controlar o fluxo de informações entre os clientes e servidores FTP. Assim, pode-se coletar todos os dados de interesse a partir dessas duas entidades. Esse processo de coleta de dados é semelhante ao utilizado no gerenciamento dos protocolos das camadas inferiores (TCP, IP, ...), ou seja, para os protocolos das camadas inferiores, os dados são coletados no *Kernel* do Unix e os valores estatísticos são armazenados em variáveis do *Kernel*. Como os protocolos de aplicação não fazem parte do *Kernel* do Unix, os dados coletados devem ser armazenados em arquivos de dados.

   A desvantagem desse método de coleta de dados é a necessidade do acesso aos códigos fontes dos clientes e servidores FTP instalados nos sistemas que se deseja gerenciar, ou de se instalar nos sistemas gerenciados um cliente e/ou servidor FTP adaptado para realizar a coleta de dados. A principal vantagem desse método de

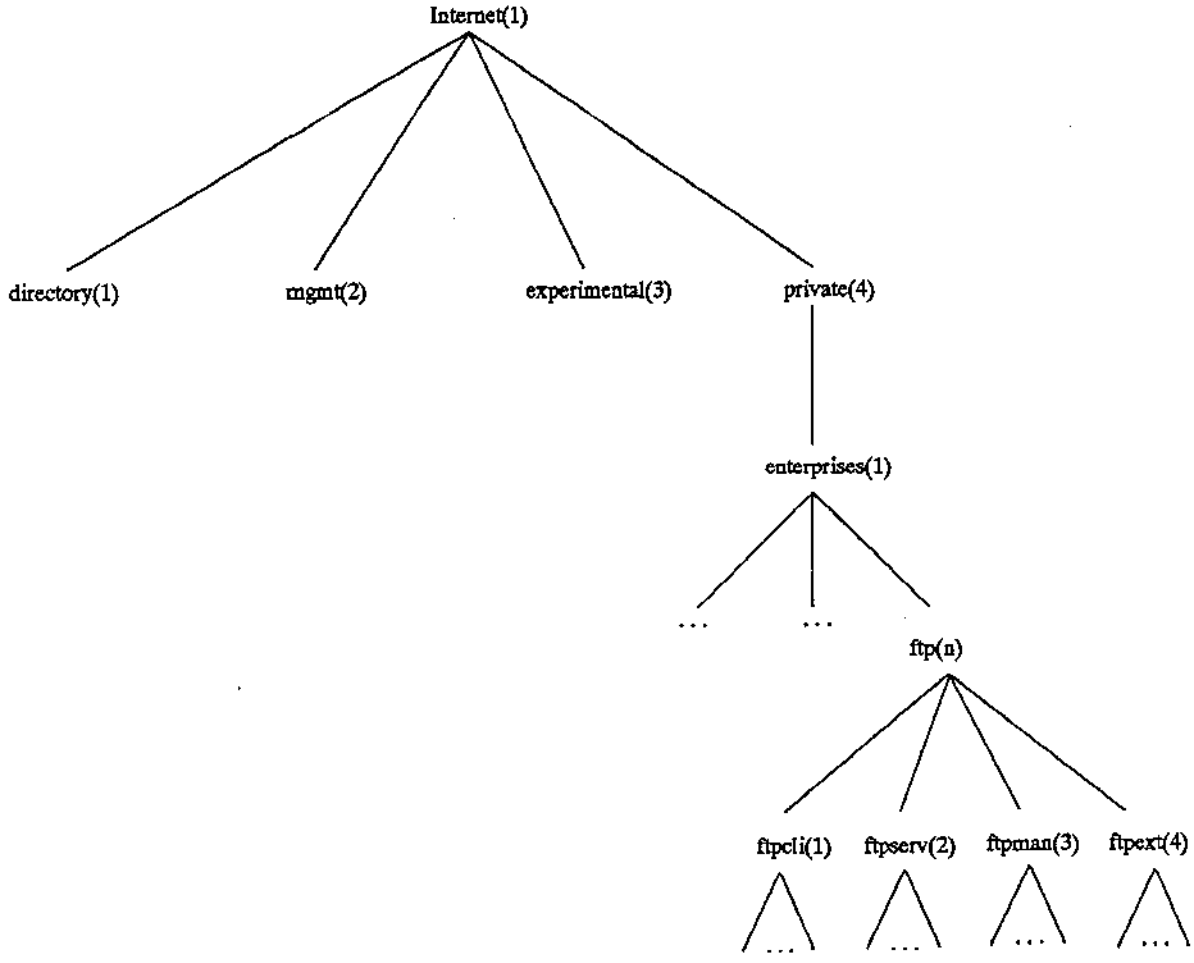


Figura 5.1: Grupo *ftp* na MIB

coleta de dados é possibilitar um maior refinamento dos serviços oferecidos, uma vez que pode-se obter qualquer tipo de informação sobre as transações FTP a partir dos códigos fontes dos clientes e servidores FTP.

2. A partir dos pacotes que estão circulando na rede: nesse método os pacotes em trânsito na rede são capturados por um programa de coleta de dados [Sil]. Cada pacote é aberto e analisado, caso seja um pacote do protocolo FTP, este deve ser contabilizado, caso contrário deve ser descartado. O programa coletor de dados deve ser instalado em sistemas que estejam localizados em pontos estratégicos da rede. A principal vantagem desse sistema é que ele pode ser facilmente adaptado para a coleta de informações de outros tipos de protocolos de aplicação, como por exemplo o Telnet e o SMTP. O problema desse método de coleta de dados é a possibilidade de causar atraso na comunicação pela captura de todos os pacotes que circulam na rede ou perda de informações pelo acúmulo de pacotes para análise, quando o fluxo de dados na rede for muito intenso. O uso desse método de coleta de dados é recomendável quando se deseja informações genéricas sobre a origem do tráfego na rede. A coleta de informações mais detalhadas sobre as transações FTP, como as informações descritas no Capítulo 4, torna-se praticamente inviável a partir desse método.

Para fins experimentais, no protótipo do sistema de gerenciamento do protocolo FTP, foi adotada a coleta de dados a partir dos códigos de um cliente e servidor FTP. Para uso em redes reais, entretanto, é inviável adaptar todos os clientes e servidores FTP disponíveis ou exigir que todos passem a adotar o cliente e servidor adaptado para a coleta de dados. Dessa maneira, a instalação de programas de coleta de dados em pontos estratégicos da rede torna-se uma opção mais adequada para situações reais, embora não permita oferecer todos os serviços descritos no capítulo anterior com o nível de refinamento especificado.

É importante ressaltar que o esquema de gerenciamento do protocolo FTP definido no Capítulo 4 independe do método de coleta de dados adotado para manter as informações atualizadas. O fundamental é que as informações sobre o protocolo FTP sejam corretamente armazenadas nos arquivos de dados para que o subagente possa acessá-las.

Para coletar as informações necessárias para o gerenciamento do protocolo FTP, no protótipo, inicialmente foram alterados os códigos fontes de um servidor de arquivos *Wuarchive* [Mye] desenvolvido na *Washington University in Saint Louis*, e que utiliza o protocolo FTP.

As informações necessárias para o gerenciamento do protocolo FTP são coletadas em duas partes do servidor:

- no interpretador de comandos (*ftpcmd.y*): onde são coletadas as informações sobre o número de mensagens e *bytes* que são recebidos pelo servidor de arquivos; e

- no *daemon* do servidor de arquivos *ftpd.c*: onde são coletadas as informações sobre os arquivos transferidos e sobre o número de mensagens e *bytes* enviados pelo servidor.

As estatísticas do subgrupo *ftpserv* começam a ser computadas quando o servidor recebe uma solicitação de conexão; todos os pacotes FTP recebidos e enviados são contabilizados. Quando um arquivo é transferido, lido ou gravado no servidor, as informações sobre o arquivo são atualizadas imediatamente na tabela que contém os dados relativos a cada arquivo disponível no servidor de arquivos, tabela *ftpservFileTable*. Quando uma conexão é fechada todas as estatísticas coletadas são atualizadas no arquivo de dados do servidor. Além disso, um arquivo de histórico é mantido com o nome e endereço do sistema remoto que solicitou a conexão e com as estatísticas do fluxo de dados gerados nesta. Essas informações serão utilizadas na atualizção dos objetos dos subgrupos *ftpman* e *ftpext*, isto é, nas bases extra e do gerente.

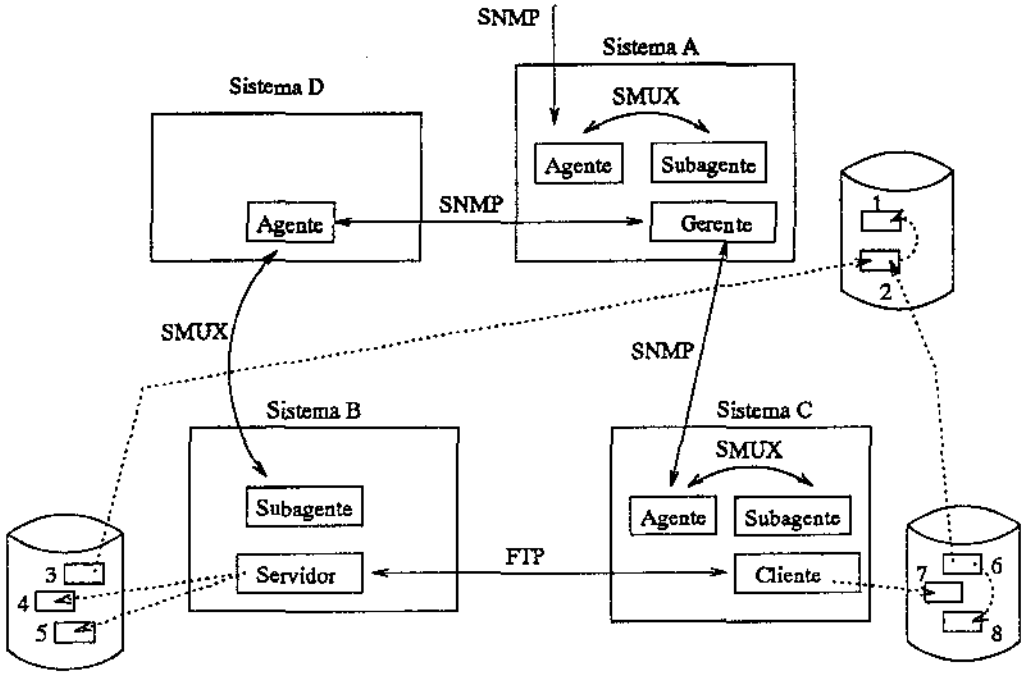
Do mesmo modo, um cliente FTP deve ser adaptado para fornecer as informações relativas às conexões FTP que ele solicita, ou seja, as informações do subgrupo *ftpcli*. O processo é semelhante ao do servidor de arquivos, isto é, quando um cliente solicita uma conexão, as informações começam a ser computadas e quando a conexão é fechada, todos os dados coletados são armazenados em dois arquivos, o arquivo de dados do cliente e o histórico de transações do cliente. No arquivo de dados são atualizadas as estatísticas descritas no Capítulo 4, e no histórico são armazenados: nome e endereço do servidor que foi acessado e as estatísticas referentes a essa conexão. Esses dados serão coletados para atualização das bases do gerente e extra.

