UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE SOFTWARE PARA GERENCIAMENTO DE DESEMPENHO DE UM SERVIDOR EM UMA REDE DE COMPUTADORES, UTILIZANDO O PROTOCOLO SNMP

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

DAVI FLORIANI COELHO

BLUMENAU, JUNHO/2001

2001/1-20

PROTÓTIPO DE SOFTWARE PARA GERENCIAMENTO DE DESEMPENHO DE UM SERVIDOR EM UMA REDE DE COMPUTADORES, UTILIZANDO O PROTOCOLO SNMP

DAVI FLORIANI COELHO

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Sérgio Stringari — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Sérgio Stringari

_____

Prof. Francisco Adell Péricas

_____

Prof. Dalton Solano dos Reis

AGRADECIMENTOS

Agradeço a minha mãe e a meus familiares por estarem a meu lado em todos os momentos, me incentivando e me ajudando a realizar mais esta etapa em minha vida.

Agradeço a meus colegas, Cássio, Henrique e Marciano, que no decorrer do curso, nos tornamos grandes amigos.

Ao coordenador do trabalho de conclusão de curso, professor José Roque Voltolini da Silva, por toda a atenção e apoio dispensado.

Ao professor Sérgio Stringari, pela paciência e dedicação e confiança durante a orientação deste trabalho final.

A todos aqueles que direta ou indiretamente contribuíram para a conclusão desta etapa em minha vida.

E finalmente agradeço a Deus por tudo.

RESUMO

Este Trabalho de Conclusão de Curso (TCC), apresenta um estudo sobre gerência de redes de computadores enfatizando a gerência de desempenho. Apresenta também a especificação e implementação de um protótipo de software para facilitar este gerenciamento, para um servidor em uma *Local Area Network* (LAN), utilizando o *Simple Network Management Protocol* (SNMP), e usando o padrão *Common Object Request Broker Architecture* (CORBA).

ABSTRACT

This Work of Conclusion of Course (TCC), presents a study on management of networks computers emphasizing the management of performance. It also presents the specification and implementation of a software prototype to facilitate this management, for a server in a Local Area Network (LAN), using the Simple Network Management Protocol (SNMP), and using the standard Common Object Request Broker Architecture (CORBA).

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

API - *Application Program Interface*

ARP - *Address Resolution Protocol*

ASN.1 - *Abstract Syntax Notation One*

AT - *Address Translation*

CMIP - *Common Management Information Protocol*

EGP - *External Gateway Protocol*

EJB - *Enterprise Java Beans*

ICMP - *Internet Control Message Protocol*

IDL - *Interface Definition Language*

IP - *Internet Protocol*

ISO - *International Organization for Standardization*

LAN - *Local Area Network*

MIB - *Management Information Base*

NMS - *Network Management Stations*

ORB - *Object Request Broker*

OSI - *Open Systems Interconnection*

SMI - *Structure of Management Information*

SNMP - *Simple Network Management Protocol*

TCP - *Transmission Control Protocol*

UDP - *User Datagram Protocol*

UML - *Unified Modeling Language*

1 INTRODUÇÃO

Segundo Specialski (1999), por menor e mais simples que seja, uma rede de computadores precisa ser gerenciada a fim de garantir, aos seus usuários, a disponibilidade de serviços a um nível de desempenho aceitável.

À medida que a rede cresce, aumenta a complexidade de seu gerenciamento, forçando a adoção de ferramentas automatizadas para a sua monitoração e controle.

Segundo Klauck (1999), o gerenciamento de desempenho preocupa-se com o desempenho corrente da rede, incluindo parâmetros estatísticos tais como atrasos, vazão, disponibilidade e número de retransmissões. Consiste em um conjunto de funções responsáveis por manter e examinar registros com histórico dos estados do sistema para fins de planejamento e análise.

Segundo Krupskaia (2000), o padrão *Common Object Request Broker Architecture* (CORBA) é um modelo proposto pela *Object Management Group* (OMG), objetivado a promover na teoria e na prática a tecnologia dos objetos de forma distribuída, ou seja, é uma estrutura comum para o desenvolvimento independente de aplicações, usando técnicas de orientação a objeto em redes de computadores heterogêneas. Visa diminuir consideravelmente os custos, reduzir a complexidade, e proporcionar caminhos para o surgimento de novas aplicações a partir dos conceitos propostos pela OMG.

Desta forma, o presente trabalho de conclusão de curso apresenta o desenvolvimento de um protótipo de software para gerência de desempenho de um servidor em uma rede de computadores utilizando o SNMP e implementado utilizando o padrão CORBA/OMG. Para descrição da modelagem foi utilizada a ferramenta Rational Rose, pela qual suporta a metodologia UML. Para desenvolvimento do protótipo utilizou-se a ferramenta JBuilder 4.

1.1 MOTIVAÇÃO

Com o advento dos modelos abertos de computação distribuída orientada a objetos, uma alternativa para a integração de sistemas de gerenciamento de redes e aplicações que tem recebido grande atenção é a utilização de CORBA.

Tendo em vista que existem alguns trabalhos em nível de pós-graduação envolvendo gerência de redes e objetos distribuídos, escolheu-se a área funcional de gerência de desempenho para avaliar a junção destas tecnologias.

1.2 OBJETIVOS

O objetivo principal do trabalho desenvolvido é o de demonstrar um protótipo de software para auxiliar no gerenciamento de desempenho de um servidor em uma rede de computadores utilizando o protocolo SNMP.

Os objetivos específicos são:

a) a utilização do padrão CORBA/OMG;
b) monitoramento de uma rede da família Microsoft Windows 9.x e Microsoft Windows NT.

1.3 ORGANIZAÇÃO DO TEXTO

Este trabalho está organizado da seguinte forma:

O capitulo 1 apresenta, objetivamente, uma introdução ao trabalho, suas motivações, seus objetivos e a organização do texto.

O capitulo 2 são apresentadas as considerações sobre gerência de redes, bem como as etapas no processo de gerência, os modelos de gerenciamento, os protocolos de gerenciamento e as áreas funcionais da gerência de redes.

O capitulo 3 apresenta considerações sobre gerência de desempenho, suas características como atraso e *throughput* e suas relações.

O capitulo 4 apresenta uma visão sobre o SNMP, sua descrição, seus elementos e suas operações.

O capitulo 5 descreve as tecnologias e ferramentas utilizadas, como a UML e o CORBA.

O capitulo 6 descreve a especificação, implementação do protótipo e a apresentação da operacionalidade do mesmo.

O capitulo 7 apresenta as conclusões do trabalho, identificando o sucesso dos objetivos definidos para este trabalho como as dificuldades e os problemas encontrados na elaboração do mesmo.

2 GERÊNCIA DE REDES

Segundo Klauck (1999), gerência de redes de computadores pode ser conceituada como a coordenação (controle de atividades e monitoração do uso) de recursos materiais (*modems*, roteadores, *switches*, etc.) e lógicos (protocolos), fisicamente distribuídos na rede, assegurando na medida do possível, confiabilidade, tempos de resposta aceitáveis e segurança nas informações. Portanto, gerenciar uma rede significa dotar este sistema de mecanismos de monitoramento e controle dos elementos de rede de tal sorte a permitir seu funcionamento dentro de uma qualidade de serviço esperada pelos usuários da mesma.

Segundo Specialski (1999), a adoção de um software de gerenciamento não resolve todos os problemas da pessoa responsável pela administração da rede. Geralmente o usuário de um software de gerenciamento espera muito dele e conseqüentemente, fica frustrado quanto aos resultados que obtém. Por outro lado, esses mesmos softwares quase sempre são sub-utilizados, isto é, possuem inúmeras características inexploradas ou utilizadas de modo pouco eficiente. Para gerenciar um recurso, é necessário conhecê-lo muito bem e visualizar claramente o que este recurso representa no contexto da rede.

O investimento em um software de gerenciamento pode ser justificado pelos seguintes fatores:

a) as redes e recursos de computação distribuídos estão se tornando vitais para a maioria das organizações. Sem um controle efetivo, os recursos não proporcionam o retorno que a corporação requer;
b) contínuo crescimento da rede em termos de componentes, usuários, interfaces, protocolos e fornecedores ameaçam o gerenciamento com perda de controle sobre o que está conectado na rede e como os recursos estão sendo utilizados;
c) os usuários esperam uma melhoria dos serviços oferecidos (ou no mínimo, a mesma qualidade), quando novos recursos são adicionados ou quando são distribuídos;
d) Os recursos computacionais e as informações da organização geram vários grupos de aplicações de usuários com diferentes necessidades de suporte nas áreas de desempenho, disponibilidade e segurança. O gerente da rede deve atribuir e controlar recursos para balancear estas várias necessidades;

e) À medida que um recurso fica mais importante para a organização, maior fica a sua necessidade de disponibilidade. O sistema de gerenciamento deve garantir esta disponibilidade;
f) A utilização dos recursos deve ser monitorada e controlada para garantir que as necessidades dos usuários sejam satisfeitas a um custo razoável.

Segundo Klauck (1999), em geral, numa empresa moderna as metas são fixadas no sentido de torná-la cada vez maior, o que implica necessariamente em sistemas de comunicação cada vez mais complexos, suportando cada vez mais aplicações e usuários. À medida que a empresa cresce, cria-se uma dependência funcional da empresa com o sistema de comunicação utilizado. Como conseqüência deste crescimento, maiores são os pontos de erro no sistema, capazes de prejudicar o seu desempenho como um todo. Surge desse fato a importância do uso eficiente de mecanismos de gerência que permitam manter os índices de desempenho das redes de computadores em níveis aceitáveis.

Uma implicação imediata no crescimento das redes de computadores é o aumento substancial de problemas e a conseqüente necessidade de gerência do maior número possível de recursos materiais e lógicos destas redes. É importante notar também que a complexidade desta gerência é provocada, principalmente, pela convivência de sistemas heterogêneos devido à convivência de sistemas proprietários (IBM, DEC, etc.) e suas conseqüentes diferentes soluções de gerência. Assim, a complexidade na atividade de gerenciamento será tão maior quanto mais diversificados forem os equipamentos de rede fornecidos por diferentes fabricantes.

Segundo BRISA (1993), normalmente, tais ferramentas de gerenciamento não permitem a interoperabilidade com outras ferramentas e um manuseio integrados entre elas, fazendo com que, os proprietários das redes não possam dispor em seu ambientes de produtos de outros fornecedores, muitas vezes mais eficientes tecnicamente e de custos mais reduzidos. Assim sendo a necessidade de produtos de gerenciamento de múltiplos fabricantes que interagem entre si é extremamente real e urgente.

O mecanismo de apoio à operação de gerenciamento da rede constitui a ferramenta mais importante para rastrear e resolver problemas. Um esquema desse tipo deve incorporar as seguintes características:

a) gerenciamento da rede deve ser parte integral da mesma;
b) devem ser permitidos múltiplos pontos de acesso ao gerenciamento da rede (estações de controle);
c) a informação sobre a rede, bem como as estatísticas de sua atividade (normal ou com problemas), incluindo os dados de eventos, deverão ser passíveis de obtenção de maneira centralizados e apresentados em forma facilmente compreensíveis, possivelmente usando gráficos;
d) um esquema de tratamento prioritário deve permitir que as mensagens de controle da rede precedam outros tipos de tráfego;
e) devem existir mecanismos de segurança que limitem o acesso à rede e detecte o seu uso não autorizado;
f) as funções de gerenciamento de rede devem operar independentemente do meio de transmissão;
g) deve haver um banco de dados contendo informações sobre todos os componentes da rede e de seus usuários;
h) alterações na rede e redirecionamento devem poder ser efetivados de forma flexível e simples.

A integração das funções de gerenciamento é importante e não pode ser acomodada pela simples adição de equipamento extra. Nas atuais arquiteturas de redes, nas quais é usada uma abordagem de estratificação de funções em níveis, o gerenciamento deve também ser parte das funções inerentes a cada nível. Por exemplo, testes que envolvem a interface física do equipamento (tais como o comando de teste por *loop-back*) devem estar embutidos no nível físico, devendo ser possível seu acionamento de alguma forma controlada pelo sistema gerenciador da rede.

Por outro lado, para que isto seja implementado, devem ser evitadas soluções particulares que impliquem, por exemplo, o uso de modems com características não

padronizadas. É preciso buscar uma solução que possibilite o gerenciamento de redes num ambiente heterogêneo.

2.1 ETAPAS NO PROCESSO DE GERÊNCIA

Segundo Klauck (1999), a primeira idéia na solução de um sistema de gerência consiste em se utilizar um computador interagindo com os diversos componentes da rede a serem gerenciados para deles extrair as informações necessárias a sua gerência.

Etapas no processo de gerência de redes:

a) coleta de dados: é um processo, em geral automático, que consiste de monitoração sobre os recursos gerenciados;
b) diagnóstico: esta etapa consiste no tratamento e análise realizados a partir dos dados coletados. O processo de gerenciamento executa uma série de procedimentos (por intermédio de um operador ou não) com o intuito de determinar a causa do problema representado no recurso gerenciado;
c) ação: uma vez diagnosticado o problema, cabe uma ação, ou controle, sobre o recurso, caso o evento não tenha sido passageiro (incidente operacional).

2.2 MODELOS DE GERENCIAMENTO DE REDE

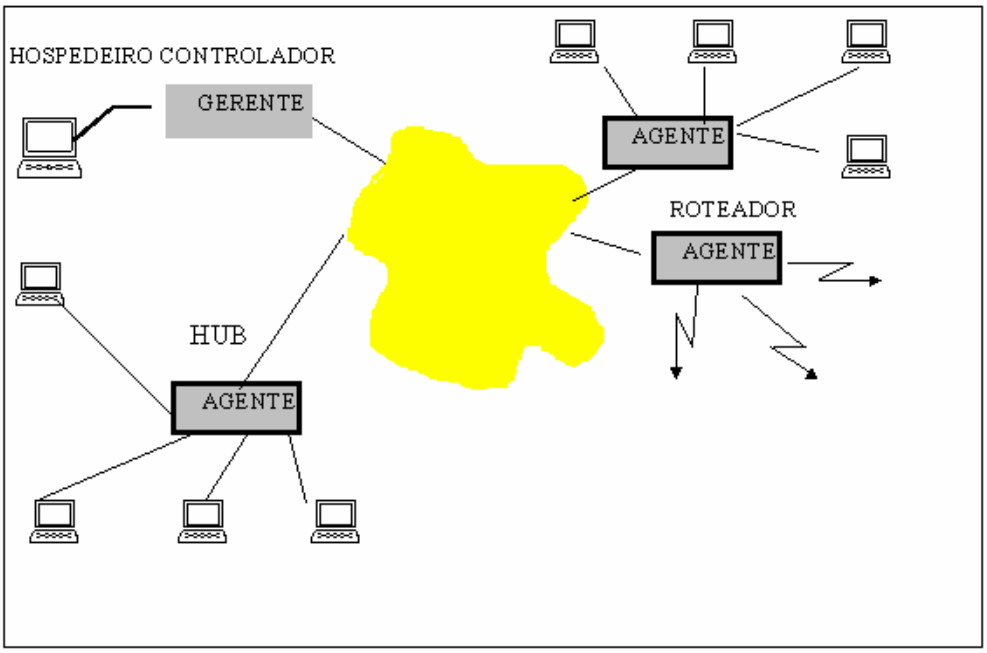
Segundo Specialski (1999), um sistema de gerenciamento de rede é uma coleção de ferramentas para monitorar e controlar a rede, integradas da seguinte forma:

a) uma única interface de operador, com um poderoso, mas amigável conjunto de comandos, para executar a maioria ou todas as tarefas de gerenciamento da rede;
b) uma quantidade mínima de equipamentos separados, isto é, a maioria do hardware e software necessário para o gerenciamento da rede é incorporado nos equipamentos de usuários existentes.

O software usado para realizar as tarefas de gerenciamento reside nos computadores hospedeiros (estações de trabalho) e nos processadores de comunicação (*switches*, *routers*, *hubs*,...).

Todos os equipamentos da rede, que fazem parte do sistema de gerenciamento, possuem um conjunto de software destinado às tarefas de coletar informações sobre as atividades relacionadas com a rede, armazenar estatísticas localmente e responder aos comandos do centro de controle da rede. Estes nodos são referenciados como agentes. No mínimo um hospedeiro da rede é designado para as tarefas de controlador da rede (gerente) e possui uma coleção de software chamada aplicação de gerenciamento da rede. A aplicação de gerenciamento da rede possui uma interface que permite, a um usuário autorizado, gerenciar a rede. A figura 1 apresenta um cenário possível de um sistema de gerenciamento de rede.

Figura 1 - Configuração de um sistema de gerenciamento de rede.



FONTE: Specialski (1999)

Um software de gerenciamento genérico é composto por:

a) elementos gerenciados;
b) agentes;
c) gerentes;
d) bancos de dados de informações;
e) protocolos para troca de informações de gerenciamento;
f) interfaces para programas aplicativos;
g) interfaces com o usuário.

2.3 PROTOCOLOS DE GERENCIAMENTO

Segundo Klauck (1999), é clara a necessidade de estabelecer monitoramento e controle sobre todos os componentes da rede, de forma a garantir que esta esteja sempre em funcionamento e que os problemas sejam identificados, isolados e solucionados o mais rápido possível. Entretanto, esta não é uma tarefa fácil. As redes têm assumido grandes proporções, com um grande número de computadores, além da constante adição de novos componentes, oferecendo integração dados/voz, multiplexadores e roteadores, além de tantos outros, o que tem adicionado mais complexidade a esse ambiente.

Para atender a esta necessidade de gerenciamento foram desenvolvidos protocolos de gerenciamento. A principal preocupação de um protocolo de gerenciamento é permitir aos gerentes de rede realizar tarefas, tais como: obter dados sobre desempenho e tráfego da rede em tempo real, diagnosticar problemas de comunicação e reconfigurar a rede atendendo às mudanças nas necessidades dos usuários e do ambiente. Porém, vários obstáculos teriam que ser superados, entre eles a heterogeneidade dos equipamentos de rede (computadores, roteadores e dispositivos de meio), dos protocolos de comunicação e das tecnologias de rede. Adicionalmente, era necessário que esse gerenciamento fosse integrado, pois uma solução genérica e integrada auxiliaria os usuários a evitar os altos custos de uma solução específica, além de facilitar a manutenção, o monitoramento, o crescimento e a evolução da rede. Sem um gerenciamento integrado, a rede pode degradar até se tornar completamente ineficiente.

As informações de gerenciamento permitem, dentre outras tarefas, produzir registros de auditoria para todas as conexões, desconexões, falhas na rede e outros eventos significativos na rede. Esses registros, por sua vez, permitem determinar futuras necessidades de adicionar equipamentos, identificar e isolar erros comuns de um cliente, além de estudar outras tendências de uso. Devemos destacar ainda, que estas informações devem possibilitar uma atuação preventiva, e não meramente reativa, com relação aos problemas.

Ciente destas dificuldades, a ISO, vem desenvolvendo padrões para o gerenciamento de redes OSI, tendo como protocolo de gerência o CMIP. Do outro lado, existe o *Institute of Electrical and Electronics Engineers* (IEEE) com um conjunto de padrões para o gerenciamento de redes TCP/IP, normalmente referenciados como SNMP. Atualmente, quase

todas as plataformas de gerenciamento de redes internet comercialmente disponíveis implementam o protocolo SNMP devido à sua simplicidade de implementação em relação ao CMIP.

2.4 ÁREAS FUNCIONAIS DA GERÊNCIA DE REDES

Segundo Klauck (1999) e BRISA (1993), o gerenciamento é uma prática vital para a operação de redes de computadores. O uso dos serviços das redes é afetado pela disponibilidade e eficiência do gerenciamento de redes. Atualmente, as atividades de controle e monitoração, quando disponíveis, em sua maioria, são realizadas por meio do uso de ferramentas ou softwares proprietários, cujo manuseio não é integrado. Isto torna o gerenciamento da rede caro e ineficiente. Assim sendo, a necessidade de produtos de gerenciamento que, independente do fabricante, interajam entre si é fundamental.

Segundo Klauck (1999), com o objetivo de suprir estas necessidades, a ISO desenvolve padrões de gerenciamento que permitem a interoperabilidade de múltiplos e diversificados sistemas computacionais e redes de computadores.

Dentro deste contexto, a ISO dividiu a atividade de gerenciamento em cinco áreas funcionais específicas: gerenciamento de configuração, gerenciamento de falhas, gerenciamento de contabilização, gerenciamento de segurança e gerenciamento de desempenho. Dentro de cada área funcional, foram desenvolvidos padrões de funções (incluindo requisitos, modelos e serviços) para o gerenciamento das redes. Essas funções são processos de aplicação de gerenciamento que utilizam os serviços oferecidos pela camada de aplicação.

O SNMP, devido a sua natureza como uma solução rápida e reduzida do CMIP da ISO, implementa apenas um pequeno conjunto destas funções definidas pela ISO. Por isso ele é considerado “simples” de implementação e é um protocolo bem mais “leve” que o CMIP.

2.4.1 GERENCIAMENTO DE CONFIGURAÇÃO

Segundo Klauck (1999), a função do gerenciamento de configuração envolve a manutenção e monitoração da estrutura física e lógica da rede, incluindo a existência de

componentes e sua interconectividade. Corresponde ao conjunto de processos que exercem o controle sobre os objetos gerenciados, identificando-os, coletando e fornecendo dados sobre os mesmos a fim de dar suporte a funções de:

a) atribuição de valores iniciais aos parâmetros do sistema;
b) início e encerramento de operações sobre objetos gerenciados;
c) alteração da configuração do sistema;
d) associação de nomes a conjuntos de objetos gerenciados.

O objetivo principal do gerenciamento de configuração é manter um registro detalhado das configurações de rede antigas, atuais e propostas. Dependendo do ambiente, esse conhecimento detalhado pode compreender uma lista muito grande de informações sobre configuração.

Mudanças, inclusões e exclusão na rede devem ser acompanhadas pelos sistemas de gerenciamento para que sempre se conheça a configuração da mesma. Esta é provavelmente a parte mais importante do gerenciamento de rede, pois não se pode gerenciar uma rede sem que se conheça ou se tenha acesso à configuração da mesma.

Através do gerenciamento de configuração é possível, por exemplo, controlar informações gerais sobre licença de software, computadores e outros dispositivos, bem como informações detalhadas sobre versão de aplicativo e driver.

Segundo Specialski (1999), o gerenciamento de configuração está relacionado com a inicialização da rede e com uma eventual desabilitação de parte ou de toda a rede. Também está relacionado com as tarefas de manutenção, adição e atualização de relacionamentos entre os componentes e do *status* dos componentes durante a operação da rede.

Alguns recursos podem ser configurados para executar diferentes serviços como, por exemplo, um equipamento pode atuar como roteador, como estação de trabalho ou ambos. Uma vez decidido como o equipamento deve ser usado, o gerente de configuração pode escolher o software apropriado e um conjunto de valores para os atributos daquele equipamento.

O gerente da rede deve ser capaz de, inicialmente, identificar os componentes da rede e definir a conectividade entre eles. Também deve ser capaz de modificar a configuração em resposta a avaliações de desempenho, recuperação de falhas, problemas de segurança, atualização da rede ou a fim de atender a necessidades dos usuários.

Relatórios de configuração podem ser gerados periodicamente ou em resposta a requisições de usuários.

2.4.2 GERENCIAMENTO DE FALHAS

Segundo Klauck (1999), o gerenciamento de falhas é responsável pela manutenção e monitoração do estado de cada um dos objetos gerenciados e pelas ações necessárias ao restabelecimento ou isolamento das unidades com problemas. As informações coletadas podem ser usadas em conjunto com o mapa da rede, para indicar quais elementos da rede estão funcionando, quais operam precariamente ou quais permanecem fora de operação.

Também é possível que o gerenciamento de falhas gere um registro das ocorrências, um diagnóstico de falhas e uma correlação entre os resultados do diagnóstico e as subseqüentes ações de reparo.

O gerenciamento de falhas utiliza hardware e software para alertar os gerentes a respeito de uma falha e para auxiliar no reparo. Pode ser também utilizado hardware e software de tolerância a falhas ou redundantes que podem continuar a fornecer serviços de rede mesmo quando ocorrer falha.

Por exemplo, seria possível utilizar as seguintes ferramentas para realizar o gerenciamento de falhas:

a) sistema de gerenciamento de rede: um sistema de gerenciamento de rede é uma combinação de hardware e software que acompanha o funcionamento dos componentes da rede. Este sistema geralmente inclui um terminal que concentra e emite os alarmes, uma indicação visual dos dispositivos que falharam e uma interface com o dispositivo de relatório remoto;
b) analisador de protocolo: um analisador de protocolo é uma ferramenta de hardware e de software que monitora o tráfego na rede. Essa ferramenta pode ajudar a

compreender as complexas interações que ocorrem na rede, identificando como os protocolos são utilizados em cada comunicação;

c) verificador de cabo: um verificador de cabos é um dispositivo de hardware que identifica falhas no meio de transmissão. Dependendo do meio, pode ser identificado o cabo específico que está falhando e onde ocorreu a falha;
d) sistemas redundantes: os sistemas redundantes utilizam peças idênticas de hardware ou de software para realizar as mesmas funções. Por exemplo, servidores de arquivos espelhados que armazenam exatamente os mesmos dados. Se um servidor falhar, o outro imediatamente o substitui continuando a servir os clientes da rede. Com isso, pode-se identificar o servidor com defeito e repará-lo sem que os usuários da rede sintam o impacto disto;
e) dispositivo de *backup* e arquivamento de dados: os dispositivos de *backup* e o arquivamento de dados não ajudam a identificar falhas, mas podem reduzir significativamente o impacto.

O número crescente de produtos de detecção e reparo de falhas que têm sido comercializados proporciona sistemas de gerenciamento de falhas cada vez mais eficazes.

Segundo Specialski (1999), falhas não são o mesmo que erros. Uma falha é uma condição anormal cuja recuperação exige ação de gerenciamento. Uma falha normalmente é causada por operações incorretas ou um número excessivo de erros. Por exemplo, se uma linha de comunicação é cortada fisicamente, nenhum sinal pode passar através dela. Um grampeamento no cabo pode causar distorções que induzem a uma alta taxa de erros. Certos erros como, por exemplo, um bit errado em uma linha de comunicação, podem ocorrer ocasionalmente e normalmente não são considerados falhas.

Para controlar o sistema como um todo, cada componente essencial deve ser monitorado individualmente para garantir o seu perfeito funcionamento. Quando ocorre uma falha, é importante que seja possível, rapidamente:

a) determinar o componente exato onde a falha ocorreu;
b) isolar o resto da rede da falha, de tal forma que ela continue a funcionar sem interferências;

c) reconfigurar ou modificar a rede para minimizar o impacto da operação sem o componente que falhou;
d) reparar ou trocar o componente com problemas para restaurar a rede ao seu estado anterior.

O impacto e a duração do estado de falha pode ser minimizado pelo uso de componentes redundantes e rotas de comunicação alternativas, para dar à rede um grau de “tolerância à falhas”.

2.4.3 GERENCIAMENTO DE CONTABILIZAÇÃO

Segundo Klauck (1999), o gerenciamento de contabilização preocupa-se com a manutenção e monitoração de quais recursos e de quanto destes recursos estão sendo utilizados. Estas informações podem ser utilizadas para estatísticas ou para "faturamento".

As informações colhidas pelo gerenciamento de contabilização podem ser utilizadas para alocar novos recursos na rede ou simplesmente para planejar melhorias.

Utilizando o gerenciamento de contabilização, pode-se compreender os custos reais da rede, definir suas capacidades e estabelecer políticas e procedimentos para torná-la mais eficiente.

Segundo Specialski (1999), mesmo que nenhuma cobrança interna seja feita pela utilização dos recursos da rede, o administrador da rede deve estar habilitado para controlar o uso dos recursos por usuário ou grupo de usuários, com o objetivo de:

a) evitar que um usuário ou grupo de usuários abuse de seus privilégios de acesso e monopolize a rede, em detrimento de outros usuários;
b) evitar que usuários façam uso ineficiente da rede, assistindo-os na troca de procedimentos e garantindo a desempenho da rede;
c) conhecer as atividades dos usuários com detalhes suficientes para planejar o crescimento da rede.