Quando é necessário manter as informações sobre o fluxo de dados gerado pelo protocolo FTP entre dois sistemas específicos, um arquivo de configuração é instalado nos sistemas do cliente e/ou servidor com o nome e endereço do *host* remoto. Um programa de coleta de dados para a base extra é executado periodicamente e verifica o arquivo de histórico do cliente/servidor, para identificar possíveis conexões com o sistema especificado. Quando é encontrada uma entrada no histórico correspondendo ao sistema remoto predefinido, o arquivo de dados do grupo *ftpext* é atualizado.

Periodicamente, as informações do histórico são consultadas por um coletor de dados do gerente que atualiza os dados do grupo *ftpman*. Cada entrada do histórico é analisada para verificar se esta envolve duas entidades de diferentes domínios. Quando as duas entidades, o cliente e o servidor, pertencerem ao domínio do gerente, a entrada é descartada, caso contrário os dados são armazenados no arquivo de dados do gerente e uma nova entrada é gravada no histórico do gerente. Essas informações serão consultadas por um gerente de nível mas alto. Dessa forma, é possível manter as informações sobre o fluxo de dados entre domínios.

A Figura 5.2 ilustra o processo de coleta de dados para os principais objetos do grupo *ftp*. Neste exemplo, o sistema A contém um gerente FTP e um subagente FTP que mantém as informações do subgrupo *ftpman*; o sistema B contém os programas de um

servidor de arquivos e um subagente que mantém os objetos do grupo *ftpserv*; o sistema C contém um cliente FTP e um subagente que mantém as informações dos subgrupos *ftpcli* e *ftpext*, uma vez que no exemplo foi solicitada a instalação de uma base extra em C; e o sistema D contém apenas um agente SNMP, para quem o subagente do sistema B exporta seus objetos. Além dos programas descritos acima, os sistemas A e C contêm um agente SNMP para quem os subagentes FTP exportam seus objetos. A descrição dos arquivos de dados é apresentada a seguir.



Arquivos: 1. ftpman.dat
2. ftpman.log
3. ftpserv.log
4. ftpserv.dat
5. ftpfiles.dat
6. ftpcli.log
7. ftpcli.dat
8. ftpext.dat

Figura 5.2: Processo de Coleta de Dados para o Grupo *ftp*

Como o volume de informações gerenciadas é bastante reduzido, no protótipo não foi adotado um sistema de banco de dados para armazená-las, foram utilizados arquivos de dados comuns.

Os arquivos de dados utilizados para armazenamento das informações dos objetos do grupo *ftp* são os seguintes:

- *ftpd.rc*: arquivo de configuração que define os subgrupos que são mantidos pelo suba-

gente FTP, e a descrição dos programas do cliente, servidor e/ou gerente instalados no sistema;

- *ftpcli.dat*: mantém as informações do subgrupo *ftpcli* da MIB, menos a descrição do programa;
- *ftpcli.log*: histórico das transações FTP solicitadas pelo cliente, incluindo o nome e endereço do servidor que foi acessado em cada transação;
- *ftpserv.dat*: contém as informações sobre o subgrupo *ftpserv*, com exceção da descrição do programa e da tabela de arquivos disponíveis;
- *ftpfile.dat*: arquivo que contém a descrição de cada arquivo disponível no servidor e todas suas informações, incluindo a lista dos últimos usuários que o acessaram;
- *ftpserv.log*: nome e endereço da entidade remota e dados de cada conexão com o servidor FTP;
- *ftpman.dat*: mantém as informações referentes ao subgrupo *ftpman* da MIB, com exceção da descrição do programa utilizado e das tabelas dos clientes, servidores e gerentes do domínio do gerente em questão;
- *ftpman.log*: histórico das conexões FTP que envolvem entidades externas ao domínio do gerente FTP;
- *ftpmanserv.dat*: contém a tabela dos servidores FTP que estão no domínio do gerente FTP;
- *ftpmancli.dat*: contém a tabela dos clientes FTP disponíveis no domínio do gerente FTP;
- *ftpmanman.dat*: contém a tabela dos gerentes FTP sob o controle desse gerente;
- *ftpext.dat*: contém as informações sobre as conexões entre o sistema local e um sistema remoto predefinido.

A associação entre os objetos do grupo *ftp* e seus valores armazenados nos arquivos de dados é realizada pelo subagente FTP.

5.3 Subagente FTP

O subagente FTP é responsável pela manutenção de diferentes subgrupos do grupo *ftp*, dependendo de sua configuração e das necessidades de gerenciamento do sistema onde este é instalado. Cada subagente exporta seu grupo *ftp* para um único agente SNMP, e

da mesma maneira cada agente SNMP só se associa a um subagente FTP. A separação do subagente e do agente se deve principalmente ao interesse de se ter um sistema de gerenciamento para o protocolo FTP modular. Assim o subagente FTP pode exportar seus objetos para agentes SNMP desenvolvidos por diferentes fabricantes, desde que estes suportem o protocolo SMUX.

Para que o agente SNMP do ISODE reconheça um subagente FTP é necessário que haja uma entrada no arquivo de configuração dos subagentes ou *SMUX peers*, o arquivo *snmpd.peers*, com os seguintes dados do subagente FTP:

1. nome do subagente, *ftpsubd*;
2. identificador do grupo *ftp* na MIB: 1.3.6.1.4.1.n;
3. senha de acesso, para o subagente FTP obter êxito na comunicação com o agente SNMP; e
4. prioridade de processamento, que é um campo opcional.

O subagente FTP possui duas funções principais:

- comunicação com o agente SNMP, envolvendo a recepção, a interpretação e o envio de mensagens SMUX; e
- instanciação dos objetos do grupo *ftp*, ou seja, associação dos objetos mantidos por ele com os seus respectivos valores armazenados nos arquivos de dados e de configuração descritos na seção anterior. A associação de cada subgrupo *ftpcli, ftpserv, ftpman e ftpext* com seus arquivos de dados foi implementada em separado, facilitando assim sua instalação. O código de cada subgrupo é instalado de acordo com a configuração do subagente.

O subagente FTP executa os seguintes passos quando é ativado, ou seja, em sua fase de configuração e estabelecimento de associação com o agente SNMP:

1. montar o grupo *ftp*: o subagente FTP monta sua MIB a partir de seu arquivo de configuração, que contém os subgrupos que devem ser instalados, e dos arquivos de objetos de cada subgrupo definidos em ASN.1 e compilados pelo *mosy*;
2. associar cada objeto com sua instância no arquivo de dados: cada objeto é associado a um procedimento de acesso que localiza seu valor no arquivo de dados para leitura ou gravação quando necessário. Todos os objetos simples, não tabulares, de um grupo são associados ao mesmo procedimento de acesso. Todos os objetos de uma tabela são associados a um procedimento de acesso à tabela que eles pertencem. Os procedimentos de acesso são acionados uma única vez para cada consulta que use algum de seus objetos;

3. ativar o protocolo SMUX: como a comunicação com o agente SNMP é realizada com o uso do protocolo SMUX, este deve ser ativado para o estabelecimento de uma associação. O estabelecimento de uma associação SMUX deve ser solicitado pelo subagente através de uma mensagem de *open* do SMUX; e

4. registrar seus objetos no agente SNMP: o subagente envia uma mensagem de *register-request* para registrar seus objetos na MIB do agente SNMP e aguarda uma mensagem de *register-response* do agente confirmando o registo de cada objetos.

Quando o subagente FTP termina esta fase inicial de execução, ele entra em um processo de espera das solicitações, que são repassadas pelo agente SNMP sempre que um gerente desejar informações do grupo de objetos mantidos pelo subagente. Nesta fase o subagente aceita as seguintes primitivas SMUX:

- *get-request, get-next-request e set-request*: que são similares às primitivas do protocolo SNMP. Quando o agente SNMP recebe uma solicitação de um gerente que envolve objetos do grupo *ftp* cadastrado em sua MIB pelo subagente FTP, ele separa esses objetos e envia uma primitiva SMUX equivalente para o subagente. Enquanto o subagente resolve seu pedido, o agente faz o processamento local da solicitação sobre seus objetos. Quando o subagente recebe uma dessas primitivas do agente SNMP, ele ativa os procedimentos de acesso a cada variável envolvida na solicitação. Se todos os procedimentos retornam com sucesso, o subagente compõe uma mensagem SMUX de *get-response* com os valores de cada variável solicitada e envia para o agente SNMP. Se o processamento falha para uma das variáveis o subagente monta uma mensagem de *get-response* de erro e envia para o agente SNMP. O agente SNMP compõe a resposta enviada pelo subagente com o resultado do processamento local, e se tudo foi executado com sucesso a operação é efetivada, no caso do *set-request*, e um *SNMP get-response* com os valores requisitados é enviado para o gerente. Se o processamento de uma variável solicitada falhar no agente ou no subagente, o agente SNMP envia um *SNMP get-response* com informações sobre o erro para o gerente e a operação é cancelada;

- *commit-or-rollback*: primitiva enviada pelo agente SNMP para efetivar ou cancelar uma operação de *set* em um conjunto de variáveis do grupo *ftp*. A operação de *set* é atômica, ou seja, deve ser efetivada se todas as variáveis, da MIB do agente SNMP e dos subgrupos exportados pelos subagentes, contidas no *SNMP set-request* enviado pelo gerente puderem ter seus valores alterados por este. O agente SNMP reune as respostas de todos os subagentes envolvidos com o resultado de seu processamento local e se tudo foi processado com sucesso, a operação é efetivada, caso contrário, a operação é cancelada.

- *close*: primitiva enviada pelo agente SNMP quando ele deseja terminar uma associação SMUX. O subagente FTP deve então encerrar o seu processamento.

Qualquer outra mensagem recebida pelo subagente FTP é tratada como erro e descartada.

O subagente FTP pode, a qualquer momento, enviar uma mensagem de *SMUX close* para terminar a associação com o agente SNMP. Normalmente essa mensagem é enviada quando há algum erro no processamento do subagente que não pode ser recuperado ou quando o subagente é desativado.