2.4.4 GERENCIAMENTO DE SEGURANÇA

Segundo Klauck (1999), o gerenciamento de segurança refere-se à proteção e controle de acesso das informações de gerenciamento. Isso pode envolver a geração, distribuição e armazenamento de chaves de criptografia. Senhas, autorizações e outros controles de acesso devem ser mantidos de forma a proteger qualquer informação de gerenciamento de cada elemento da rede.

O gerenciamento de segurança aborda os aspectos de segurança essenciais para operar uma rede corretamente e proteger os objetos gerenciados. O sistema de gerenciamento deve providenciar alarmes para o administrador da rede quando eventos relativos à segurança forem detectados.

A tarefa de gerenciamento de segurança pode abranger o seguinte:

a) identificar os riscos de segurança e suas conseqüências;
b) implementar projetos e equipamentos de rede seguros;
c) administrar grupos e senhas de usuários;
d) usar equipamentos de monitoração da rede para registrar o uso, relatar violações ou fornecer alarmes para atividades de alto risco.

Segundo Specialski (1999), o gerenciamento da segurança provê facilidades para proteger recursos da rede e informações dos usuários. Estas facilidades devem estar disponíveis apenas para usuários autorizados. É necessário que a política de segurança seja robusta e efetiva e que o sistema de gerenciamento da segurança seja, ele próprio, seguro.

3 GERENCIAMENTO DE DESEMPENHO

Segundo Klauck (1999), enquanto o gerenciamento de falhas é principalmente reativo, o gerenciamento de desempenho é ativo, envolvendo a coleta e interpretação das medições periódicas dos indicadores de desempenho, identificando gargalos, avaliando tendências, e fazendo previsões do desempenho futuro da rede.

Segundo Specialski (1999), o gerenciamento do desempenho de uma rede consiste na monitoração das atividades da rede e no controle dos recursos através de ajustes e trocas. Algumas das questões relativas ao gerenciamento do desempenho são:

a) qual é o nível de capacidade de utilização;
b) se o tráfego é excessivo;
c) se o *throughput* foi reduzido para níveis aceitáveis;
d) se existem gargalos;
e) se o tempo de resposta está aumentando.

Para tratar estas questões, o gerente deve focalizar um conjunto inicial de recursos a serem monitorados, a fim de estabelecer níveis de desempenho. Isto inclui associar métricas e valores apropriados aos recursos de rede que possam fornecer indicadores de diferentes níveis de desempenho. Muitos recursos devem ser monitorados para se obter informações sobre o nível de operação da rede. Colecionando e analisando estas informações, o gerente da rede pode ficar mais e mais capacitado no reconhecimento de situações indicativas de degradação de desempenho.

Estatísticas de desempenho podem ajudar no planejamento, administração e manutenção de grandes redes. Estas informações podem ser utilizadas para reconhecer situações de gargalo antes que elas causem problemas para o usuário final. Ações corretivas podem ser executadas, tais como, trocar tabelas de roteamento para balancear ou redistribuir a carga de tráfego durante horários de pico, ou ainda, em longo prazo, indicar a necessidade de expansão de linhas para uma determinada área.

Segundo BRISA (1993), o gerenciamento de desempenho é confundido, às vezes com o gerenciamento de falhas. Muitas instalações de processamento de dados tendem a confundir desempenho com disponibilidade. O gerenciamento de desempenho é importante não só para

garantir a qualidade do serviço acordada com os usuários, como também para assegurar que esta é atingida com os menores custos possíveis. Pode-se, por meio de gerenciamento de desempenho adequar os meios de comunicação utilizados pelos usuários às suas reais necessidades, auxiliando o setor responsável pela administração de redes a antecipar-se aos usuários na manutenção dos níveis de desempenho dos serviços oferecidos, como por exemplo, o tempo de resposta.

O gerenciamento de desempenho está diretamente relacionado ao planejamento da capacidade do sistema sob gerenciamento.

3.1 CARACTERÍSTICAS DE DESEMPENHO DE REDE

Segundo Comer (2001), informalmente, as redes podem ser classificadas como de baixa velocidade ou de alta velocidade. Porém, tais definições são inadequadas porque as tecnologias de rede mudam tão rapidamente que uma rede classificada como "alta velocidade" pode ser tornar de velocidade média ou baixa em apenas três ou quatro anos. Deste modo, quando os cientistas ou engenheiros precisam especificar precisamente as velocidades de uma rede, eles não usam termos informais, as medidas quantitativas são importantes porque possibilitam comparar duas redes qualquer. Este tópico define as duas medidas quantitativas fundamentais de uma rede e explica como elas relacionam a capacidade.

3.1.1 ATRASO

Segundo Comer (2001), a primeira propriedade importante das redes que pode ser quantitativamente medida é o atraso *(delay)*. O atraso de uma rede especifica quanto tempo leva um bit de dados viajar através da rede de um computador até outro; o atraso é medido em segundos ou frações de segundos. Os atrasos podem diferir ligeiramente, dependendo da localização do par específico de computadores que se comunicam. Embora os usuários se importem apenas com o atraso total de uma rede, os engenheiros precisam fazer medidas mais precisas. Deste modo, os engenheiros normalmente reportam tanto o atraso máximo como o atraso médio e dividem o atraso em várias partes.

Parte do atraso em uma rede se deve ao fato de que um sinal exige uma pequena quantia de tempo para viajar através de um fio ou fibra ótica. Tais atrasos são conhecidos

como atrasos de propagação *(propagation delays)* e são geralmente proporcionais à distância abrangida. Por exemplo, uma LAN típica usada dentro de um único edifício tem um atraso de um milissegundo. Embora tais atrasos pareçam irrelevantes para um ser humano, um computador moderno pode executar mais de cem mil instruções em um milissegundo. Deste modo, um atraso de um milissegundo é significativo para um computador. Uma rede que usa um satélite em órbita da Terra para encaminhar dados de um continente a outro tem um atraso muito maior - mesmo na velocidade da luz, leva mais de cem milissegundos para um bit viajar para o satélite e voltar a Terra.

Dispositivos eletrônicos em uma rede (por exemplo, *hubs*, *bridges* ou *switches* de pacote) introduzem outra origem de atraso conhecido como atraso de comutação *(switching delays)*. Um dispositivo eletrônico espera até que todos os bits de um pacote tenham chegado, e então leva uma pequena quantia de tempo para escolher o próximo *hop* antes de enviar um pacote. CPUs rápidas e o hardware de propósito especial têm tornado os atrasos de comutação entre os menos significativos.

Como a maioria das LANs usa meios compartilhados, os computadores devem esperar até que o meio esteja disponível. Por exemplo, uma rede Ethernet usa *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) e que uma rede de *Token Ring* exige que um transmissor espere por um *token*. Tais atrasos, que não são normalmente grandes, são conhecidos como atrasos de acesso *(access delays)*.

Uma forma final acontece em uma *Wire Area Network* (WAN) com comutação de pacotes. Em cada *switch* de pacote, enfileira pacotes sendo recebidos como parte do processo de armazenamento e encaminhamento. Se a fila já contém pacotes, o novo pacote pode precisar esperar enquanto a CPU encaminha pacotes que chegaram mais cedo. Tais atrasos são conhecidos como atrasos de filas *(queueing delays)*.

3.1.2 THROUGHPUT

Segundo Comer (2001), a segunda propriedade fundamental de redes que pode ser quantitativamente medida é o *throughput*. O *throughput* é uma medida da taxa em que podem ser enviados dados através da rede e está normalmente especificada em bits por segundo

(bps). A maioria das redes tem um *throughput* de vários milhões de bits por segundo *(Mbps)*. Porém, sistemas de comunicação mais antigos oferecem *throughputs* baixos de até mesmo 300 ou 1200 bits por segundo, e existem redes com *throughputs* altos que chegam a vários gigabits por segundo *(Gbps)*.

Como o *throughput* pode ser medido de vários modos, deve-se ter cuidado ao se especificar exatamente o que foi medido. Por exemplo, sabe-se que a capacidade de *throughput* do hardware subjacente é chamada de largura da banda. De fato, o termo largura de banda é às vezes usado como sinônimo para *throughput*. Porém, programadores e usuários não se importam com a capacidade do hardware subjacente - estão interessados na taxa em que podem ser enviados dados através de uma rede. Em particular, na maioria das tecnologias cada quadro contem um cabeçalho, o que significa que o *throughput efetivo* (ou seja, a taxa em que um computador pode enviar dados) é menor que a largura de banda de hardware. Não obstante, a largura de banda de hardware é freqüentemente usada como uma aproximação do *throughput* da rede porque a largura de banda fornece um limite superior de *throughput* - é impossível para um usuário enviar dados mais rapidamente do que a taxa em que o hardware pode transferir bits.

Os profissionais de redes usam freqüentemente o termo velocidade como um sinônimo para *throughput*. Por exemplo, poderia ouvir-se: "A rede tem uma velocidade de dez milhões de bits por segundo". Embora tais declarações sejam comuns, elas podem gerar confusão por que atraso e *throughput* são idéias separadas. De fato, o *throughput* é uma medida de capacidade, não de velocidade. Para entender a relação, imagine uma rede como uma estrada entre dois lugares, e os pacotes que viajam através da rede são análogos a carros viajando através da estrada. A taxa de *throughput* determina quantos carros podem entrar na estrada a cada segundo, mas o atraso determina quanto tempo um único carro leva para viajar a estrada inteira de uma cidade a outra. Por exemplo, uma estrada que pode aceitar um carro a cada cinco segundos, tem um *throughput* de 0,2 carros por segundo. Se o carro exige 30 segundos para percorrer a estrada inteira, a estrada tem um atraso de 30 segundos. Agora considere o que acontece se uma segunda pista é aberta (ou seja, o dobro da capacidade). Será possível que dois carros entrem a cada cinco segundos, portanto o *throughput* dobra para 0,4 carros por

segundo. Naturalmente, os 30 segundos de atraso permanecerão inalterados porque cada carro ainda deve percorrer a distância inteira.

3.1.3 A RELAÇÃO ENTRE ATRASO E THROUGHPUT

Segundo Comer (2001), em teoria, atraso e *throughput* de uma rede são independentes. Na prática, eles podem estar relacionados. Para entender por quê, pense sobre a analogia da estrada discutida acima. Se os carros entram na estrada em intervalos de tempo regulares, os carros que viajam ao longo da estrada em velocidade uniforme são espaçados em intervalos uniformes. Se um carro diminui a velocidade por alguma razão, os demais atrás dele irão desacelerar também, causando um congestionamento de tráfego temporário. Os carros que entram na estrada quando o congestionamento estiver ocorrendo experimentarão atrasos mais longos que os carros que viajam em uma estrada não congestionada. Uma situação semelhante acontece nas redes. Se um *switch* de pacote tem uma fila de pacote esperando quando um novo pacote chega, esse pacote será colocado no fim da fila e precisará esperar enquanto o *switch* encaminha os pacotes anteriores. Análogo aos engarrafamentos em uma estrada, o tráfego excessivo em uma rede é chamado de congestionamento *(congestion)*. Claramente, os dados entrando em uma rede congestionada experimentarão atrasos mais longos do que os dados entrando em uma rede ociosa.

Os cientistas da computação estudaram a relação entre o atraso e o congestionamento e descobriram que, em muitos casos, o atraso esperado pode ser estimado a partir da porcentagem da capacidade de rede sendo usada. Se D_0 denota o atraso quando a rede está ociosa, e U é um valor entre *0* e *1* que denota a utilização atual, o atraso efetivo, D, é dado por uma fórmula simples:

$$D = D_0 \ / \ (1- U).$$

Quando uma rede estiver completamente ociosa, U é zero, e o atraso efetivo é D_0. Quando uma rede operar a 50% de sua capacidade, o atraso efetivo dobra. À medida que o tráfego se aproxima da capacidade da rede (isto é, U se torna perto de 1), o atraso tende ao infinito. Embora a fórmula forneça somente uma estimativa que pode acontecer pode-se concluir:

O throughput e o atraso não são completamente independentes. Com o aumento do tráfego na rede de computadores, aumentam os atrasos; uma rede que opera perto dos 100% de sua capacidade de *throughput* experimentará atrasos severos.

3.1.4 PRODUTO THROUGHPUT-ATRASO

Segundo Comer (2001), Uma vez que o atraso e o *throughput* de uma rede são conhecidos, é possível também computar outra medida interessante, o produto throughput-atraso *(throughput-delay product)*. Para entender o significado do produto throughput-atraso, considere a analogia da estrada: quando os carros estiverem entrando em uma estrada em uma taxa fixa de T carros por segundo. E levar D segundos para um carro percorrer a estrada, então T x D carros adicionais terão entrado na estrada quando o primeiro carro tiver feito uma viagem completa. Deste modo, pode haver um total de T x D carros na estrada a qualquer momento.

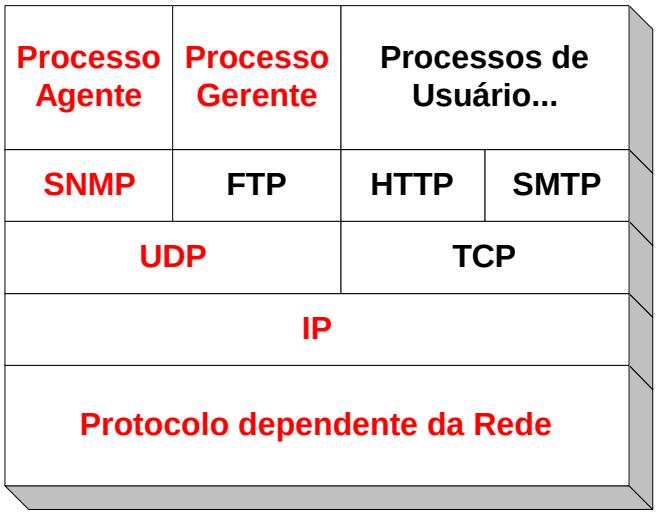
O produto throughput-atraso é importante para qualquer rede com atrasos especialmente longos ou *throughput* especialmente grande porque significa que um computador enviando dados naquela rede pode gerar um grande volume de dados antes de o destino receber o primeiro bit.

4 SIMPLE NETWORK MANAGEMENT PROTOCOL - SNMP

Segundo Klauck (1999), o SNMP foi desenvolvido nos anos 80 como resposta para os problemas de gerenciamento em ambiente TCP/IP Internet, envolvendo redes heterogêneas. Inicialmente foi concebido para ser apenas uma solução provisória até o completo desenvolvimento de um protocolo de gerenciamento mais completo, o CMIP. Neste contexto, sem um protocolo melhor disponível, o SNMP passou a ser o protocolo mais utilizado.

O SNMP é um protocolo da camada de aplicação, como mostrado na figura 2, desenvolvido para facilitar a troca de informações de gerenciamento entre dispositivos de rede. Estas informações transportadas pelo SNMP (como pacotes por segundo e taxa de erro na rede), permitem aos administradores da rede gerenciar o desempenho da rede de forma remota, encontrando e solucionando os problemas da rede, e planejar o crescimento da rede.

Figura 2 - Protocolo SNMP sobre as camadas do TCP/IP.



FONTE: Klauck (1999)

4.1 DESCRIÇÃO DE SNMP

Segundo Comer (1999), o SNMP ajuda administradores de rede a localizar e corrigir problemas em uma interligação em redes TCP/IP. Um administrador chama um gerente SNMP em seu computador local (geralmente uma estação de trabalho) e utiliza o gerente para contatar um ou mais agentes SNMP executados em máquinas remotas (normalmente

gateways). O SNMP emprega um paradigma de busca e armazenamento *(fetch-store paradigm)* no qual cada agente mantém um conjunto de variáveis conceituais (objetos) que incluem estatísticas simples, tais como contagem de pacotes recebidos, e variáveis complexas que correspondem à estrutura de dados do TCP/IP, como cache ARP e tabelas de roteamento IP. As mensagens do SNMP especificam que o agente deve buscar valores em variáveis ou armazenar valores em variáveis, e o agente converte as solicitações para operações equivalentes sobre estruturas de dados locais. Como o protocolo não abrange outras operações, todo o controle precisa ser realizado por intermédio do paradigma de busca e armazenamento. Além do protocolo SNMP um padrão à parte, referente a uma MIB, define o conjunto de variáveis que agentes SNMP mantêm, bem como a semântica de cada variável. Variáveis da MIB registram o estado de cada rede conectada, estatísticas de tráfego, contagens de erros encontrados e os conteúdos correntes de estruturas de dados internas, como a tabela de roteamento IP da maquina.

Segundo Specialski (1999), o modelo arquitetural SNMP é uma coleção de estações de gerenciamento e elementos de rede. As estações de gerenciamento executam aplicações de gerenciamento que monitoram e controlam os elementos de rede. Os elementos de rede são equipamentos tais como hospedeiros, *gateways*, servidores de terminais, e similares, que possuem agentes de gerenciamento, responsáveis pela execução das funções de gerenciamento de rede, requisitadas pelas estações de gerenciamento. O protocolo SNMP é usado para transportar a informação de gerenciamento entre as estações de gerenciamento e os agentes existentes nos elementos de rede.

O modelo proposto busca minimizar o número e a complexidade de funções de gerenciamento realizadas pelos agentes de gerenciamento. As razões que tornam este objetivo atrativo são:

a) custo de desenvolvimento do software de agente de gerenciamento, necessário para suportar o protocolo é significativamente reduzido;
b) grau de funcionalidade suportado remotamente é proporcionalmente aumentado, à medida que se aumenta a utilização dos recursos da internet na tarefa de gerenciamento;

c) a quantidade de funções de gerenciamento, que são suportadas remotamente, é gradativamente aumentada, através da imposição de algumas restrições sobre a forma e sofisticação das ferramentas de gerenciamento;
d) conjuntos simplificados de funções de gerenciamento são facilmente entendidos e utilizados pelos desenvolvedores de ferramentas de gerenciamento de redes.

O segundo objetivo do protocolo é que o paradigma funcional para monitoração e controle deve ser suficientemente extensível para acomodar aspectos adicionais, e possivelmente não previstos, da operação e gerenciamento de redes.

O terceiro objetivo é que a arquitetura deve ser, tanto quanto possível, independente da arquitetura e dos mecanismos de hospedeiros e *gateways* particulares.

Segundo Comer (1999), o SNMP define a sintaxe e o significado das mensagens que gerentes e agentes intercambiam. Ele utiliza a ASN.1 para especificar tanto o formato de mensagens como nomes de variáveis da MIB. Assim, ao contrário da maioria dos protocolos TCP/IP, as mensagens do SNMP não possuem campos fixos e não podem ser definidas com estruturas fixas.

Segundo Klauck (1999), hoje o SNMP é o protocolo mais implementado nos produtos comerciais de gerenciamento assim como em universidades e organizações de pesquisa. Isso graças ao fato de o SNMP ser um protocolo relativamente simples de ser implementado.

A maior vantagem em usar o SNMP é o fato de que sua implementação é relativamente simples. Esta simplicidade torna mais fácil para o usuário programar as variáveis que gostaria de gerenciar. Para o SNMP cada variável consiste das seguintes informações:

a) um título que identifica a variável;
b) a estrutura de dado da variável;
c) status da variável;
d) valor da variável.

Outra vantagem do SNMP é a sua vasta utilização atualmente. Esta popularidade deve-se ao fato de que até agora nenhum outro protocolo melhor apareceu para substituir a

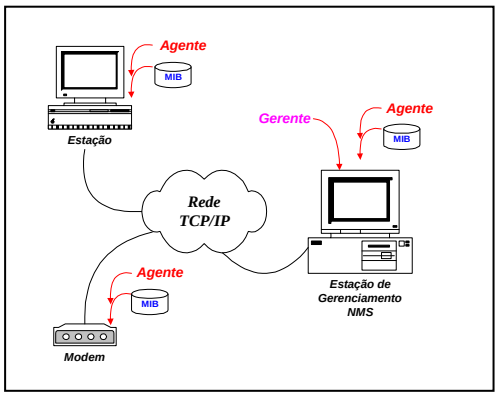
implementação "provisória" do SNMP. O resultado disso é que a maioria dos vendedores de hardware/software para redes desenvolvem seus produtos para suportar o SNMP como protocolo de gerenciamento padrão.

A grande expansibilidade é outro benefício do SNMP. A relativa facilidade de implementação torna também mais fácil a atualização deste protocolo, visando atender as novas necessidades dos usuários de redes.

4.2 ELEMENTOS DO SNMP

Segundo Klauck (1999), no modelo SNMP, os recursos de uma rede são representados como objetos. Cada objeto é, essencialmente, uma variável que representa um aspecto do dispositivo gerenciado. Todos os objetos (variáveis) são armazenados na MIB.

Figura 3 - Elementos básicos do SNMP



FONTE: Klauck (1999)

4.2.1 AGENTES

Segundo Klauck (1999), no modelo de gerenciamento SNMP, *hosts*, pontes, roteadores e *hubs* devem ser equipados com agentes SNMP para que possam ser gerenciados pelas NMS através do gerente SNMP. O agente responde a requisições da estação de gerenciamento, que pode ser o envio de informações de gerência ou ações sobre as variáveis do dispositivo onde está.

O funcionamento desta estrutura só é possível graças ao acesso direto a MIB que o agente possui, pois todas as informações de gerência encontram-se lá. Ao receber uma mensagem SNMP do gerente, o agente identifica que operação está sendo requisitada e

qual(is) a(s) variável(is) relacionadas, e a partir daí executa a operação sobre a MIB; em seguida, monta uma nova mensagem de resposta, que será enviada ao gerente.

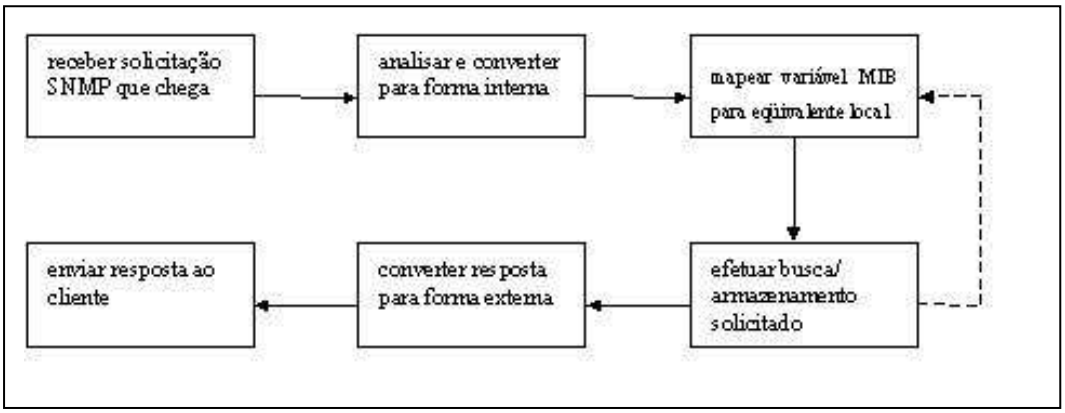
À primeira vista a comunicação do agente com o gerente pode parecer injusta, uma vez que o agente apenas responde ao que lhe é questionado. Mas há momentos em que o agente pode "falar" com o gerente sem que antes seja questionado. Isso ocorre quando o agente detecta, a partir da análise do contexto da MIB, alguma situação inesperada. Neste momento, o agente gera uma mensagem especial, o TRAP, e a envia ao gerente, relatando sobre a situação.

Cabe ao agente um papel fundamental em todo o processo de gerenciamento da rede, acessando e disponibilizando informações de gerência contidas na MIB, além de indicar situações inesperadas de funcionamento do dispositivo ao qual gerencia através do envio de TRAPs ao gerente.

4.2.1.1 ORGANIZAÇÃO DE UM AGENTE E MAPEAMENTO DE NOMES

Segundo Comer (1999), um agente SNMP precisa aceitar uma solicitação que chega, executar a operação especificada e retornar uma resposta. O entendimento dos aspectos básicos de como agentes processam mensagens é importante por que ajuda a explicar o software de mapeamento apresentado ao longo do restante deste capítulo. A figura 4 ilustra a circulação de uma mensagem por um agente SNMP.

Figura 4 - A circulação de uma mensagem do SNMP por um agente.



FONTE: Comer (1999)

Como a figura 4 indica, o agente primeiro analisa a mensagem e efetua a conversão para a forma interna. Depois, ele mapeia a especificação da variável da MIB para o item de dados local responsável pelo armazenamento das informações necessárias e efetua a operação de busca ou armazenamento. Nas operações de busca, ele preenche a área de dados da mensagem do SNMP com o valor buscado. Caso a mensagem especifique diversas variáveis, o agente percorre o terceiro e o quarto passos para cada uma. Finalmente, depois de todas as operações terem sido efetuadas, o agente converte a resposta da forma interna para a forma externa e retorna a mesma ao cliente.

4.2.2 GERENTES

Segundo Klauck (1999), servindo de interface entre as aplicações de gerência correntes no NMS e os agentes espalhados pelos dispositivos da rede está o gerente. Cabe ao gerente enviar comandos aos agentes, solicitando informações sobre variáveis de um objeto gerenciado, ou modificando o valor de determinada variável.

Os gerentes então processam estas informações colhidas pelos agentes, e as repassam à aplicação que as requisitou. A comunicação entre o gerente e as aplicações é possível através da utilização das APIs do gerente SNMP pelo sistema.

A API é um conjunto de funções que intermediam a execução de comandos entre um programa e outro, de forma a simplificar a um programa o acesso a funções do outro programa, e que no caso do SNMP intermediam as execuções entre uma aplicação de gerência e o gerente SNMP.

Quando uma solicitação da aplicação de gerência chega ao gerente, através da API, o gerente mapeia esta solicitação para um ou mais comandos SNMP que, através da troca de mensagens apropriadas, chegarão aos agentes correspondentes. Devemos ressaltar que este comando SNMP montado pelo gerente pode ser enviado a um outro gerente, que pode ou não repassá-lo a um agente. Só que a comunicação gerente-gerente só é possível na versão estendida do SNMP, o SNMPv2, e não faz parte do SNMP padrão.

Cabe também ao gerente encaminhar à aplicação de gerência os TRAPs que porventura sejam enviados pelos agentes. Assim, o software de gerência terá conhecimento da

presença de um novo equipamento na rede ou do mau funcionamento de algum dos dispositivos da rede.

Quando da inicialização do software de gerência, nada se sabe sobre a configuração ou funcionamento da rede. Essas informações vão sendo conhecidas através de TRAPs que são enviados pelos agentes; a partir daí o gerente realiza *polling* a fim de manter a comunicação com os agentes, possibilitando ao software de gerência mapear, monitorar e controlar a rede.

4.2.3 MANAGEMENT INFORMATION BASE - MIB

Segundo Specialski (1999), a MIB é uma coleção estruturada de objetos gerenciados. Objetos gerenciados representam os recursos sujeitos ao gerenciamento. Cada nodo do sistema de gerenciamento mantém uma MIB que reflete o estado dos recursos gerenciados naquele nodo. Uma entidade de gerenciamento pode monitorar os recursos de um nodo, lendo os valores dos objetos na MIB e pode controlar os recursos de um nodo, modificando estes valores.

Os nomes para todos os tipos de objetos contidos na MIB são definidos explicitamente na MIB padrão Internet ou em outros documentos que seguem as convenções de nomeação definidas na SMI. A SMI requer que todos os protocolos de gerenciamento definam mecanismos para identificar instâncias individuais dos tipos de objetos de um elemento de rede particular.

4.2.3.1 ESTRUTURA DA MIB

Segundo Klauck (1999), todos os objetos gerenciados no modelo SNMP estão armazenados e dispostos numa estrutura hierárquica, mais conhecida como **árvore**. As folhas da árvore são os objetos gerenciados atuais, onde cada um representa algum recurso, atividade ou informação relacionada que deve ser gerenciada. A própria estrutura em árvore define o agrupamento de objetos em conjuntos relacionados.

Para cada tipo de objeto na MIB é associado um identificador de tipo ASN.1, o *Object Identifier*. O identificador serve para nomear o objeto. Como o valor associado com o tipo

Object Identifier é hierárquico, a nomeação também serve para identificar a estrutura dos tipos de objeto.