O subagente FTP envia mensagens de *SMUX trap* para o agente SNMP que as converte para mensagens de *SNMP trap* e envia para o gerente responsável. O subagente FTP está programado para enviar um *trap* quando uma nova versão de um arquivo é armazenado em um servidor de arquivos. O subagente identifica a existência de uma nova versão de um arquivo através do campo de estado do arquivo armazenado na tabela de arquivos do servidor. Se o campo estado indicar uma nova versão o subagente envia um *New-Version trap* para o agente SNMP, que irá transmiti-lo para o gerente FTP.

A Figura 5.3 exemplifica a troca de mensagens SMUX entre um subagente FTP e um agente SNMP e a correspondência dessas mensagens com as mensagens SNMP usadas na comunicação do agente SNMP com o gerente FTP.

5.4 Gerente FTP

O gerente FTP não tem conhecimento da existência de um subagente FTP que mantém apenas as informações do grupo *ftp* da MIB. O gerente considera que esse grupo de objetos faz parte da MIB do agente SNMP, e realiza toda a comunicação com este agente, enviando suas requisições para ele e recebendo as respostas e os *traps* a partir dele.

Com as informações coletadas das bases dos agentes SNMP, o gerente FTP identifica os servidores, clientes e gerentes FTP que estão localizados no seu domínio, ou seja, que estão sob sua administração.

Para o gerente FTP oferecer uma série de serviços simples, este deve ter uma interface amigável. Através dessa interface o administrador da rede poderá enviar requisições e analisar as respostas, ou melhor, o administrador poderá obter as informações armazenadas em cada subgrupo da MIB dos sistemas sob seu domínio. Dessa maneira, o gerente FTP poderá oferecer ao administrador da rede todos os serviços estatísticos relacionados na seção 4.3.3.

Uma série de serviços de gerenciamento do protocolo FTP podem ser realizados pelo gerente FTP. Por exemplo, o gerente FTP pode ser programado para, periodicamente, consultar os agentes SNMP e analisar as informações do grupo *ftp*, a fim de montar relatórios estatísticos. Os seguintes relatórios estatísticos podem ser montados pelo gerente FTP:

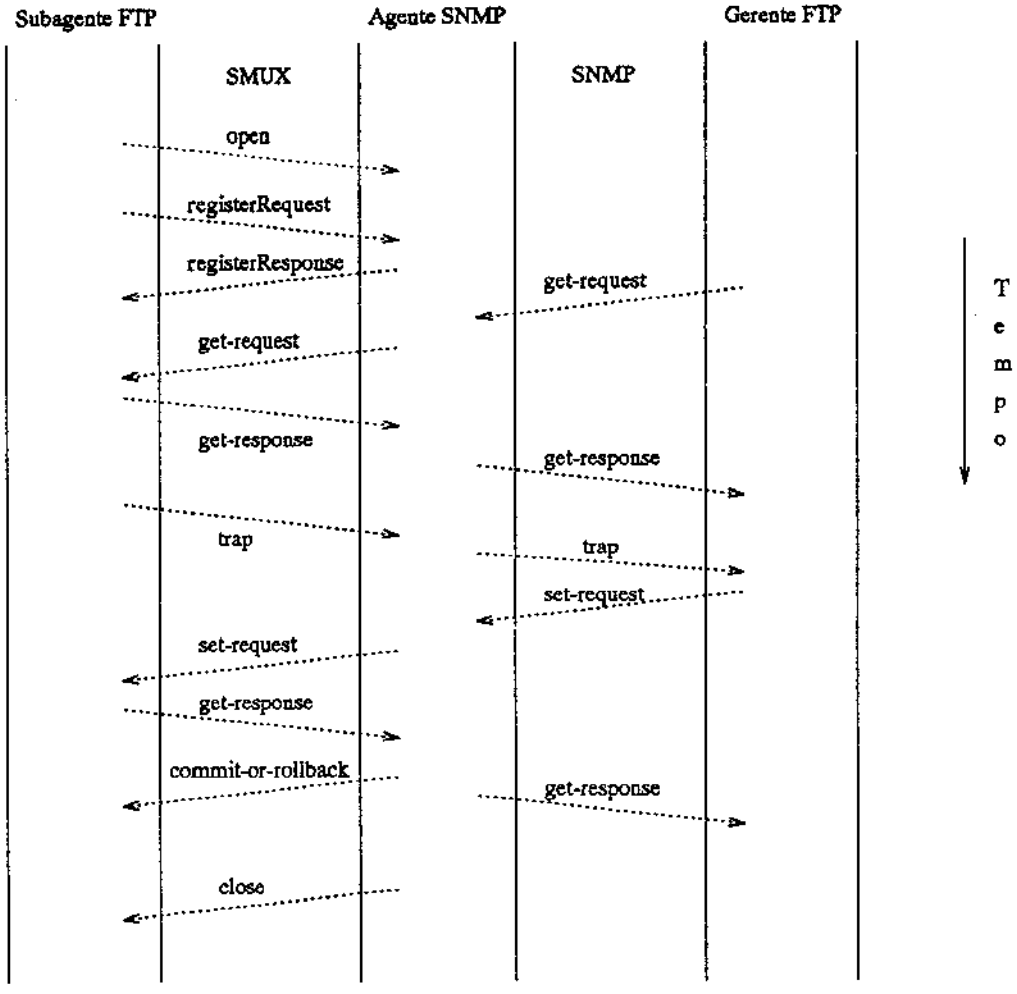


Figura 5.3: Exemplo de Troca de Mensagens entre um Subagente FTP, um Agente SNMP e um Gerente FTP

- relatórios dos períodos de menor fluxo: a cada hora o gerente coleta as informações dos servidores de seu domínio, como número de conexões, pacotes, *bytes* e arquivos transferidos, e monta um relatório com os melhores períodos do dia para se acessar os servidores de seu domínio e os períodos onde o tráfego é mais intenso;

- relatórios estatísticos sobre os servidores: periodicamente o gerente coleta as informações de cada servidor de arquivo e monta um relatório com os servidores e arquivos mais acessados. Essa informação pode ser utilizada para fazer uma melhor alocação de linhas de acesso aos servidores e a redistribuição dos arquivos disponíveis entre os servidores de seu domínio;

- relatórios sobre o fluxo de informações FTP entre dois sistemas: o gerente coleta as informações necessárias a partir do grupo *ftpext*, base extra, instalado em um dos sistemas especificados e monta um relatório com essas estatísticas por períodos predeterminados como, por exemplo, por dia; e

- relatórios estatísticos com informações sobre os clientes FTP de seu domínio: o gerente coleta as informações sobre os clientes de seu domínio periodicamente e monta relatórios dos clientes que geram maior tráfego FTP na rede.

Um outro serviço muito útil que pode ser oferecido pelo gerente FTP é o envio de mensagens automáticas aos usuários que acessaram recentemente um arquivo. Essas mensagens serão enviadas quando uma nova versão do arquivo se torna disponível no servidor onde o usuário o havia lido.

Para oferecer esse serviço o gerente FTP deve, periodicamente, verificar o estado de cada arquivo na tabela de arquivos dos servidores. Quando o estado do arquivo indicar uma nova versão, o gerente deve recuperar o endereço eletrônico dos últimos usuários que acessaram o arquivo, que é armazenada juntamente com a entrada de cada arquivo no tabela de arquivos do servidor, e enviar uma mensagem para cada endereço válido informando que há uma nova versão do arquivo disponível. Quando um novo arquivo é armazenado no servidor, o endereço de todos os usuários que o acessaram contém o valor "unknown user", que é um endereço inválido para o envio de mensagens automáticas. Após enviar as mensagens automáticas, o gerente FTP deve alterar o valor do estado do arquivo para não enviar mensagens duplicadas aos usuários, através de uma operação de *set*.

Além disso, associada à variavel de estado do arquivo há um *trap* que é enviado pelo subagente quando uma nova versão de um arquivo é armazenada no servidor, conforme foi descrito na seção anterior. Quando o gerente FTP recebe o *New-Version trap*, ele deve decidir se aciona o agente SNMP imediatamente e coleta a lista dos usuários que deverão receber as mensagens de nova versão do arquivo ou se aguarda até que chegue a vez do agente ser consultado, pelo processo de *polling*. Deve-se levar em consideração que quanto

mais tempo o gerente esperar para consultar a lista dos últimos usuários que acessaram o arquivo, maior será o risco de enviar a mensagem automática para usuários que já leram a nova versão do arquivo.

O gerente FTP não foi implementado no protótipo com todas as funcionalidades descritas no capítulo anterior. No protótipo foi implementada apenas uma interface bem simplificada para que um administrador da rede possa coletar as informações de gerenciamento do grupo *ftp*, através de comandos SNMP.

A partir do protótipo implementado é possível oferecer ao administrador da rede o gerenciamento do protocolo FTP em diversos níveis de complexidade:

1. Gerenciamento simples em um sistema: fornece informações simples sobre as transações FTP no sistema, envolve a inserção dos subgrupos *ftpcli* e/ou *ftpserv* na subárvore do *ftp*, a mantenção dos arquivos de dados *ftpcli.dat* e/ou *ftpserv.dat e ftpfile.dat* e a instalação um subagente no sistema em questão para exportar esses grupos.

2. Gerenciamento do protocolo FTP em domínios hierárquicos e federativos: oferece o controle do fluxo de dados entre domínios, conforme definido no Capítulo 4. É necessário que os sistemas de um domínio tenham a seguinte configuração:

   - subagente FTP;
   - grupo *ftpcli* e/ou *ftpserv*, juntamente com os arquivos de dados de cada subgrupo, *ftpcli.dat, ftpserv.dat* e *ftpfile.dat*; e
   - arquivos de log: *ftpcli.log e ftpserv.log.*

   Além disso, o sistema que contenha o gerente FTP deve manter o subgrupo *ftpmam*, todos os arquivos de dados a ele associados (*ftpman.dat, ftpmam.log, ftpmamcli.dat, ftpmanserv.dat e ftpmanman.dat*), um arquivo de histórico, o *ftpmam.log*, e um subagente FTP que estará recebendo as solicitações de um gerente de um nível superior. O subagente se torna necessário pra fazer a comunicação entre dois gerentes de níveis distintos, porque na primeira versão do protocolo SNMP não há a comunicação direta entre dois gerentes, problema que foi resolvido na segunda versão, como foi descrito no Capítulo 2.