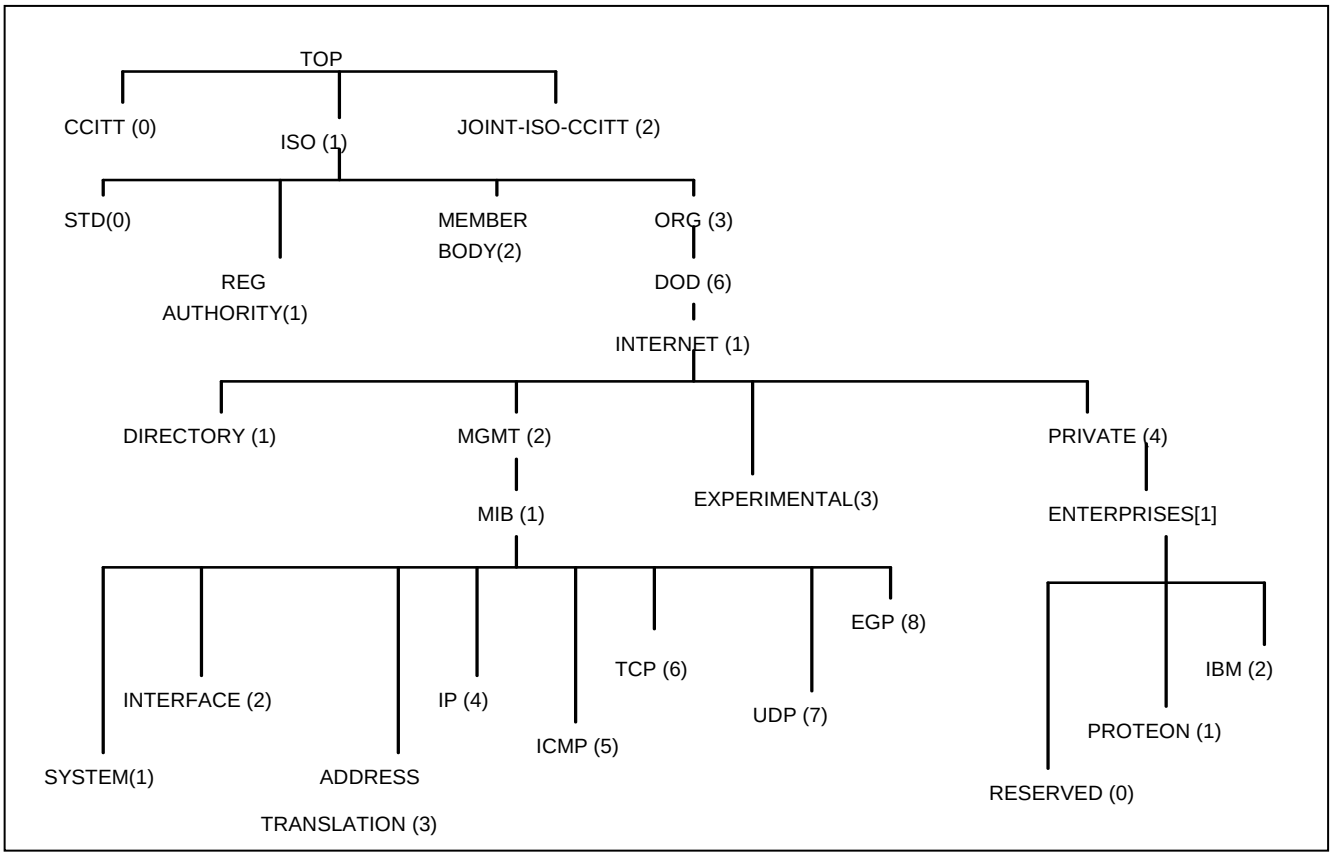
O identificador é único para cada tipo de objeto. Seu valor consiste numa seqüência de inteiros. Iniciando pela raiz da árvore, cada parte inteira componente do valor do identificador representa um ramo da árvore. A partir da raiz, encontramos três ramos principais no primeiro nível da árvore: **iso**, **ccitt** e uma junção **iso-ccitt**. Sob o ramo iso uma sub-árvore é utilizada por outras organizações (org), uma das quais é o *U.S. Departament of Defense* (DoD).

A especificação MIB define os grupos de variáveis necessárias à monitoração e controle de vários componentes da rede. Para o SNMP, nem todos os grupos de variáveis definidas pela especificação MIB são obrigatórios. Um dos mais importantes e obrigatório para o SNMP é o grupo Internet

Segundo Specialski (1999), a sub-árvore MGMT contém a definição das bases de informação de gerenciamento que foram aprovadas pelo *Internet Architecture Board* (IAB). Atualmente, existem duas versões da MIB: mib-1 e mib-2. A mib-2 é uma extensão da primeira. As duas possuem o mesmo identificador na sub-árvore porque apenas uma das duas estará presente em qualquer configuração.

A figura 5 mostra a árvore de registro utilizada para nomeação de objetos definidos na MIB.

Figura 5 - Árvore de registro de tipos de objetos



FONTE: Specialski (1999)

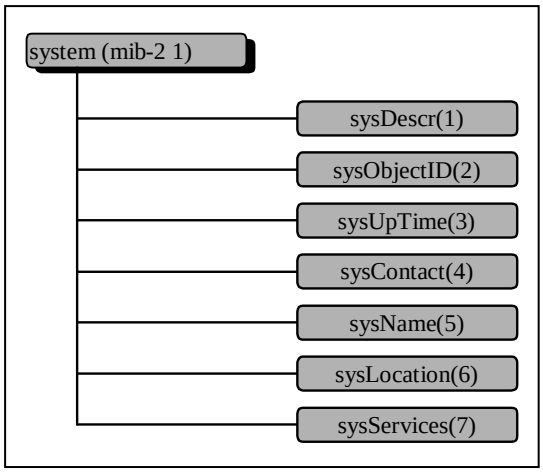
Os objetos da mib-2 são subdivididos nos seguintes grupos:

a) *system*: informações gerais sobre o sistema;
b) *interfaces*: informações sobre cada uma das interfaces do sistema para a sub-rede;
c) at: descreve a tabela de translação de endereços para mapeamento de endereços internet para endereços de sub-rede;
d) *ip*: informação relativa a experiências de implementação e execução do protocolo *Internet Protocol* (IP) no sistema;
e) *icmp*: informação relativa a experiências de implementação e execução do protocolo *Internet Control Message Protocol* (ICMP) no sistema;
f) *tcp*: informação relativa a experiências de implementação e execução do protocolo *Transmission Control Protocol* (TCP) no sistema;
g) *udp*: informação relativa a experiências de implementação e execução do protocolo *User Datagram Protocol* (UDP) no sistema;

h) *egp*: informação relativa a experiências de implementação e execução do protocolo *External Gateway Protocol* (EGP) no sistema;
i) *transmission*: fornece informações sobre esquemas de transmissão e protocolos de acesso em cada interface do sistema;
j) *snmp*: informação relativa a experiências de implementação e execução do protocolo *Simple Network Management Protocol* (SNMP) no sistema.

A organização em grupos é conveniente porque os objetos são organizados de acordo com as funções das entidades gerenciadas e também porque ela oferece um guia para os implementadores de agentes, no sentido de identificar quais objetos devem ser implementados. Se a semântica de um grupo for aplicável para uma determinada implementação, então todos os objetos do grupo devem ser implementados. Por exemplo, uma implementação deve incluir todos os objetos do grupo TCP se e somente se ela implementa o protocolo TCP; portanto, uma *bridge* ou um *router* não necessita implementar os objetos do grupo TCP. Uma exceção a esta regra é o grupo *at*. A figura 6 ilustra a estrutura do grupo *system* e a tabela 1 fornece a sintaxe do objeto, a forma de acesso permitida e uma descrição sucinta da semântica.

Figura 6 - Grupo *system* da MIB-II



FONTE: Specialski (1999)

Tabela 1 - Objetos do grupo *system* da MIB-II

Objeto	Sintaxe	Acesso	Semântica
SysDescr	*DisplayString* (*Size* (0..255)	RO	Descrição de uma entidade(hardware, sistema operacional, etc.)
SysObjectID	*Object Identifier*	RO	Identificação do subsistema contido na entidade
SysUpTime	*TimeTicks*	RO	Tempo decorrido desde a última reinicialização
SysContact	*DisplayString* (*Size* (0..255)	RW	Identificação da pessoa de contato para este nodo gerenciado
SysName	*DisplayString* (*Size* (0..255)	RW	Nome atribuído administrativamente para este nodo
SysLocation	*DisplayString* (*Size* (0..255)	RW	Localização física do nodo
SysServices	*Integer* (0..127)	RO	Valor indicando o conjunto de serviços oferecidos pela entidade

FONTE: Specialski (1999)

4.2.3.2 TIPOS DE OBJETOS DE UMA MIB.

Segundo Mello (2000), os objetos da MIB têm um tipo específico de valores. O SNMP define vários tipos primitivos de objetos, como:

a) **de texto**: define-se o tipo "*DisplayString*" que pode conter informação textual arbitrária (normalmente para um máximo de 256 caracteres). O texto deve conter somente caracteres de impressão. Exemplos deste tipo de objeto incluem valores "*sysDescr*" e "*ifDescr*". Este tipo de objeto é bastante utilizado nas MIBs;

b) **contadores**: define-se o tipo "*Counter*" que é um valor numérico que só pode aumentar. Este é o tipo mais comum de objeto de SNMP na MIB padrão e inclui objetos como "*ipInReceives*", "*ipOutRequests*", e "*snmpInPkts*". Contadores ultrapassam o valor máximo da variável, mas nunca pode ser negativo;

c) **de medida**: define-se do tipo "*Gauge*" que é um valor numérico que pode aumentar ou pode diminuir. Este tipo é encontrado em apenas algumas localizações dentro da MIB padrão. Exemplos deste tipo de objeto incluem o objeto "*tcpCurrEstab*";

d) **inteiros**: define-se do tipo "*Integer*" que pode conter valores positivos ou negativos. Este valor normalmente é fornecido por valores do tipo "*Counter*" ou "*Gauge*", mas às vezes é expresso em MIB de equipamentos fabricantes;

e) **enumerados**: define-se um tipo "*Enumerated Value*" que associa um campo textual com um valor numérico. Este tipo é bastante comum na MIB padrão e inclui objetos como "*ifAdmimStatus*", cujo valores enumerados são "*up(1)*", "*down(2)*", e "*testing(3)*". Geralmente os valores enumerados são formatados da seguinte forma "nome(número)";

f) **tempo**: define-se um tipo "*TimeTicks*" que representa um tempo decorrido. Este tempo sempre tem uma resolução de um centésimo de segundo, até mesmo quando esta resolução não é usada. Administradores de rede freqüentemente formatam este valor de tempo como "HH:MM:SS:cc" para exibição. Por exemplo, o valor "*sysUpTime*" indica o tempo decorrido desde que um dispositivo foi reiniciado;

g) **objeto**: define-se um tipo "*Object*" que pode conter o identificador para outro objeto SNMP. Se o objeto nomeado é compilado na MIB, o nome geralmente é apresentado como mesmo nome no objeto SNMP. Um exemplo deste tipo de objeto é a variável "*sysObjectID*" da MIB;

h) **endereço IP**: define-se um tipo "*IP address*" que contém o endereço IP de um dispositivo de rede. Este tipo de objeto é exibido freqüentemente como um endereço IP convencional. Exemplos deste tipo de objeto incluem "*ipRouteDest*", "*ipRouteNextRop*" e "*ipRouteMask*";

i) **endereço físico**: define-se um tipo "*Physical address*", que contém o endereço físico de um dispositivo de rede. Gerentes exibem freqüentemente este tipo de objeto como uma sucessão de valores hexadecimal, precedidos pela palavra chave "*hex:*", e separados através de dois pontos. Exemplos deste tipo de objetos incluem o objeto "*ifPhysAddress*";

j) **string**: define-se um tipo "*String*", que contém uma tabela de valores arbitrários. Quando a cadeia de caracteres contém só caracteres ASCII, os gerentes exibem este valor como uma *string* de texto. Caso contrário, gerentes exibem este tipo como uma sucessão de valores hexadecimal, precedidos pela palavra chave "*hex:*",

e separados através de dois pontos. Este tipo de valor é incomum, mas ocasionalmente encontrados em MIBs proprietárias;

k) **tabela**: o tipo "*Table*" é um objeto que contém entradas da tabela. Este tipo de objeto sempre é um nome intermediário que contém um diretório de "Entrada" e que contém vários objetos da tabela;

l) **auxiliares**: o tipo "*Branch*" é um objeto que contém tipos adicionais, tabelas, ou qualquer tipo de objeto listado acima.

4.2.3.3 ACESSO A VALORES DA MIB

Segundo Mello (2000), cada objeto SNMP é definido para ter um tipo de acesso "somente leitura", "leitura escrita" ou "apenas escrita". Isso determina se o usuário pode ler o valor de objeto, pode ler e pode escrever o objeto (com um comando "*SET*") ou só pode escrever o objeto.

Antes que qualquer objeto possa ser lido ou possa ser escrito, o nome comunitário do SNMP deve ser conhecido. Estes nomes comunitários são configurados pelo administrador, e podem ser vistos como senhas necessárias para anexar dados ao SNMP. No sentido exato, nomes comunitários existem para permitir que partes da MIB no SNMP, e subconjuntos de objetos sejam referenciados. Como o termo "Comunitário" é requerido, espera-se que o verdadeiro propósito destes valores seja identificar comunitariamente entre objetos SNMP configurados. Porém é prática comum fazer estes nomes comunitários limitarem o acesso da capacidade do SNMP para usuários sem permissão.

4.3 OPERAÇÕES SNMP APLICADAS ÀS VARIÁVEIS DA MIB

Neste tópico são apresentados, conceitualmente, os mecanismos de busca e atualização de informações nas variáveis da MIB, referentes à primeira versão do protocolo SNMP.

4.3.1 GET

Segundo Klauck (1999), o gerente SNMP envia o comando GET a um determinado agente toda vez que necessita recuperar uma informação de gerenciamento específica do

objeto gerenciado pelo agente. Estas informações encontram-se na forma básica de variáveis, que por sua vez estão na MIB do elemento de rede gerenciado. Logo, o comando GET, na verdade, solicita ao agente a leitura de determinada(s) variável (is) da MIB em questão.

4.3.2 GETNEXT

Segundo Klauck (1999), o comando GETNEXT assemelha-se ao comando GET, no entanto, enquanto o comando GET solicita ao agente a leitura de determinada instância de uma variável, ao receber um comando GETNEXT, o agente deve ler a próxima instância disponível, na ordem da MIB, da(s) variável (is) associadas.

Esta operação é especialmente poderosa na recuperação de uma tabela de instâncias da MIB, cujo tamanho seja desconhecido. Para isto, é inicialmente especificado no comando GETNEXT o nome da tabela, o que retornará como resposta o primeiro item (neste caso, uma instância de variável) da tabela. Com o nome do primeiro item da tabela, o gerente pode enviar um outro GETNEXT, desta vez passando a resposta do GETNEXT anterior como parâmetro, a fim de recuperar o próximo item da tabela, e assim sucessivamente até recuperar todas as instâncias da variável na tabela. Para este caso, se o comando GET fosse utilizado ele falharia, pois o nome da tabela não corresponde a uma instância individual da variável.

4.3.3 SET

Segundo Klauck (1999), a operação SET requisita a um determinado agente a atribuição/alteração do valor de determinada(s) variável (is) de uma MIB. Alguns desenvolvedores acreditam que este comando não deve retornar um RESPONSE. Outros acham que a operação SET deve retornar alguma indicação de que a operação foi efetuada. Porém o mais correto seria que após cada operação SET sobre uma variável, uma operação GET fosse efetuada sobre a mesma variável a fim de assegurar que a operação SET foi efetuada.

4.3.4 TRAP

Segundo Klauck (1999), a operação TRAP difere de todas as outras. Ela é utilizada por um agente SNMP para notificar de forma assíncrona a um gerente algum evento extraordinário que tenha ocorrido no objeto gerenciado.

Diversos questionamentos são feitos quanto a esta operação. Talvez o maior deles seja: Quais eventos devem realmente ser notificados ao gerente? Embora todos concordem que o gerente deva ser informado de alguns eventos significativos, muitos fornecedores de produtos que implementam o SNMP trazem TRAPs específicos, muitos deles desnecessários.

4.3.5 RESPONSES

Segundo Klauck (1999), como já descrito, sempre que um agente recebe um comando GET, GETNEXT ou SET ele tenta executar a operação associada e, conseguindo ou não, constrói uma outra mensagem que é enviada ao emissor da requisição. Esta mensagem é a GETRESPONSE. Das operações SNMP, apenas o TRAP não gera um RESPONSE.

4.4 STRUCTURE OF MANAGEMENT INFORMATION - SMI

Segundo Klauck (1999), a Estrutura da Informação de Gerenciamento (SMI) é especificada na RFC 1155 e define a estrutura através da qual uma MIB pode ser definida e construída. A SMI define os tipos de dados que devem ser usados na MIB e especifica como os recursos na MIB são representados e nomeados.

A SMI evita tipo de dados complexos para simplificar a tarefa de implementação e aumentar a interoperabilidade. Inevitavelmente, as MIBs desenvolvidas por fornecedores irão conter tipo de dados particulares, a menos que restrições sejam impostas na definição de cada tipo de dado a fim de manter a interoperabilidade.

Para fornecer um caminho de padronização na representação das informações de gerenciamento, a SMI deve fornecer o seguinte:

a) padronização técnica na definição estrutural de uma MIB;
b) padronização técnica na definição de objetos individuais, incluindo a sintaxe e o valor de cada objeto;

c) padronização técnica na codificação dos valores dos objetos.

Segundo Specialski (1999), o modelo de informação de um sistema de gerenciamento, fornece a estrutura para representação, armazenamento e transferência das informações de gerenciamento. Esta estrutura é denominada SMI e, dependendo do sistema, poderá apresentar maior ou menor complexidade.

No modelo OSI, a SMI é baseada no paradigma de orientação a objetos, enfatizando as hierarquias de classe, de *containment* e de registro. Já o modelo SNMP, utiliza conceitos de tipos de dados, embora a sua nomenclatura refira-se a objetos.

Dois documentos definem a informação de gerenciamento no modelo SNMP: o RFC 1065 - Estrutura da Informação de Gerenciamento e o RFC 1066 - Base de Informação de Gerenciamento. Os dois documentos foram projetados para serem compatíveis tanto com o SNMP quanto com o modelo de gerenciamento OSI, porém, o modelo de informação de gerenciamento OSI utiliza definições mais complexas.

As funções de gerenciamento de rede podem ser agrupadas em duas categorias: monitoração de rede e controle de rede. A monitoração da rede está relacionada com a tarefa de observação e análise do estado e configuração de seus componentes; é uma função de "leitura". O controle da rede é uma função de "escrita" e está relacionada com a tarefa de alteração de valores de parâmetros e execução de determinadas ações.

4.4.1 MONITORAÇÃO DA REDE

Segundo Specialski (1999), a monitoração consiste na observação de informações relevantes ao gerenciamento. Estas informações podem ser classificadas em três categorias:

a) estática: caracteriza a configuração atual e os elementos na atual configuração, tais como o número e identificação de portas em um roteador;
b) dinâmica: relacionada com os eventos na rede, tais como a transmissão de um pacote na rede;
c) estatística: pode ser derivada de informações dinâmicas; ex. média de pacotes transmitidos por unidade de tempo em um determinado sistema.

A informação de gerenciamento é coletada e armazenada por agentes e repassada para um ou mais gerentes. Duas técnicas podem ser utilizadas na comunicação entre agentes e gerentes: *polling* e *event-reporting.*

A técnica de polling consiste em uma interação do tipo *request/response* entre um gerente e um agente. O gerente pode solicitar a um agente (para o qual ele tenha autorização), o envio de valores de diversos elementos de informação. O agente responde com os valores constantes em sua MIB.

Na técnica de *event-reporting*, a iniciativa é do agente. O gerente fica na escuta, esperando pela chegada de informações. Um agente pode gerar um relatório periodicamente para fornecer ao gerente o seu estado atual. A periodicidade do relatório pode ser configurada previamente pelo gerente. Um agente também pode enviar um relatório quando ocorre um evento significativo ou não usual.

4.4.2 CONTROLE DE REDE

Segundo Specialski (1999), esta parte do gerenciamento de rede diz respeito à modificação de parâmetros e à execução de ações em um sistema remoto. Todas as cinco áreas funcionais de gerenciamento (falhas, desempenho, contabilização, configuração e segurança), envolvem monitoração e controle. Tradicionalmente, no entanto, a ênfase nas três primeiras destas áreas, tem sido na monitoração, enquanto que nas duas últimas, o controle tem sido mais enfatizado. Alguns aspectos de controle na gerência de configuração e de segurança são apresentados a seguir.

O controle de configuração inclui as seguintes funções:

a) definição da informação de configuração, recursos e atributos dos recursos sujeitos ao gerenciamento;
b) atribuição e modificação de valores de atributos;
c) definição e modificação de relacionamentos entre recursos ou componentes da rede;
d) inicialização e terminação de operações de rede;
e) distribuição de software;

f) exame de valores e relacionamentos;
g) relatórios de *status* de configuração.

O controle de segurança é relativo à segurança dos recursos sob gerenciamento, incluindo o próprio sistema de gerenciamento. Os principais objetivos em termos de segurança, são relativos a confidencialidade, integridade e disponibilidade. As principais ameaças à segurança referem-se à interrupção, interceptação, modificação e mascaramento.

As funções de gerenciamento de segurança podem ser agrupadas em três categorias:
a) manutenção da informação de segurança;
b) controle de acesso aos recursos;
c) controle do processo de criptografia.

4.5 O SNMP SOBRE A CAMADA DE TRANSPORTE

Segundo Klauck (1999) e Mello (2000), o protocolo SNMP foi desenvolvido para rodar sobre a camada de transporte, na camada de aplicação da pilha de protocolo TCP/IP. A maioria das implementações do SNMP utilizam o UDP, como protocolo de transporte. O UDP é um protocolo não-confiável, não garantindo a entrega, a ordem ou proteção contra duplicação das mensagens.

O SNMP utiliza o UDP, pois foi desenvolvido para funcionar sobre um serviço de transporte sem conexão. A maior razão para isto é o desempenho. Utilizando um serviço orientado à conexão, como o TCP, a execução de cada operação necessitaria de uma prévia conexão, gerando um *overhead* significativo, o que é inaceitável na atividade de gerência onde as informações devem ser trocadas entre as entidades da forma mais rápida possível.

Segmentos UDP são transmitidos em datagramas IP. O cabeçalho UDP inclui os campos de origem e destino, e um *checksum* opcional que protege o cabeçalho UDP e os dados de usuário.

Duas portas UDP são associadas às operações SNMP. As operações GET, GETNEXT e SET utilizam a porta 161. Por ser acionada em casos de exceção, a operação TRAP têm reservada para si a porta 162.

Como o UDP é um protocolo não-confiável, é possível que mensagens SNMP sejam perdidas. O SNMP por si só não fornece garantia sobre a entrega das mensagens. As ações a serem tomadas quando da perda de uma mensagem SNMP não são abordadas pelo padrão. No entanto, algumas considerações podem ser feitas sobre a perda de mensagens SNMP. No caso de uma mensagem de GET ou GETNEXT, a estação de gerenciamento deve assumir que esta mensagem (ou o GETRESPONSE correspondente) tenha sido perdida se não receber resposta num certo período de tempo. A estação de gerenciamento então pode repetir a mensagem uma ou mais vezes até que obtenha a resposta. Desde que um único *request-id* (que identifica a mensagem) seja utilizado para cada operação distinta, não haverá dificuldade em identificar mensagens duplicadas.

No caso de uma mensagem de SET, a recuperação deve provavelmente envolver o teste no objeto associado à operação através de uma GET sobre este mesmo objeto a fim de determinar se a operação SET foi ou não efetivada.

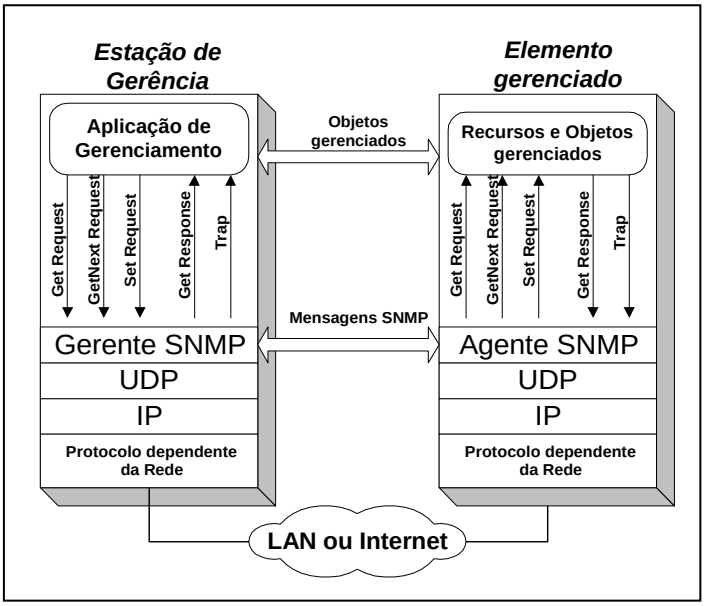
Uma vez que a operação TRAP não gera uma mensagem de resposta, não é fácil identificar a perda de uma mensagem desse tipo.

No SNMP o gerente executa o gerenciamento através da utilização do protocolo SNMP, que é implementado no topo das camadas UDP, IP e do protocolo dependente do tipo de rede (Ethernet, FDDI, X.25, por exemplo).

O SNMP requer alguns serviços da camada de transporte para que suas mensagens sejam transmitidas entre as entidades de gerenciamento. Isso independe da implementação da camada de transporte.

A figura 7 ilustra o contexto do protocolo SNMP na pilha de protocolo TCP/IP, utilizando o UDP como protocolo de conexão.

Figura 7 - Protocolo SNMP sobre a camada de transporte



FONTE: Klauck (1999)

4.5.1 SERVIÇOS EXIGIDOS PELO SNMP

Segundo Klauck (1999) e Mello (2000), rodando na camada de Aplicação da arquitetura TCP/IP, como mostrado na figura 7, o SNMP encontra-se acima das camadas de transporte e rede. Para que o SNMP possa funcionar adequadamente, estas camadas (transporte e rede) devem fornecer ao menos cinco serviços que são de vital importância à movimentação das mensagens SNMP através da rede:

a) roteamento: a camada de rede é quem fornece as funções de roteamento, as quais aperfeiçoam toda a utilidade do gerenciamento da rede, dando a possibilidade de "re-rotear" os pacotes em torno de áreas da rede com falhas. Isso permite que o gerenciamento da rede continue operando mesmo após falha em alguma(s) máquina(s) na rede;
b) independência do meio: permite ao SNMP gerenciar diferentes tipos de elementos da rede, de forma independente. Caso o SNMP fosse construído sobre a camada de enlace, estaria preso a esta implementação de enlace específica, desse modo o

SNMP não poderia gerenciar um outro dispositivo que implementasse outro protocolo de enlace, o que degradaria o gerenciamento da rede;

c) fragmentação e remontagem: este serviço está relacionado com a independência do meio. As mensagens SNMP podem ser fragmentadas em pacotes, transmitidas pelo meio e remontadas no destino. O protocolo IP permite que os pacotes SNMP trafeguem pelo meio com diferentes tamanhos. A fragmentação e a remontagem reduzem a robustez geral do gerenciamento da rede, pois se qualquer fragmento for perdido pelo caminho, toda a operação irá falhar;
d) *checksum* ponto-a-ponto: o *checksum* ponto-a-ponto é um serviço de checagem de erros fornecidos pela camada de transporte que permite a uma entidade conferir a integridade dos dados recebidos através da rede, aumentando a confiabilidade da transmissão;
e) multiplexação/demultiplexação: os serviços de multiplexação e demultiplexação são fornecidos pelo protocolo de transporte. Estes serviços facilitam em muito os possíveis relacionamentos de gerenciamento com o SNMP, permitindo que várias aplicações SNMP possam “utilizar” serviços da camada de transporte.