3. Gerenciamento do fluxo gerado pelo protocolo FTP entre dois sistemas: esse serviço requer a seguinte configuração em um dos sistemas gerenciados:

   - subgrupo *ftpcli*, arquivos de dados e de histórico relacionados a esse subgrupo, ou seja, *ftpcli.dat e ftpcli.log*, subgrupo *ftpext* e arquivo de dados *ftpext.dat*, e subagente FTP para manter esses subgrupos, caso o sistema em questão não possua um servidor de arquivos FTP.

- subgrupo *ftpserv*, arquivos de dados e histórico relacionados a esse subgrupo, ou seja, *ftpserv.dat, ftpfile.dat e ftpserv.log*, subgrupo *ftpext* e arquivo de dados *ftpext.dat*, e além disso um subagente FTP para exportar esses subgrupos para um agente SNMP, isso se o sistema em questão suportar um servidor de arquivos e não um cliente FTP.
- a união dos itens acima, caso o sistema em questão possui um cliente e um servidor FTP.

5.5 Considerações Finais

A implementação desse protótipo teve como objetivo validar o modelo de gerenciamento do protocolo FTP definido no Capítulo 4. Nesse sentido, o protótipo contém a implementação:

- do grupo *ftp* para a MIB: o que envolve a definição do grupo *ftp*, sua instalação com diferentes configurações, a coleta de dados a partir do servidor FTP. Os dados da base do cliente são simulados, uma vez que o código do cliente FTP não foi alterado. A coleta de dados do cliente é similar à coleta de dados no servidor FTP. Embora a coleta de dados tenha sido embutida nos códigos de um cliente e servidor FTP, este método pode ser alterado sem danos ao modelo de gerenciamento proposto, desde que os dados sejam corretamente armazenados nos arquivos de dados;
- do subagente FTP: o subagente foi completamente implementado, fazendo a comunicação com o agente SNMP e permitindo o acesso aos objetos de cada grupo; e
- de uma interface de comandos simples para oferecer alguns serviços estatísticos ao admistrador da rede, mostrando a viabilidade de se coletar os dados necessários para a emissão de relatórios e o envio de mensagens automáticas, serviços ainda não implementados. Uma vez que os dados necessários para o envio das mensagens automáticas e para emissão de relatórios podem ser obtidos pelo gerente, a confecção dos mesmos pode ser implementada sem muitos problemas.

O protótipo implementado no ISODE contém as características básicas do modelo de gerenciamento do protocolo FTP proposto neste trabalho e a partir dos testes realizados pode-se verificar sua validade e funcionalidade.

De acordo com os testes realizados com o *AIX NetView/6000* e *SunNet Manager*, descridos no Capitulo 3, é possível gerenciar um agente SNMP do ISODE com os gerentes desses dois outros sistemas. Assim sendo, pode-se usar os gerentes do *NetView* e do *SunNet* para gerenciar também o subagente FTP. Com relação ao *SunNet Manager* é

necessário mapear o grupo *ftp* da MIB para arquivos ASCII e inseri-los na MDB. Além disso, deve-se configurar o *SNMP proxy agent* para que este identifique os novos objetos gerenciados pelo agente SNMP/subagente FTP.

Como o subagente FTP foi implementado separadamente, ele pode exportar seu grupo de objetos para um agente SNMP do *AIX NetView/6000*, sendo necessário apenas configurar o agente SNMP do *NetView* para aceitar os objetos no novo subagente e comunicar-se com este através do protocolo SMUX.

No próximo capítulo, é apresentada a conclusão deste trabalho e suas possíveis extensões.

Capítulo 6

Conclusão

O rápido crescimento das redes, junto com a necessidade de mantê-las sempre operacionais e com o tráfego em níveis aceitáveis, impulsionou a adoção de sistemas de gerenciamento de redes. O uso destes sistemas possibilita a detecção de falhas de comunicação e sua resolução, de forma automática ou não, com maior rapidez. Os sistemas de gerenciamento de redes têm facilitado o trabalho dos administradores de redes e reduzido as preocupações dos usuários em geral com relação ao funcionamento adequado das redes.

Os sistemas de gerenciamento de redes disponíveis no mercado trabalham principalmente com o gerenciamento dos protocolos das camadas inferiores da rede, procurando mantê-las funcionando adequadamente. O gerenciamento de protocolos de aplicação ainda não está disponível nesses sistemas.

O gerenciamento dos protocolos de aplicação é, também, muito importante, uma vez que este permite a obtenção de informações sobre o tráfego gerado por cada aplicação. A partir da coleta de estatísticas sobre o fluxo de dados gerado pelos protocolos de aplicação, pode-se fazer uma melhor alocação de recursos como:

- linhas de comunicação de dados mais velozes: devem ser alocadas para os pontos da rede com maior tráfego, enquanto nas ligações subutilizadas seriam mantidas linhas com velocidade compatíveis;

- servidores de arquivos: devem ser instalados em pontos estratégicos da rede, isto é, pontos mais próximos dos sistemas que os necessitam com maior frequência e onde as linhas não estão sobrecarregadas, possibilitando maior facilidade de acesso; e

- distribuição de arquivos nos servidores: arquivos que são acessados com grande freqüência em servidores distantes podem ser armazenados em um servidor mais próximo, reduzindo assim o tráfego na rede.

Em países como o Brasil, é fundamental possuir informações mais precisas para uma melhor alocação dos recursos disponíveis, uma vez que esses são escassos.

Atualmente, está em fase final de instalação a Rede Nacional de Pesquisas, que interliga as principais instituições de ensino e pesquisa do país, e possui ligações com instituições do exterior. Nessa fase, é de grande importância identificar se as linhas que foram solicitadas são compatíveis com o uso e se suportarão a demanda futura. Quais linhas estão sobrecarregadas ou subutilizadas, quais as aplicações que geram maior tráfego são também informações necessárias para planejar melhor a utilização dos recursos disponíveis e assim obter um melhor desempenho da rede. Portanto, o uso de sistemas de gerenciamento de redes que ofereçam também o controle dos protocolos de aplicação é indispensável.

O modelo proposto nesse trabalho é uma solução para o problema do gerenciamento do protocolo FTP, e pode ser adaptado para o gerenciamento dos outros protocolos de aplicação, como SMTP e Telnet.

As principais vantagens do modelo proposto são a modularidade, a flexibilidade e a extensibilidade.

O modelo de gerenciamento do procotolo FTP proposto neste trabalho é modular, pois pode ser utilizado com diferentes sistemas de gerenciamento de redes disponíveis no mercado. Como o custo dos sistemas de gerenciamento de redes é elevado, a possibilidade de acoplar novas facilidades de gerenciamento aos sistemas já instalados é muito importante, pois facilita a adoção dos novos sistemas. O gerenciamento do protocolo FTP foi definido levando isso em consideração, isto é, de forma a poder ser adaptado nos principais sistemas de gerenciamento de redes disponíveis no mercado. O Capítulo 5 descreve como o gerenciamento do protocolo FTP pode ser implementado no sistema de gerenciamento de redes do ISODE e acessado a partir do *AIX NetVIew/6000* e do *SunNet Manager*.

O modelo proposto neste trabalho é bastante flexível, pois permite diferentes tipos de gerenciamento, de acordo com as necessidades existentes. Os seguintes níveis de gerenciamento do protocolo FTP são possíveis:

- controle do fluxo de mensagens FTP de um sistema: onde é gerenciado apenas as mensagens FTP com origem ou destino em um sistema, sem preocupação com domínios;
- gerenciamento em domínios: onde é feito o controle do fluxo de mensagens entre os domínios hierárquicos e federativos; e
- controle do fluxo de mensagens FTP entre dois sistemas específicos: onde o interesse é controlar as transações FTP entre dois sistemas;

O gerenciamento do protocolo FTP pode ser expandido para o controle de outras características do próprio FTP.

A coleta de dados a partir da alteração dos códigos fontes de clientes e servidores FTP, adotada no protótipo, possibilita a obtenção de informações mais detalhadas sobre as transações FTP e, assim, a implementação de serviços mais refinados. Esse método de

coleta de dados não é recomendado para uso em situações reais, por ser inviável alterar todos os códigos fontes dos clientes e servidores disponíveis. Em redes reais recomenda-se a coleta e análise dos pacotes em pontos estratégicos da rede, embora as informações coletadas a partir desse método devam ser mais genéricas.

O protótipo implementado para o gerenciamento do protocolo FTP no ISODE contém as características básicas do modelo proposto neste trabalho:

1. definição de quatro subgrupos para o grupo *ftp*, possibilitando assim diferentes configurações da MIB de acordo com as características dos sistemas onde o grupo *ftp* será instalado;

2. definição de domínios hierarquicos e federativos, com o objetivo de reduzir a complexidade do gerenciamento de redes com grande número de sistemas interligados e de facilitar o controle do fluxo de informações;

3. a possibilidade de serviços progressivos, de acordo com o interesse dos administradores da rede e com as características dos sistemas que serão gerenciados; e

4. a implementação do controle do protocolo FTP através de um subagente FTP garantiu a modularidade do gerenciamento desse protocolo e possibilita uma melhor configuração dos sistemas gerenciados, que podem manter as seguintes entidades:

   - apenas o subagente FTP, que exportaria seu grupo de objetos *ftp* para um agente SNMP remoto;
   - o subagente FTP e o agente SNMP para quem ele exporta seu grupo de objetos localmente;
   - apenas o gerente FTP, quando não está sendo realizado o gerenciamento em domínios;
   - o gerente FTP e um subagente FTP quando há o gerenciamento em domínios, e o subagente FTP exporta seus objetos para um agente SNMP remoto, que será conectado por um gerente do nível superior; e
   - o gerente FTP, um subagente FTP e um agente SNMP, todos no mesmo sistema, quando há o gerenciamento em domínios e o agente SNMP e o subagente FTP mantêm suas informações de gerenciamento para um gerente do nível superior.

O gerente FTP implementado no protótipo oferece um subconjunto das funcionalidades propostas suficiente para validar o modelo descrito no Capítulo 4.

O uso do ISODE como plataforma de desenvolvimento para o protótipo realmente facilitou a implementação por possibilitar o acesso a seus códigos fontes, que serviram

como um modelo para os códigos do subagente FTP. Por outro lado, a documentação sobre o pacote de gerenciamento de redes do ISODE é praticamente inexistente, o que tornou a compreensão de seus códigos bastante árdua.

No sentido de complementar o protótipo desenvolvido, são sugeridos os seguintes trabalhos futuros:

- implementação dos serviços de gerenciamento do protocolo FTP propostos no Capítulo 4. Esses serviços envolvem a emissão de relatórios com estatísticas periódicas do fluxo de dados gerado pelo FTP e o envio de mensagens automáticas;
- adaptação do protótipo para o gerenciamento do protocolo FTP a partir de gerentes do *AIX NetView/6000* e do *SunNet Manager*; e
- desenvolvimento de uma interface mais amigável para facilitar o uso do protótipo. O desenvolvimento de uma interface gráfica para o gerente FTP é facilitado nos sistemas comerciais como, por exemplo, no *SunNet Manager*.

Com relação ao modelo proposto para o gerenciamento do protocolo FTP, sugere-se os seguintes trabalhos futuros:

- adaptação do modelo para o SNMPv2: a segunda versão do gerenciamento de redes oferece uma série de facilidades para a comunicação entre gerentes e para uma maior segurança que merecem ser analisadas com mais detalhe e aproveitadas para o gerenciamento de protocolos de aplicação em domínios;
- alteração do processo de coleta de dados para ser realizado através da identificação dos pacotes que circulam na rede e uma análise de qual dos processos apresenta um melhor desempenho;
- expansão do modelo proposto para o gerenciamento de outros protocolos de aplicação, o que envolve:
  - a definição dos grupos de objetos necessários para o gerenciamento dos outros protocolos de aplicação;
  - implementação de subagentes encarregados de manter as informações dos novos grupos de objetos, seguindo o modelo do subagente FTP; e
  - a especificação e implementação dos serviços de gerenciamento necessários para cada protocolo;
- análise do impacto do gerenciamento de protocolos de aplicação em redes *Internet*; e
- adaptação do modelo proposto para o gerenciamento de protocolos de aplicação em redes OSI/ISO.

Apêndice A

Protocolo FTP

O protocolo FTP é o protocolo da camada de aplicação encarregado de prover um serviço de transferência de arquivos entre dois sistemas remotos no conjunto de protocolos *Internet* [PR85]. O FTP usa o protocolo TCP a nível de transporte e além disso utiliza algumas primitivas do protocolo Telnet em suas conexões de controle.

Os objetivos do protocolo FTP são:

- transferir dados de forma segura e eficiente;
- promover o compartilhamento de arquivos;
- estimular o uso de computadores remotos.

A transfêrencia de arquivos utilizando o protocolo FTP pode ser solicitada por um cliente FTP de duas maneiras distintas:

- entre o cliente e o servidor FTP;
- entre dois servidores FTP.

A Figura A.1 ilustra o processo de transferência de arquivos entre um cliente e um servidor FTP. O usuário solicita uma conexão para transferência de arquivos a um cliente FTP via sua interface. O Interpretador do Protocolo no Usuário - IPU - atende essa solicitação e se encarrega de iniciar uma conexão de controle com o Interpretador do Protocolo no Servidor - IPS.

O cliente FTP utiliza a conexão de controle para a inicialização da conexão FTP, o envio de comandos de navegação em diretórios e outros comandos de controle, além é claro da solicitação de transferências de arquivos. O servidor FTP utiliza a mesma conexão para enviar as respostas às solicitações do cliente.

Quando uma transferência de arquivos é solicitada pelo IPU, o servidor abre uma conexão de dados especificando seus parâmetros de acordo com os comandos previamente

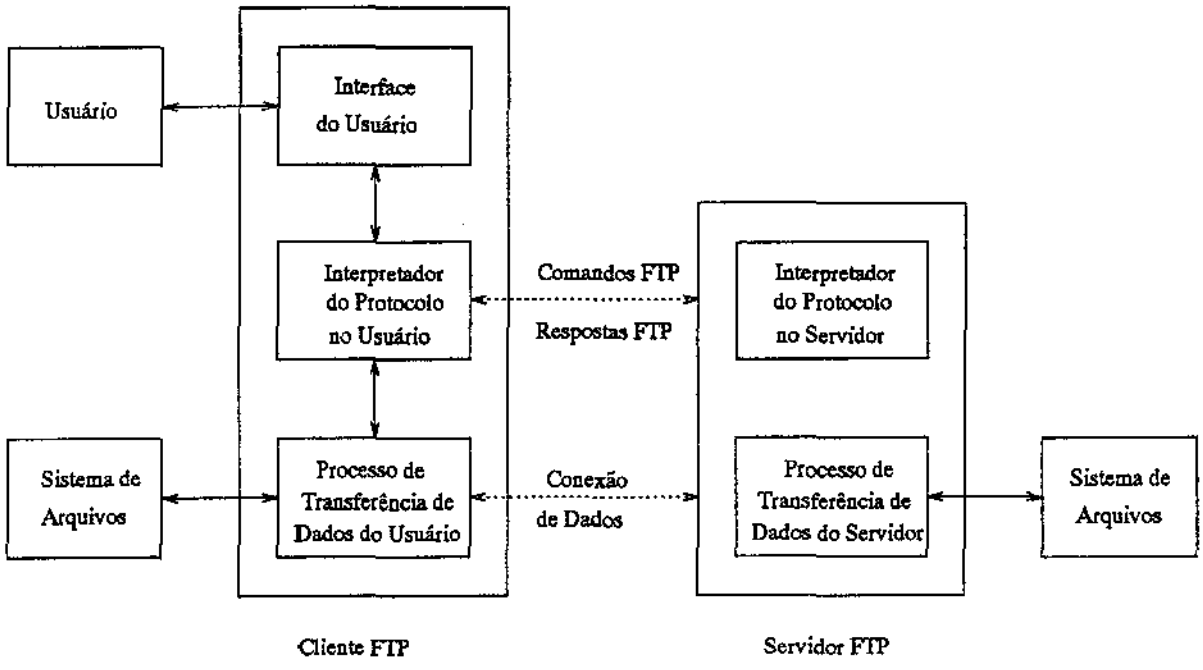


Figura A.1: Transferência de Arquivos entre Cliente e Servidor FTP

enviados pelo IPU ou com formato padrão. Os parâmetros dizem respeito à porta de dados a ser usada, o modo de transferência, tipo de representação, entre outros. A conexão de dados é bidirecioanl, isto é, pode ser utilizada tanto para o envio coma para a recepção de dados no servidor FTP. Após a transferência dos dados a conexão de dados é fechada.

A Figura A.2 mostra a transferência de arquivos entre dois sistemas, onde nenhum dos dois é o sistema local. Neste caso, o usuário abre duas conexões de controle, uma com cada servidor FTP, e programa-os para a abertura de uma conexão de dados entre eles e a transfêrencia dos arquivos.

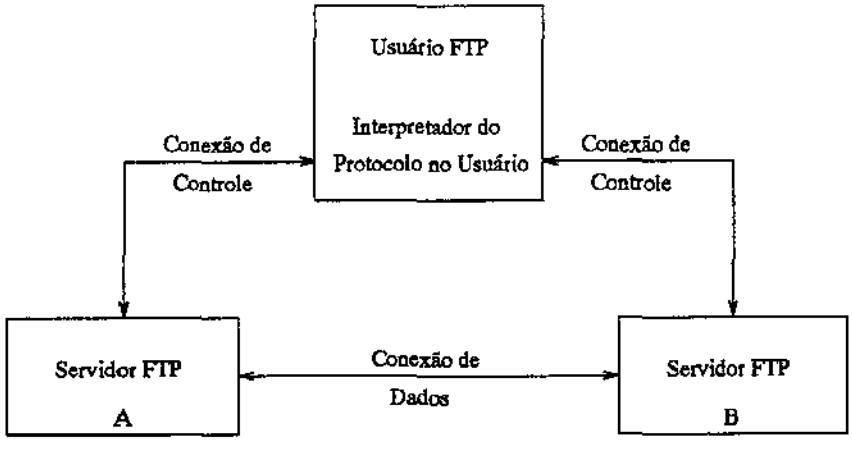


Figura A.2: Transferência de Arquivos entre dois Servidores

Em ambas as situações o protocolo FTP requer que a conexão de controle esteja aberta enquanto a transferência de dados está em progresso. É responsabilidade do usuário requerer o fechamento da conexão de controle quando ele finaliza o uso dos serviços do protocolo FTP. O servidor FTP pode abortar uma transferência de dados se a conexão de controle for fechada sem o comando adequado.

Os serviços oferecidos pelo protocolo FTP podem ser classificados da seguinte maneira:

1. Serviços de Controle de Acesso: permitem a um usuário acessar uma conta remota, navegar nos subdiretórios remotos, finalizar ou reinicializar uma conexão FTP.

2. Serviços para Alteração dos Parâmetros de Transferência de Arquivos: permitem ao usuário alterar os valores *default* dos parâmetros usados para a transferência de arquivos como, por exemplo, número da porta de dados, tipo de representação de dados, estrutura do arquivo e modo de transferência.

3. Serviços de Transferência de Arquivos: permitem aos usuários solicitar uma transferência de arquivos em qualquer sentido, reservar uma área no servidor de arquivos para uma posterior transferência de aquivos, remover ou renomear arquivos, construir diretórios, listar arquivos, entre outros serviços.

A lista dos comandos FTP é apresentada a seguir:

- Comandos para o controle de acesso ao servidor:
  - *user* <*username*>: identificação do usuário.
  - *pass* <*password*>: autenticação do usuário.
  - *acct* <*account-information*>: informações sobre a conta do usuário.
  - *cwd* <*pathname*>: comando para troca do diretório na máquina remota.
  - *cdup*: comando para voltar ao diretório anterior, pai na árvore de diretórios.
  - *rein*: comando para reiniciar uma conexão FTP.
  - *smnt* <*pathname*>: comando para montar uma estrutura diferente no sistema de arquivos.
  - *quit*: comando para encerrar uma conexão FTP.
- Comandos para a alteração dos parâmetros para transfêrencia de arquivos:
  - *port* <*host-port*>: altera a porta de transmissão de dados utilizada para transferência de arquivos.
  - *pasv*: coloca o servidor em modo passivo.
  - *type* <*type-code*>: altera o tipo de codificação para a transferência de arquivos.

- *stru <structure-code>*: altera a estrutura de arquivo para futuras transferências.
- *mode <mode-code>*: altera o modo de transmissão de dados.