5 TECNOLOGIAS E FERRAMENTAS UTILIZADAS

Neste capítulo é feita uma explanação sobre as tecnologias e ferramentas utilizadas para o desenvolvimento do presente trabalho.

5.1 UNIFIED MODELING LANGUAGE - UML

Segundo Furlan (1998), a UML é uma tentativa de padronizar a modelagem orientada a objetos de forma que qualquer sistema seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível.

A UML é a linguagem padrão para especificar, visualizar, documentar e construir artefatos de um sistema e pode ser utilizada como todos os processos ao longo do ciclo de desenvolvimento e através de diferentes tecnologias de implementação.

Diante da diversidade de conceitos das diversas metodologias de orientação a objetos, Grady Booch, James Rumbaugh e Ivar Jacobson decidiram criar uma Linguagem de Modelagem Unificada.

Existem cinco fases no desenvolvimento de sistemas de software em UML: análise de requisitos, análise, projeto, programação e testes. Estas cinco fases não devem ser executadas nesta ordem, mas concomitantemente de forma que problemas detectados numa certa fase modifiquem e melhorem as fases desenvolvidas anteriormente de forma que o resultado global gere um produto de alta qualidade e performance.

Na analise de requisitos, é feita a captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através do uso de funções chamadas caso de uso.

A fase de análise está preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classes são modeladas e ligadas através de relacionamentos com outras classes e são descritas no Diagrama de Classes.

Na fase de projeto, o resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover uma infra-estrutura técnica: a interface do usuário e de

periférico, gerenciamento de banco de dados, comunicação entre outros sistemas, dentre outros.

Na fase de programação, as classes provenientes do projeto são convertidas para o código da linguagem orientada a objetos

5.1.1 DIAGRAMAS DA UML

Segundo Furlan (1998), na UML, o modo para descrever os vários aspectos de modelagem, é através da notação definida pelos seus vários tipos de diagramas. Na UML, os principais diagramas são: diagrama de casos de uso, diagrama de classes e diagramas de seqüência.

Um diagrama de caso de uso é um gráfico de atores, um conjunto de casos incluídos por um limite de domínio, comunicação, partição e associação entre atores, assim como generalizações entre casos de uso. Como primeiro passo para a análise, definir os casos de uso que descrevam o que o sistema irá fornecer em termos de funcionalidade.

O diagrama de classes é a essência da UML, resultado de uma combinação de diagramas propostos pela *Object Modelling Technique*, Booch e vários outros métodos. A metodologia define esse diagrama como uma estrutura lógica estática em uma superfície de duas dimensões mostrando uma coleção de elementos declarativos de modelo, como classes, tipos e seus respectivos conteúdos e relações.

O diagrama de classes consiste de Objetos, Classes, Operações, Associações e Restrições.

Os diagramas de seqüência expõem o aspecto do modelo que enfatiza o comportamento dos objetos em um sistema, incluindo suas operações, interações, colaborações e histórias de estado em seqüência temporal de mensagem e representação explicita de ativação de operações;

Para maiores informações sobre a UML pode ser encontrada em Furlan (1998).

5.2 COMMON OBJECT REQUEST BROKER ARCHITECTURE - CORBA

Segundo Klauck (1999), o padrão CORBA/OMG é um conjunto de padrões e conceitos para objetos distribuídos em ambientes abertos. Segundo essa arquitetura, métodos de objetos remotos podem ser ativados de forma transparente, em ambientes distribuídos e heterogêneos, através de um ORB. No CORBA, é possível também obter interoperabilidade entre objetos desenvolvidos em linguagens diferentes de programação (C, C++, Ada Cobol, Java, Smaltalk), em que as diferenças de cada um são mascaradas pelo CORBA através do mapeamento adequado da IDL, para a linguagem de programação desejada. Cada objeto CORBA tem sua interface especificada em IDL, uma linguagem declarativa, sem nenhuma estrutura algorítmica, com sintaxe e tipos predefinidos, baseados na linguagem C++. Portanto, o uso desta linguagem permite tratar das heterogeneidades do ambiente, tais como os diferentes tipos de máquinas, sistemas operacionais e linguagens de programação.

Segundo Capeletto (1999), o CORBA, desenvolvido em 1989 pelo OMG, permite que objetos invoquem métodos em objetos distribuídos em redes como estes fossem locais. Numa implementação CORBA é necessário que haja no mínimo, dois ORB, localizados em ambos os lados do sistema cliente/servidor, para criar e gerenciar a comunicação entre os objetos. ORB's são a chave para a arquitetura CORBA de objetos distribuídos, são o *middleware* que interfaceia os componentes. Eles permitem que os objetos clientes façam requisições para o servidor sem o conhecimento de onde estes objetos estão, em qual linguagem foram implementados ou em qual sistema operacional estão sendo executados.

5.2.1 MODELO DE OBJETOS CORBA

Segundo Krupskaia (2000), a principal característica de um objeto é de dispor um serviço a um determinado cliente, onde esse cliente é qualquer entidade capaz de requisitar tal serviço do objeto. Abaixo estão relacionadas algumas semânticas existentes:

a) objetos: é uma entidade de identidades encapsuladas, que tem a capacidade de prover serviços aos clientes;
b) requisições: é o nome dado ao evento que pode ser utilizado por um cliente para solicitar um serviço de um determinado objeto. Nesta operação, deve conter

informações do tipo da operação executada, o objeto alvo, zero ou mais parâmetros, e alguns textos adicionais para a requisição;

c) criação e destruição de objetos: são determinados objetos que podem ser criados ou destruídos de acordo com as necessidades do pedido de um emissor;
d) tipos: é a identificação de uma entidade que possui características associadas ao seu valor. Os "tipos" são usados para restringirem a possibilidade de determinados parâmetros ou caracteres sejam apresentados como resultados;
e) interfaces: interface é a descrição de um conjunto de operações que podem ser requeridas por um cliente satisfazendo suas necessidades;
f) operações: uma entidade identificadora que denota os serviços que podem ser requeridos e identificados por um identificador de operações. As informações necessárias para sua ativação (envio de parâmetros, retorno de resultados, tratamento de exceções, semântica e informação de contextos) são relacionadas em sua assinatura. Na especificação CORBA, as "operações" são tratadas de forma genérica, em outras palavras, podem ser enviadas a mesma operação para vários objetos de diferentes características, tudo isso é possível, pois há uma total desvinculação dos detalhes da implementação de uma interface e também porque há utilização de herança;
g) atributos: são declarações feitas para permitir a manipulação de valores do objeto servidor. Um atributo eqüivale logicamente a declaração de funções de acesso (uma para receber o valor do atributo e outra para escolher o valor do atributo). Entretanto um atributo pode ser definido como só de leitura;
h) herança: este é um dos conceitos herdados da orientação a objeto, muito bem adotado e utilizados pela especificação CORBA. O CORBA permite utilizar herança para criar tanto novas interfaces como novos objetos. Ele implementa múltipla herança, porém não é possível herdar de duas classes que tenham mesmo nome de operação ou de atributo. E também não é possível alterar os nomes das operações ou atributos herdados nas classes derivadas, podendo apenas serem redefinidos nomes de tipos, constantes ou exceções. O CORBA ainda permite múltipla herança com metaclasses.

5.2.2 CONTEXTO DO CORBA

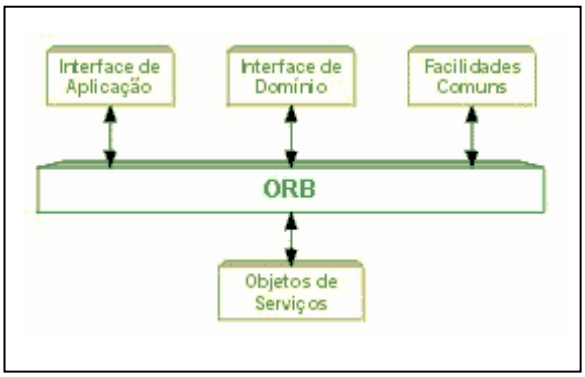
Segundo Capeletto (1999), a chave para entender a estrutura da arquitetura CORBA é o Modelo de Referência, que consiste nos seguintes componentes:

a) *object request broker*: permite aos objetos a transparência de fazer e receber solicitações e respostas em um sistema distribuído. É a fundação para construir aplicações para objetos distribuídos e possibilitar a interoperabilidade entre aplicações em ambientes heterogêneos e homogêneos;
b) serviços comuns de objetos: é uma coleção de serviços (interfaces e objetos) que suportam funções básicas para usar e implementar objetos. Serviços são necessários para construir qualquer aplicação distribuída e são sempre independentes da aplicação. Por exemplo, a gerência do ciclo de vida define convenções para criar, apagar, copiar e mover objetos, não definindo como os objetos são implementados na aplicação;
c) facilidades comuns: é uma coleção de serviços que muitas aplicações podem compartilhar, mas que não são tão fundamentais como os serviços comuns de objetos. Por exemplo, um sistema gerenciador ou correio eletrônico podem ser classificados como uma facilidade comum;
d) objetos de aplicações: são os produtos de um único fornecedor que controla suas interfaces. Os objetos de aplicações correspondem a tradicional noção de aplicações, então elas não são padronizadas pelo OMG. Então os objetos de aplicações constituem a camada mais alta do Modelo de Referência.

O *Object Request Broker*, é então, o cerne do Modelo de Referência. Ele é como uma central telefônica, disponibilizando os mecanismos básicos para fazer e receber chamadas.

A figura 8, demonstrar melhor a Arquitetura do Modelo de Referência da OMG.

Figura 8 - Arquitetura do modelo de referência da OMG.



FONTE: Krupskaia (2000)

5.2.2.1 SERVIÇOS COMUNS DE OBJETOS

Segundo Krupskaia (2000), esse tipo de objeto, nada mais é do que uma coleção de serviços (interfaces) que se apresentam independentes de qualquer domínio de aplicação e que abrangem funções básicas que poderão ser usadas para a implementação de um objeto. Na verdade, os serviços comuns de objetos podem ser visto como uma adição, um complemento das funções exercidas pelo ORB. Existem dezesseis diferentes tipos de objetos de serviço.

Uma das grandes vantagens dos serviços comuns de objetos, é que permite que aos desenvolvedores não precisem se preocupar com os serviços em nível de sistemas, e sim apenas com os seus objetos.

5.2.2.2 FACILIDADES COMUNS

Segundo Krupskaia (2000), assim como os objetos de serviços, as "facilidades comuns" também se apresentam como uma coleção de serviços (interfaces), os quais podem ser compartilhados horizontalmente em nível de usuário final, diferentemente dos objetos de serviços que possuem seus serviços direcionados ao sistema. Outra diferença quanto aos objetos de serviços, é que os serviços oferecidos por essa interface (facilidades comuns) não são considerados fundamentais.

5.2.2.3 OBJETOS DE APLICAÇÕES

Segundo Krupskaia (2000), trata-se de componentes desenvolvidos especificamente para aplicações do usuário final. Esses objetos correspondem às noções tradicionais das aplicações. Dentro deles, temos os objetos de negócio, que correspondem aos objetos que formam as aplicações compatíveis com o CORBA. A partir de um conjunto de objetos de negócio se é possível construir uma aplicação. Outro ponto importante, é que ele (objeto de negócio) manipula processos de negócios que estão presentes no mundo real. Devido a todas essas características, os objetos de aplicação não são padronizados pela OMG.

Visando o gerenciamento da complexidade da aplicação, os objetos de negócio se concentram mais nas interfaces e suas implementações.

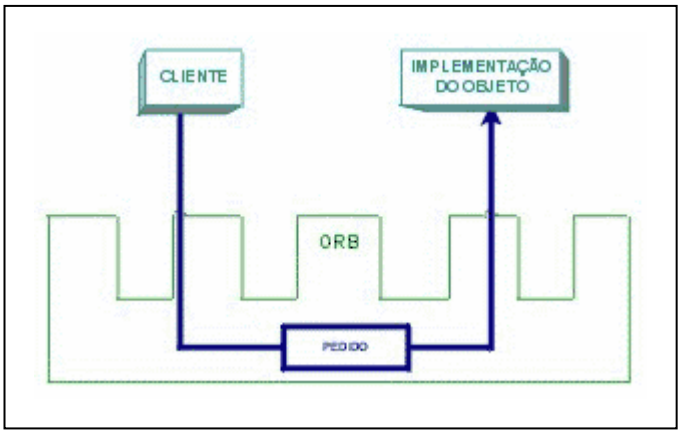
5.2.3 OBJECT REQUEST BROKER - ORB

Segundo Capeletto (1999), objetos clientes requisitam serviços às implementações de objetos através de um ORB. O ORB é responsável por todos os mecanismos requeridos para encontrar o objeto, preparar a implementação de objeto para receber a requisição, e executar a requisição. O cliente vê a requisição de forma independente de onde o objeto está localizado, em qual linguagem de programação ele foi implementado, ou qualquer outro aspecto que não está refletido na interface do objeto.

Segundo Klauck (1999), ele permite que um objeto cliente possa invocar, de modo transparente, um método em um objeto servidor, que pode estar na mesma máquina ou em outra máquina qualquer da rede. O ORB intercepta as invocações de métodos e é responsável por encontrar o objeto que implementa o método invocado, passar os parâmetros fornecidos, executar o método e retornar os resultados para o objeto cliente. O cliente não necessita conhecer onde está localizado o objeto servidor, em qual linguagem de programação ele foi implementado ou em que sistema operacional ele reside: necessita apenas, saber qual a interface do objeto (quais são os seus métodos e atributos públicos).

A figura 9 mostra um ORB repassando uma requisição para a implementação do objeto.

Figura 9 - Cliente enviando requisição através do ORB



FONTE: Klauck (1999)

Para facilitar estas requisições e prover a interoperabilidade entre os ORB's a especificação do CORBA 2.0 descreve o protocolo chamado *Internet Inter-ORB Protocol* (IIOP) que está sendo rapidamente aceito pelas empresas líderes mundiais na produção de softwares.