• Comandos para transferência de arquivos:

- *retr <pathname>*: transmite um arquivo armazenado no servidor para o sistema local.
- *stor <pathname>*: armazena um arquivo local no servidor de arquivos. Atualiza arquivos existentes.
- *stou* : semelhante ao *stor*, mas o nome do arquivo deve ser único para no diretório.
- *appe <pathname>*: junta um arquivo no final de um outro arquivo do servidor.
- *allo <decimal-integer> [R <decimal-integer>]*: aloca um espaço no disco do servidor.
- *rest <marker>*: restaura uma conexão de dados a partir de um marcador.
- *rnfr <pathname>*: usado em conjunto com o rnto para troca de nomes dos arquivos do servidor.
- *rnto <pathname>*: idem ao anterior.
- *abor*: usada para terminar à força uma conexão.
- *dele <pathname>*:remove um arquivo do servidor.
- *rmd <pathname>*: remove um diretório no sistema do servidor.
- *mkd <pathname>*: constrói um diretório na máquina remota.
- *pwd <pathname>*: informa o diretório corrente a máquina remota.
- *list [<pathname>]*: lista as informações dos arquivos no diretório corrente ou especificado.
- *nlst [<pathname>]*:lista as informações dos arquivos no diretório corrente ou especificado. O resultado é apresentado em um formato que pode ser processado automaticamente por programas.
- *site <string>*: usada para o servidor prover serviços especificos para um determinado sistema.
- *syst*: usado para recuperar o tipo de sistema operacional do servidor.
- *stat [pathname]*: comando para o envio do estado de uma transferência em progresso.

- *help [string]*: o servidor envia informações de auxilio gerais ou de um determinado comando.
- *noop*: comando enviado apenas para receber uma resposta do servidor.

Todos os comandos do protocolo FTP devem ser confirmados pelo servidor FTP. A confirmação é realizada através de um *reply* que contém três dígitos e uma mensagem. Os dígitos informam se a operação que foi solicitada se completou com sucesso ou se houve algum erro e a mensagem descreve o que aconteceu.

Apêndice B

Grupo ftp para MIB

Este Anexo contém a definição do grupo *ftp* de objetos em ASN.1. Esse grupo foi introduzido na MIB sob a subárvore *enterprises*.

A definição do grupo *ftp* segue as normas e faz uso dos tipos predefinidos em SMI [Ros91a].

O grupo *ftp*, definido no arquivo *ftp.my*, foi dividido em quatro subgrupos conforme descrito no Capítulo 4. Cada subgrupo foi definido em um arquivo (*ftpcli.my*, *ftpserv.my*, *ftpman.my* e *ftpext.my*). Esses arquivos são apresentados a seguir.

```
-- ftp.my - FTP MIB

        FTP-MIB DEFINITIONS ::= BEGIN


        IMPORTS
                enterprises, Counter, IpAddress, Gauge,
 TimeTicks
                    FROM RFC1155-SMI
                OBJECT-TYPE
                    FROM RFC-1212;


        --  This MIB module uses the extended OBJECT-TYPE macro as
        --  defined in [9], and the TRAP-TYPE macro as defined in [10]


        -- this is the FTP MIB module
```

```
        ftp     OBJECT IDENTIFIER ::= { enterprises 50 }


        -- FTP group


        -- Implementation of the FTP group is mandatory for all
        -- managed systems which support an FTP protocol entity.
        -- Some of the objects defined below will be zero-valued
        -- in those FTP implementations that are optimized to
        -- support only those functions specific to either a FTP
        -- client or a FTP server.


        -- FTP subgroups (or bases)


        ftpcli  OBJECT IDENTIFIER ::= { ftp 1 }

        ftpserv OBJECT IDENTIFIER ::= { ftp 2 }

        ftpman  OBJECT IDENTIFIER ::= { ftp 3 }

        ftpext  OBJECT IDENTIFIER ::= { ftp 4 }

END


-- ftpcli.my - FTP Client MIB

        FTPCLI-MIB DEFINITIONS ::= BEGIN


        IMPORTS
                ftpcli, Counter, IpAddress, Gauge,
  TimeTicks
                    FROM RFC1155-SMI
```

```
            OBJECT-TYPE
                FROM RFC-1212;


        --  This MIB module uses the extended OBJECT-TYPE macro
        --  as  defined  in [9], and  the  TRAP-TYPE  macro  as
        --  defined in [10]


        -- this is the FTPCLI MIB module


--          ftpcli    OBJECT IDENTIFIER ::= { ftp 1 }
--          ftp    OBJECT IDENTIFIER ::= { enterprises xx }


        -- The ftpcli group

        -- Implementation of the ftpcli group is mandatory for
        -- all systems which support a ftp client.

        ftpcliDescr OBJECT-TYPE
           SYNTAX DisplayString (SIZE (0..255))
           ACCESS read-only
           STATUS mandatory
           DESCRIPTION
                   \"A textual description of the FTP  entity
                    (client). It is mandatory that this only
                    contains printable ASCII characters."
           ::= { ftpcli 1}

        ftpcliInMsgs OBJECT-TYPE
            SYNTAX  Counter
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                   "The total number of Messages delivered to
                    the FTP entity from the transport service."
            ::= { ftpcli 2 }
```

```
ftpcliOutMsgs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of FTP Messages sent by
             the FTP entity to the transport service."
    ::= { ftpcli 3 }

ftpcliInBytes OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The total number of bytes delivered to the
            FTP entity from the transport service."
    ::= { ftpcli 4 }

 ftpcliOutBytes OBJECT-TYPE
     SYNTAX Counter
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
            "The total number of bytes  sent by  from
             the FTP entity to the transport service."
     ::= { ftpcli 5}

ftpcliInFiles OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The total number of files delivered to the
            FTP entity from the transport service."
    ::= { ftpcli 6 }

 ftpcliOutFiles OBJECT-TYPE
     SYNTAX Counter
```

```
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The total  number of  files sent by from
            the FTP entity to the transport service."
    ::= { ftpcli 7}

ftpcliConns OBJECT-TYPE
   SYNTAX  Counter
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
           "The number of FTP connections."
   ::= { ftpcli 8 }

ftpcliDatConns OBJECT-TYPE
   SYNTAX  Counter
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
           "The number of FTP data connections."
   ::= { ftpcli 9 }

ftpcliConnTime OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The  Time  which  the  connections were
            opened."
    ::= { ftpcli 10 }

ftpcliBadAdds OBJECT-TYPE
   SYNTAX Counter
   ACCESS read-only
   STATUS mandatory
   DESCRIPTION
          "The number of FTP connections recused by
           address error."
```

```
            ::= { ftpcli 11 }

         ftpcliConnErrors OBJECT-TYPE
            SYNTAX Counter
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                   "The number of FTP connections errors."
            ::= { ftpcli 12 }

         ftpcliBadAuts OBJECT-TYPE
            SYNTAX Counter
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                   "The number of FTP access recused by bad
                    autorization."
            ::= { ftpcli 13 }

END


-- ftpserv.my - FTP Server MIB

         FTPSERV-MIB DEFINITIONS ::= BEGIN


         IMPORTS
                ftpserv, Counter, IpAddress, Gauge,
  TimeTicks
                    FROM RFC1155-SMI
                OBJECT-TYPE
                    FROM RFC-1212;


         --  This MIB module uses the extended OBJECT-TYPE macro
         --  as  defined  in [9], and  the  TRAP-TYPE  macro  as
         --  defined in [10]
```

```
-- this is the FTPSERV MIB module


--          ftpserv    OBJECT IDENTIFIER ::= { ftp 2 }
--          ftp    OBJECT IDENTIFIER ::= { enterprises xx }


-- The ftpserv group

-- Implementation of the ftpserv group is mandatory
-- for all systems which support a ftp server.

ftpservDescr OBJECT-TYPE
   SYNTAX DisplayString (SIZE (0..255))
   ACCESS read-only
   STATUS mandatory
   DESCRIPTION
           \"A textual description of the FTP entity.
            It is mandatory that this only contains
            printable ASCII characters."
   ::= { ftpserv 1}

ftpservInMsgs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
           "The total number of Messages delivered to
            the FTP entity from the transport service."
    ::= { ftpserv 2 }

ftpservOutMsgs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
           "The total number of FTP Messages sent by
```

```
                the FTP entity to the transport service."
    ::= { ftpserv 3 }

ftpservInBytes OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The total number of bytes delivered to the
            FTP entity from the transport service."
    ::= { ftpserv 4 }

ftpservOutBytes OBJECT-TYPE
     SYNTAX Counter
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
            "The total number of bytes sent by the FTP
             entity to the transport service."
     ::= { ftpserv 5}

ftpservInFiles OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The total number of files delivered to the
            FTP entity from the transport service."
    ::= { ftpserv 6 }

ftpservOutFiles OBJECT-TYPE
     SYNTAX Counter
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
            "The  total  number of files sent by the
             FTP entity to the transport service."
     ::= { ftpserv 7}
```

```
ftpservConns OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of FTP connections."
    ::= { ftpserv 8 }

ftpservDatConns OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of FTP data connections."
    ::= { ftpserv 9 }

ftpservConnTime OBJECT-TYPE
     SYNTAX TimeTicks
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
            "The  Time  which  the connections were
             opened."
     ::= { ftpserv 10 }

ftpservExcConns OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The number of FTP connections recused by
            excess of opened connections."
    ::= { ftpserv 11 }

ftpservConnErrors OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
```

```
            "The number of FTP connections errors."
    ::= { ftpserv 12 }

ftpservBadAuts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The number of FTP access recused by bad
             autorization."
    ::= { ftpserv 13 }

ftpservRscFlrs OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The number of resource failures in the File
             Server."
    ::= { ftpserv 14 }

ftpservBadCmds OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The  number of bad commands received by the
             File Server."
    ::= { ftpserv 15 }

ftpservFails OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The number of fails in the File Server."
    ::= { ftpserv 16 }

ftpservFileTable OBJECT-TYPE
```

```
            SYNTAX  SEQUENCE OF FtpservFileEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "The FTP File Table."
            ::= { ftpserv 17 }

        ftpservFileEntry OBJECT-TYPE
            SYNTAX  FtpservFileEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    \"Information about one File."
            INDEX   { ftpservFileDate }
            ::= { ftpservFileTable 1 }

        FtpservFileEntry ::=
            SEQUENCE {
                ftpservFileName
                        DisplayString,
                ftpservFileStatus
INTEGER,
                ftpservFileType
                        INTEGER,
                ftpservFileSize
                        INTEGER,
                ftpservFileOwner
                        DisplayString,
                ftpservFileDate
                        TimeTicks,
 ftpservFileGets
Counter,
 ftpservFileUser1
DisplayString,
 ftpservFileUser2
DisplayString,
 ftpservFileUser3
DisplayString,
 ftpservFileUser4
```

```
DisplayString,
 ftpservFileUser5
DisplayString,
 ftpservFileUser6
DisplayString,
 ftpservFileUser7
DisplayString
               }

          ftpservFileName  OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..255))
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                      "The file name."
              ::= { ftpservFileEntry 1 }

          ftpservFileStatus OBJECT-TYPE
              SYNTAX  INTEGER {
                         good(1),
                         bad(2),
                         new(3),
newversion(4),
deleted(5),
other(6)
                      }
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                      "The file status."
              ::= { ftpservFileEntry 2 }

          ftpservFileType OBJECT-TYPE
              SYNTAX  INTEGER {
                         ascii(1),
                         binary(2),
                         other(3)
                      }
              ACCESS  read-write
```

```
        STATUS  mandatory
        DESCRIPTION
                "The file type."
        ::= { ftpservFileEntry 3 }

    ftpservFileSize OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The file size."
        ::= { ftpservFileEntry 4 }

    ftpservFileOwner OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The e-mail address of the file owner."
        ::= { ftpservFileEntry 5 }

    ftpservFileDate OBJECT-TYPE
        SYNTAX  TimeTicks
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The  date  and  time when  the  file was
                 stored."
        ::= { ftpservFileEntry 6 }

    ftpservFileGets OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The total number of times that this file
                 was read."
        ::= { ftpservFileEntry 7 }
```

```
ftpservFileUser1  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The e-mail address of the last user that
             read this file."
    ::= { ftpservFileEntry 8 }

ftpservFileUser2  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The  e-mail  address  of  a  user  that
             recentely read this file."
    ::= { ftpservFileEntry 9 }

ftpservFileUser3  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The  e-mail  address  of  a  user  that
             recentely read this file."
    ::= { ftpservFileEntry 10 }

ftpservFileUser4  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The  e-mail  address  of  a  user  that
             recentely read this file."
    ::= { ftpservFileEntry 11 }

ftpservFileUser5  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
```

```
            STATUS  mandatory
            DESCRIPTION
                    "The  e-mail  address  of  a  user  that
                     recentely read this file."
            ::= { ftpservFileEntry 12 }

        ftpservFileUser6  OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The  e-mail  address  of  a  user  that
                     recentely read this file."
            ::= { ftpservFileEntry 13 }

         ftpservFileUser7  OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The  e-mail  address  of  a  user  that
                     recentely read this file."
            ::= { ftpservFileEntry 14 }

END


-- ftpman.my - FTP Manager MIB

        FTPMAN-MIB DEFINITIONS ::= BEGIN


        IMPORTS
                ftpman, Counter, IpAddress, Gauge,
  TimeTicks
                    FROM RFC1155-SMI
                OBJECT-TYPE
                    FROM RFC-1212;
```

```
        -- This MIB module uses the extended OBJECT-TYPE macro
        -- as defined in [9], and the TRAP-TYPE macro as
        -- defined in [10]


        -- this is the FTPMAN MIB module


--          ftpman    OBJECT IDENTIFIER ::= { ftp 3 }
--          ftp    OBJECT IDENTIFIER ::= { enterprises xx }


        -- The ftpman group

        -- Implementation of the ftpman group is mandatory for
        -- all systems which contain a FTP domain manager.

        ftpmanDescr OBJECT-TYPE
           SYNTAX DisplayString (SIZE (0..255))
           ACCESS read-only
           STATUS mandatory
           DESCRIPTION
                   \"A textual description of the FTP domain
                    manager. It is mandatory that this only
                    contains printable ASCII characters."
           ::= { ftpman 1}

        ftpmanInMsgs OBJECT-TYPE
            SYNTAX  Counter
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The total number of Messages delivered
                     to the FTP entity  from  the transport
                     service."
            ::= { ftpman 2 }
```

```
ftpmanOutMsgs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of FTP Messages sent
             by the  FTP  entity  to the transport
             service."
    ::= { ftpman 3 }

ftpmanInBytes OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The total number of bytes delivered to
            the  FTP  entity  from  the  transport
            service."
    ::= { ftpman 4 }

 ftpmanOutBytes OBJECT-TYPE
     SYNTAX Counter
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
            "The total number of bytes sent by the
             FTP entity to the transport service."
     ::= { ftpman 5}

ftpmanInFiles OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The total number of files delivered to
            the  FTP  entity  from  the  transport
            service."
    ::= { ftpman 6 }
```

```
ftpmanOutFiles OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The total number of Files sent by the
             FTP entity to the transport service."
    ::= { ftpman 7}

ftpmanConns OBJECT-TYPE
   SYNTAX  Counter
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
           "The number of FTP connections."
   ::= { ftpman 8 }

ftpmanDatConns OBJECT-TYPE
   SYNTAX  Counter
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
           "The number of FTP data connections."
   ::= { ftpman 9 }

ftpmanConnTime OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The Time which the connections were
            opened."
    ::= { ftpman 10 }

ftpmanBadAdds OBJECT-TYPE
   SYNTAX Counter
   ACCESS read-only
   STATUS mandatory
   DESCRIPTION
```

```
            "The number of FTP connections recused by
             address error."
    ::= { ftpman 11 }

 ftpmanConnErrors OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The number of FTP connections errors."
    ::= { ftpman 12 }

 ftpmanBadAuts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The number of FTP access recused by bad
             autorization."
    ::= { ftpman 13 }

ftpmanServTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF FtpmanServEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The FTP Server Table."
    ::= { ftpman 14 }

ftpmanServEntry OBJECT-TYPE
    SYNTAX  FtpmanServEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            \"Information about each server in this
             manager domain."
    INDEX   { ftpmanServAddr }
    ::= { ftpmanServTable 1 }
```

```
FtpmanServEntry ::=
    SEQUENCE {
        ftpmanServAddr
           IpAddress,
        ftpmanServState
           INTEGER
        }

ftpmanServAddr OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The IP address of the servers in this
             manager domain."
    ::= { ftpmanServEntry 1 }


ftpmanServState OBJECT-TYPE
    SYNTAX  INTEGER {
                accessible(1),
                notaccessible(2),
                fail(3)
            }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The File Server state."
    ::= { ftpmanServEntry 2 }

ftpmanCliTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF FtpmanCliEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The FTP Client Table."
    ::= { ftpman 15 }

ftpmanCliEntry OBJECT-TYPE
```

```
    SYNTAX  FtpmanCliEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            \"Information about each FTP client in this
             manager domain."
    INDEX   { ftpmanCliAddr }
    ::= { ftpmanCliTable 1 }

FtpmanCliEntry ::=
    SEQUENCE {
        ftpmanCliAddr
           IpAddress,
        ftpmanCliState
           INTEGER
        }

ftpmanCliAddr OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The IP address of the Clients in this
             manager domain."
    ::= { ftpmanCliEntry 1 }


ftpmanCliState OBJECT-TYPE
    SYNTAX  INTEGER {
                accessible(1),
                notaccessible(2),
                fail(3)
            }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The FTP Client state."
    ::= { ftpmanCliEntry 2 }
```

```
ftpmanManTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF FtpmanManEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The FTP Manager Table."
    ::= { ftpman 14 }

ftpmanManEntry OBJECT-TYPE
    SYNTAX  FtpmanManEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            \"Information about each manager in this
             manager domain."
    INDEX   { ftpmanManAddr }
    ::= { ftpmanManTable 1 }

FtpmanManEntry ::=
    SEQUENCE {
        ftpmanManAddr
           IpAddress,
        ftpmanManState
           INTEGER
        }

ftpmanManAddr OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The IP address of the Managers in this
             manager domain."
    ::= { ftpmanManEntry 1 }

ftpmanManState OBJECT-TYPE
    SYNTAX  INTEGER {
                accessible(1),
                notaccessible(2),
```

```
                        fail(3)
                    }
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The FTP Managers state."
            ::= { ftpmanManEntry 2 }

END


-- ftpext.my - FTP Extra MIB

        FTPEXT-MIB DEFINITIONS ::= BEGIN


        IMPORTS
                ftpext, Counter, IpAddress, Gauge,
  TimeTicks
                    FROM RFC1155-SMI
                OBJECT-TYPE
                    FROM RFC-1212;

        -- This MIB module uses the extended OBJECT-TYPE macro
        -- as defined in [9], and the TRAP-TYPE macro as
        -- defined in [10]


        -- this is the FTPEXT MIB module


--          ftpext    OBJECT IDENTIFIER ::= { ftp 4 }
--          ftp    OBJECT IDENTIFIER ::= { enterprises xx }



        -- The ftpext group
```

```
-- Implementation of the ftpext group is necessary to
-- manage FTP flow between two specific systems.

ftpextDescr OBJECT-TYPE
   SYNTAX DisplayString (SIZE (0..255))
   ACCESS read-only
   STATUS mandatory
   DESCRIPTION
           \"A textual description of the other FTP
            entity. It is mandatory that this only
            contains printable ASCII characters."
   ::= { ftpext 1}

ftpextInMsgs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
           "The total number of Messages delivered
            to the  FTP  entity from the transport
            service."
    ::= { ftpext 2 }

ftpextOutMsgs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
           "The total number of FTP Messages sent
            by the  FTP  entity  to the transport
            service."
    ::= { ftpext 3 }

ftpextInBytes OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
```

```
            "The total number of bytes delivered to
             the FTP entity from the transport
             service."
    ::= { ftpext 4 }

ftpextOutBytes OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The total number of bytes sent by the FTP
             entity to the transport service."
    ::= { ftpext 5}

ftpextInFiles OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The total number of files delivered to the
             FTP entity from the transport service."
    ::= { ftpext 6 }

ftpextOutFiles OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
            "The  total number of files sent by the
             FTP entity to the transport service."
    ::= { ftpext 7 }

ftpextConns OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of FTP connections."
    ::= { ftpext 8 }
```

```
ftpextDatConns OBJECT-TYPE
   SYNTAX  Counter
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
           "The number of FTP data connections."
   ::= { ftpext 9 }

ftpextConnTime OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
           "The  Time  which the connections were
            opened."
    ::= { ftpext 10 }

ftpextConnErrors OBJECT-TYPE
   SYNTAX Counter
   ACCESS read-only
   STATUS mandatory
   DESCRIPTION
          "The number of FTP connections errors."
   ::= { ftpext 11 }

ftpextBadAuts OBJECT-TYPE
   SYNTAX Counter
   ACCESS read-only
   STATUS mandatory
   DESCRIPTION
          "The number of FTP access recused by bad
           autorization."
   ::= { ftpext 12 }

END
```

Apêndice C

Abreviações

ACSE - *Association Control Service Element.*
APIs - *Application Programming Interfaces.*
ASEs - *Application Service Elements.*
ASN.1 - *Abstract Syntax Notation-One.*
ATM - *Asynchronous Transfer Mode.*
AT&T *Americam Telegraph & Telephone.*
CCITT - *Consultive Committe for International Telegraph and Telephone.*
CMIP - *Common Management Information Protocol.*
CMIPDUs - *Common Management Information Protocol Data Units.*
CMISE - *Common Management Information Service Element.*
CMOT - *CMIP over TCP/IP.*
DEC - *Digital Equipments Corporation.*
DECnet - *DEC Network Protocol.*
DN - *Distinguished Name.*
EGP - *Exterior Gateway Protocol.*
EMA - *Enterprise Management Architecture.*
FDDI - *Fiber Distributed Data Interface.*
FTAM - *File Transfer, Access and Management.*
FTP - *File Transfer Protocol.*
HP - *Hewlett Packard, Inc.*
IBM - *International Busines Machine.*
ICMP - *Internet Control Message Protocol.*
IEC - *International Eletrotecnical Committe.*
ISODE - *ISO Development Environment.*
ISO - *International Organization for Standardization.*
LME - *Layer Management Entity.*
LNCC - Laboratório Nacional de Ciência da Computação.

MAPDUs - *Management Application Protocol Data Units.*
MDB - *Management DataBase.*
MIB - *Management Information Base.*
MIT - *Massachusetts Institute of Technology.*
MOSY - *Managed Object Syntax Compiler (YACC based).*
NOC - *Network Operation Center.*
OSI - *Open System Interconnection.*
PDUs - *Protocol Data Units.*
PUC/RJ - Pontifícia Universidade Católica do Rio de Janeiro.
RDNs - *Relative Distinguished Names.*
RNP - Rede Nacional de Pesquisa.
ROSE - *Remote Operation Service Element.*
RPC - *Remote Procedure Call.*
RPC/XDR - *Remote Procedure Call/External Data Representation.*
SMAE - *System Management Application Entity.*
SMASE - *System Management Application Service Element.*
SMI - *Structure of Management Information.*
SMTP - *Simple Mail Transfer Protocolo.*
SMUX - *SNMP Multiplexing.*
SNA - *System Network Architeture.*
SNMP - *Simple Network Management Protocol.*
SNMPv2 - *Simple Network Management Protocol version 2.*
TCP/IP - *Transmission Control Protocol / Internet Protocol.*
UDP - *User Datagram Protocol.*
UFRJ - Universidade Federal do Rio de Janeiro.

Bibliografia

[AsS93] J. F. Cunha A. schweitzer and E. S. Specialski. TCMIP - Um Protocolo de Gerência OSI para uma rede TCP/IP. *Anais do 11 SBRC - Simpósio Brasileiro de Redes de Computadores*, May 1993.

[Bla89] U. D. Black. *Data Networks: Concepts, Theory and Practice.* Prentice-Hall International, Inc., 1st edition, 1989.

[BOU92] J. Y. LE BOUDEC. The Asynchronous Transfer Mode: a Tutorial. *Computer Network and ISDN Systems*, (24), 1992.

[BRI93] BRISA. *Gerenciamento de Redes - Uma Abordagem de Sistemas Abertos.* Makron Books, 1st edition, 1993.

[CFSD90] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol (SNMP). Request for Comments 1157, May 1990.

[CM94a] J. A. Carrilho and 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*, Jun 1994.

[CM94b] J. A. Carrilho and E. R. M. Madeira. Um Esquema para o Gerenciamento do Protocolo FTP baseado em Domínios. *Anais do 12 SBRC - Simpósio Brasileiro de Redes de Computadores*, May 1994.

[Com91] D. E. Comer. *Internetworking with TCP/IP - Volume 1 - Principles, Protocols and Architecture.* Prentice-Hall International, Inc., 2nd edition, 1991.

[CR93] J. Case and A. Rijsinghani. FDDI Management Information Base. Request for Comments 1512, Sep 1993.

[Dav89] J. R. Davin. The SNMP Development Kit. Technical report, MIT - Massachusetts Institute of Technology, Jan 1989.

[DJE92] M. F. Schwartz D. J. Ewing, R. S. Hall. A Measurement Study of Internet File Transfer Traffic. Technical report, University of Colorado, Jan 1992.

[DLRM91] E. Decker, P. Langille, A. Rijsinghani, and K. McCloghrie. Definitions of Managed Objects for Bridges. Request for Comments 1286, Dec 1991.

[FK94a] N. Freed and S. Kille. Mail Monitoring MIB. Request for Comments 1566, Jan 1994.

[FK94b] N. Freed and S. Kille. X.500 Directory Monitoring MIB. Request for Comments 1567, Jan 1994.

[GM93a] J. Galvin and K. McCloghrie. Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2) . Request for Comments 1445, Apr 1993.

[GM93b] J. Galvin and K. McCloghrie. Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2) . Request for Comments 1447, Apr 1993.

[GM93c] J. Galvin and K. McCloghrie. Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2) . Request for Comments 1446, Apr 1993.

[IBM92] IBM Corporation, International Technical Support Center . *Overview and Examples of Using AIX NetView/6000*, May 1992.

[ISOa] ISO/IEC. Common Management Information Protocol Specification. International Standard ISO/IEC/DIS 9596 .

[ISOb] ISO/IEC. Management Information Model. International Standard ISO/IEC/DIS 10165-1 .

[JCW93a] M. T. Rose J. Case, K. McCloghrie and S. Waldbusser. Introduction to version 2 of the Internet-standard Network Management Framework. Request for Comments 1441, Apr 1993.

[JCW93b] M. T. Rose J. Case, K. McCloghrie and S. Waldbusser. Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2). Request for Comments 1442, Apr 1993.

[JGW93a] M. T. Rose J. Galvin, K. McCloghrie and S. Waldbusser. Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2) . Request for Comments 1450, Apr 1993.

[JGW93b] M. T. Rose J. Galvin, K. McCloghrie and S. Waldbusser. Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework . Request for Comments 1452, Apr 1993.

[JGW93c] M. T. Rose J. Galvin, K. McCloghrie and S. Waldbusser. Manager-to-Manager Management Information Base . Request for Comments 1451, Apr 1993.

[JGW93d] M. T. Rose J. Galvin, K. McCloghrie and S. Waldbusser. Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2) . Request for Comments 1448, Apr 1993.

[JGW93e] M. T. Rose J. Galvin, K. McCloghrie and S. Waldbusser. Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2) . Request for Comments 1449, Apr 1993.

[KJ94] et al K. Jayanthi. Optimising FTP Traffic in the Internet. *Proc. of INET'94/JENC5 - The Annual Conference of Internet Society and Joint-European Network Conference*, Jun 1994.

[Kro92] E. Krol. *The Whole Internet.* O'Reilly & Associates, Inc., 1st edition, 1992.

[MR88] K. McCloghrie and M. T. Rose. Management Information Base for Network Management of TCP/IP-based Internets. Request for Comments 1066 (obsoleto), Aug 1988.

[MR91a] A. Mankin and K. Ramakrishnan. Gateway Congestion Control Survey. Request for Comments 1254, Aug 1991.

[MR91b] K. McCloghrie and M. T. Rose. Management Information Base for Network Management of TCP/IP-based Internets: MIB-II. Request for Comments 1213, Mar 1991.

[Mye] C. Myers. Wuarchive - FTP Server Description . Washington University in Saint Louis.

[Nan91] B. Nance. Dueling Protocols. *BYTE*, pages 183–190, Mar 1991.

[OSN92] OSN. How to manage cmip and snmp. *The Open Systems Newsletter*, Oct 1992.

[OSN93] OSN. Snmpv2 versu cmip. *The Open Systems Newsletter*, Apr 1993.

[Per91] Performance Systems International, Inc. *The ISO Development Environment: User's Manual*, Jul 1991.

[Pos82] J. B. Postel. Simple Mail Transfer Protocol - SMTP. Request for Comments 821, Aug 1982.

[PR83] J. B. Postel and J. Reynolds. Telnet Protocol Specification. Request for Comments 854, May 1983.

[PR85] J. B. Postel and J. Reynolds. File Transfer Protocol - FTP. Request for Comments 959, Oct 1985.

[Pro87] Projeto MAP/TOP. *Network Management Requirements Specification*, Jul 1987.

[RM90] M. T. Rose and K. McCloghrie. Structure and Identification of Management Information for TCP/IP-based Internets. Request for Comments 1155, May 1990.

[Ros90] M. T. Rose. *The Open Book - A Practical Perspective on OSI.* Prentice-Hall International, Inc., 1st edition, 1990.

[Ros91a] M. T. Rose. *The Simple Book - An Introduction to Management of TCP/IP-based Internets.* Prentice-Hall International, Inc., 1st edition, 1991.

[Ros91b] M. T. Rose. SNMP MUX Protocol and MIB. Request for Comments 1227, May 1991.

[Ros93] M. T. Rose. SNMP over OSI. Request for Comments 1418, Mar 1993.

[RW94] M. A. Rocha and C. B. Westephall. Gerência de Redes de Computadores através de Novos Agentes. *Anais do 12 SBRC - Simpósio Brasileiro de Redes de Computadores*, May 1994.

[SDFC89] M. Schoffstall, C. Davin, M. Fedor, and J. Case. SNMP over Ethernet. Request for Comments 1089, Fev 1989.

[Sil] C. K. Silveira. Um Esquema para o Gerenciamento de Protocolos de Aplicação em Redes TCP/IP. Tese de Mestrado em Andamento no DCC - IMECC - UNICAMP.

[ST94] A. C. B. Silva and L. M. R. Tarouco. Uma Proposta para Gerência de Correio Eletrônico. *Anais do 12 SBRC - Simpósio Brasileiro de Redes de Computadores*, May 1994.

[Sun91] Sun Microsystems, Inc. *SunNet Manager 1.1 : Installation and User's Guide*, 1991.

[SW93] A. C. B. Silva and C. B. Westephall. Implementação de Novos Agentes para Gerência em Redes de Computadores. *Anais do 11 SBRC - Simpósio Brasileiro de Redes de Computadores*, May 1993.

[Tan89] A.S. Tanenbaum. *Computer Networks.* Prentice-Hall International, Inc., 2nd edition, 1989.

[UWH90] L. Besaw U. Warrier and B. Handspicker. The Common Management Information Services and Protocol for the Internet (CMOT and CMIP). Request for Comments 1189, Oct 1990.

[VTH90] A. Wolisz V. Tschammer and J. Hall. Support for Cooperation and Coherence in an Open Service Environment. *Proc. 2nd IEEE Workshop on the Future Trends of Distributed Computing in the 1990s*, Sep 1990.

[WB91] S. Willis and J. Burruss. Definitions of Managed Objects for the Bordwe Gateway Protocol (Version 3). Request for Comments 1269, Oct 1991.

[WT94] C. G. Weissheimer and L. M. R. Tarouco. Distribuição da Gerência na Rede. *Anais do 12 SBRC - Simpósio Brasileiro de Redes de Computadores*, May 1994.