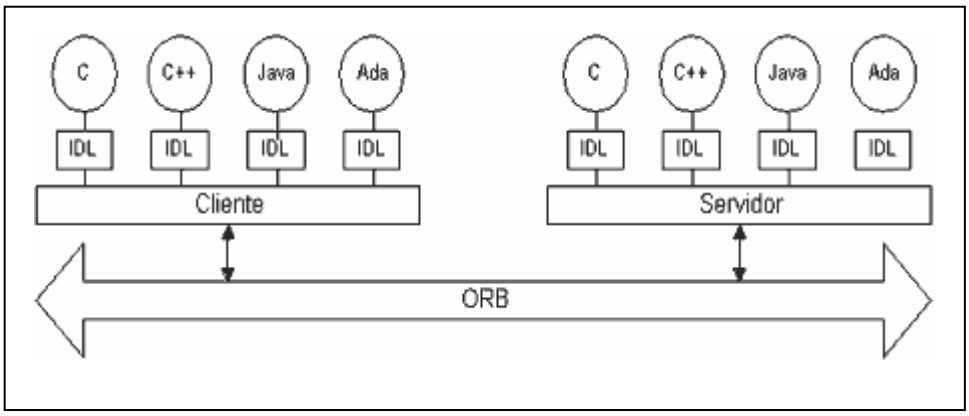
5.2.4 INTERFACE DEFINITION LANGUAGE - IDL

Segundo Capeletto (1999), a OMG IDL é a linguagem usada para descrever as interfaces que os objetos clientes solicitam e as implementações que os objetos disponibilizam. Uma definição de interface escrita em IDL define completamente a interface e especifica totalmente cada parâmetro das operações, disponibilizando toda a informação necessária para desenvolver clientes que usem as operações desta interface.

Clientes não são escritos em IDL, que é puramente uma linguagem descritiva, mas sim em linguagens nas quais já tiveram mapeamentos definidos pelo OMG. O tipo de mapeamento da IDL para uma linguagem depende das facilidades disponíveis na linguagem do cliente. Por exemplo, uma 'exception' somente poderá ser mapeada para uma linguagem que permita o tratamento de exceções. Isso garante que os componentes em CORBA sejam autodocumentáveis, permitindo que diferentes objetos, escritos em diferentes linguagens, possam interoperar através das redes e de sistemas operacionais.

A figura 10 mostra a independência da IDL em relação as linguagens de programação

Figura 10 - Mapeamento da IDL para linguagens de programação



FONTE: Capeletto (1999)

A IDL não inclui quaisquer estruturas ou variáveis algorítmicas. A gramática da IDL é um subconjunto do ANSI C++ com construções adicionais para suportar os mecanismos que invocam operações.

A IDL suporta herança. Uma interface pode ser um sub-tipo de uma outra interface, herdando todas as suas características e acrescentando características próprias.

5.2.5 A ESTRUTURA CORBA

Para visualizar melhor uma estrutura CORBA, é interessante dividi-la em duas partes distintas que são separadas pelo ORB:

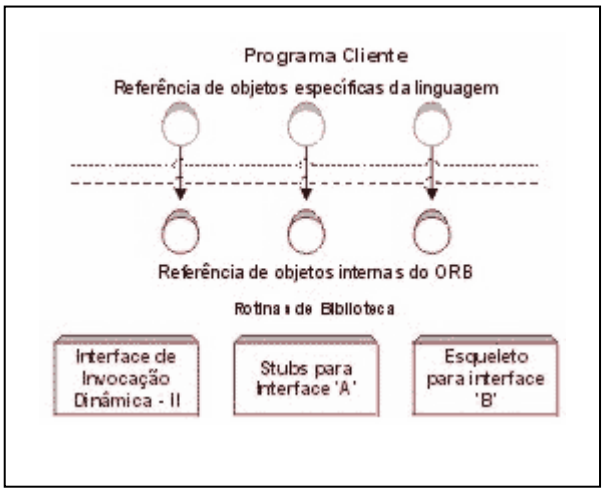
a) lado do cliente;
b) lado do servidor.

5.2.5.1 LADO DO CLIENTE

Segundo Krupskaia (2000), cliente é toda entidade (programa) responsável em invocar uma determinada operação sobre uma implementação de objeto. Como uma das características do CORBA, os detalhes utilizados (como o adaptador de objeto, ou o ORB utilizado) para que seja feito esse acesso aos serviços de um determinado objeto é transparente e dispensável do conhecimento do cliente, precisando o mesmo apenas saber interagir com a interface do tal objeto. Essa invocação feita pelo cliente, que para o mesmo aparenta ser localmente, poderá ser feita utilizando-se o *Stub*, (conjunto de rotinas de

bibliotecas que são invocadas como procedimentos e que são geradas no momento da compilação da interface), ou poderá ser feita utilizando-se de um conjunto de rotinas de invocações feitas dinamicamente através da Interface de Invocação Dinâmica (DII).

Figura 11 - Estrutura de um Objeto CORBA.



FONTE: Krupskaia (2000)

5.2.5.2 LADO DO SERVIDOR

Segundo Capeletto (1999), no lado do cliente, o máximo que se conhece das implementações dos objetos são suas definições IDL, uma para cada objeto. O cliente não consegue visualizar um servidor no outro lado do ORB. Mas no lado da implementação do objeto a estrutura da implementação é visível, e os adaptadores de objeto são equipados com mecanismo para localizar e repassar as referências do objeto alvo para o ORB.

Segundo Krupskaia (2000), o Adaptador de Objetos, é um componente da especificação CORBA que exerce o papel de intermédio na comunicação entre os objetos e o ORB, ajudando o ORB a entregar as requisições à implementação dos objetos bem como também ajuda na ativação dos objetos. Esse componente se faz necessário devido a grande diversidade existente de implementações de objetos. O Adaptador de Objetos tem a oferecer os seguintes serviços:

a) registros de implementação de objetos: é utilizado na localização da implementação do objeto;
b) geração de referência a objeto: tem a capacidade de gerar referência a objetos;

c) capacidade de invocações (em auxílio com o *Skeleton*);
d) ativar e desativar uma implementação de objeto: podem ativar ou desativar um determinado objeto quando for necessário;
e) segurança das interações: em conjunto com o ORB, deve garantir a entregar das requisições por meio de conexões múltiplas sem bloquear alguma dessas conexões de maneira indefinida.

5.2.6 A UTILIZAÇÃO DO CORBA NA GERÊNCIA DE REDES

Segundo Klauck (1999), o CORBA é um padrão cada vez mais utilizado na área de gerência de redes. Para entender a razão desta utilização é necessário, primeiramente, verificar algumas características e recursos oferecidos por este padrão.

A filosofia básica do CORBA está na separação entre a interface e a implementação do objeto. Isto possibilita que uma máquina cliente possa instanciar objetos, cuja implementação reside em outra máquina da rede. Esta característica pode ser de grande importância para o desenvolvimento de aplicações para a gerência de redes.

Aplicações de gerência de redes geralmente são complexas e exigem uma grande capacidade de memória e CPU das máquinas em que estão instaladas. Estas aplicações também devem possui um nível de segurança razoável, por acessarem recursos vitais ao pleno funcionamento da rede.

Uma solução emergente para a gerência de redes distribuída é a utilização da tecnologia *Web*, que permite a transferência automática de uma aplicação gerente (implementada como um *applet* Java) para um *browser Web* e possibilita que tal aplicação possa ser executada em qualquer plataforma, utilizando uma interface conhecida e amigável para o usuário.

Entretanto, sem a utilização de uma ferramenta como o CORBA, a criação de aplicações de gerência mais complexas, utilizando a tecnologia *Web*, fica limitada.

Com a utilização do CORBA, a interface do objeto é separada de sua implementação propriamente dita. O cliente, possuindo somente a definição da interface, pode instanciar um

objeto cuja implementação pode estar sendo executada em qualquer máquina na rede. Desta forma, a implementação de uma aplicação de gerência pode ser executada em um servidor dotado de um grande poder de processamento e, se necessário, instanciar objetos em outros servidores, distribuindo a carga de processamento entre diversas máquinas da rede. Somado a isto, a implementação do objeto pode ser realizada em praticamente qualquer linguagem de programação. A aplicação de gerência, construída como um *applet* Java, só necessita obter a definição da interface dos objetos para instanciá-los. em outras palavras, no *browser* é executado apenas o código necessário para a realização da interface com o usuário final. A implementação da aplicação de gerência pode ser executada em um, ou vários servidores distribuídos na rede.

Para maiores informações sobre esta tecnologia pode ser encontrada em Capeletto (1999), Klauck (1999) e Krupskaia (2000).

5.3 TECNOLOGIA JAVA

Segundo Comer (2001), desenvolvida pela *Sun Microsystems, Incorporated*, Java é o nome de uma tecnologia especifica usada para criar e executar documentos ativos. Java usa o termo *applet* para descrever programas de documento ativos e programas de computador convencionais. A tecnologia Java inclui três componentes chave relacionados a *applets*:

a) linguagem de programação: Java inclui uma nova linguagem de programação que pode ser usada como linguagem fonte para programas convencionais de computador ou para *applets* Java;
b) ambiente de execução (*Runtime Environment*): o sistema Java define um ambiente de execução que fornece as facilidades necessárias para executar um programa Java;
c) biblioteca de classes: para facilitar a produção de *applets*, Java fornece uma biblioteca grande que contém software para executar muitas das tarefas que um *applet* executa.

Na prática, pode ser difícil distinguir entre os componentes porque eles foram projetados para trabalhar juntos. Por exemplo, a linguagem contém características que

dependem do suporte de execução (*runtime*) e a biblioteca contém software que fornece uma interface para as facilidades no ambiente de execução.

5.3.1 A LINGUAGEM DE PROGRAMAÇÃO JAVA

Segundo Comer (2001), Quando um programador cria um documento ativo, o programador escreve o programa em uma linguagem fonte. Em vez de basear em uma linguagem existente, a tecnologia Java define uma nova linguagem de programação que tenta tornar conveniente a escrita e a leitura de programas fontes para *applets*. Formalmente a linguagem é chamada de Linguagem de Programação Java; informalmente, os programadores utilizam freqüentemente a abreviação Java.

5.3.1.1 CARACTERISTICA DA LINGUAGEM

Segundo Comer (2001), a linguagem Java pode ser caracterizada como:

a) alto nível;
b) propósito geral;
c) similar ao C++;
d) orientadas a objetos;
e) dinâmica;
f) fortemente tipada;
g) verificação de tipo estática;
h) concorrente.

5.3.2 O AMBIENTE DE EXECUÇÃO (RUNTIME) JAVA

Segundo Comer (2001), a tecnologia Java define um ambiente de execução sobre o qual programas Java rodam. O ambiente de execução é caracterizado por:

a) execução interoperativa;
b) coleta de lixo automática (*Garbage Collection*);
c) execução com múltiplos *threads (Multithreaded)*;
d) acesso à Internet;
e) suporte gráfico.

5.3.3 A BIBLIOTECA JAVA

Segundo Comer (2001), o terceiro componente da tecnologia Java é uma biblioteca de definições de classes. A biblioteca é extensa - contém dezenas de definições de classes com aproximadamente 2000 métodos individuais. Mais importante, a biblioteca contém classes que abrangem uma variedade de necessidades. Por exemplo, a biblioteca contém classes para:

a) manipulação gráfica;
b) E/S de rede de baixo nível;
c) interação com um servidor da web;
d) acesso ao sistema de execução;
e) E/S de arquivo;
f) estruturas de dados convencionais;
g) captura de eventos;
h) tratamento de exceções.

5.4 JBUILDER 4 ENTERPRISE

Segundo Borland (2001), o JBuilder é o único ambiente de produtividade empresarial que suporta desenvolvimento compatível com EJB 1.1 nas plataformas Windows, Linux e Solaris. Sua versão acelera de forma significativa o tempo de desenvolvimento, permitindo a rápida distribuição de aplicações para *e-business* nos servidores *Java 2 Enterprise Edition.*

O JBuilder 4 é integrado ao *Inprise Application Server* e agora inclui também suporte ao servidor WebLogic, oferecendo às empresas uma solução completa, flexível e escalável para a computação e-business.

Os usuários podem rodar e depurar de forma transparente aplicações EJBs e CORBA, local ou remotamente no ambiente JBuilder.

O JBuilder 4 inclui ainda um novo ambiente de desenvolvimento em equipe, expansível e integrado, além de um sistema de controle de versão, de um ambiente de desenvolvimento integrado IDE e de melhor desempenho.

6 DESENVOLVIMENTO DO PROTÓTIPO

A descrição das etapas do desenvolvimento do protótipo são apresentadas a seguir.

6.1 LEVANTAMENTO DAS INFORMAÇÕES

Observa-se que em uma rede de computadores, se trabalham com diversos tipos de softwares, como de acesso a banco de dados, acesso a Internet, acesso a serviço de e-mail entre outros.

Este protótipo pretende gerenciar o desempenho de um servidor conectado a uma rede local. Esta máquina é responsável por alguns serviços da rede como: autenticação de usuários no domínio, gerenciamento DHCP, gerenciamento WINS, servidor de arquivos e Banco de dados, entre outros.

Para fim de protótipo, escolheu-se algumas variáveis relacionadas à gerência de desempenho e se calculou algumas taxas referentes à performance da rede.

As variáveis são as seguintes:

a) Porcentagem de entrada de dados para a interface: esta porcentagem se refere a quantidade de bits por segundo, que entraram na controladora e que foram encaminhados para a mesma.

   Pode-se obter esta taxa utilizando-se das seguintes variáveis:

   - ifInOctets: número de bytes recebidos;
   - ifSpeed: capacidade máxima de transmissão da interface.

   Assim pode-se calcular esta taxa, utilizando os seguintes passos:

   1) Calcula-se o total de bytes recebidos em um intervalo de tempo entre x e y;
      - Total de bytes = (ifInOctets_y - ifInOctets_x);
   2) Calcula-se o total de bytes por segundo;
      - Total de bytes por segundo = total de bytes/(y-x);
   3) Calcula-se o total de bits por segundo;
      - Total de bits por segundo = total de bytes por segundo * 8;
   4) Finalmente, tem-se a taxa de entrada;
      - Porcentagem de entrada = (total de bits por segundo) / ifSpeed.

b) Porcentagem de saída de dados para a interface: esta porcentagem refere-se a quantidade de bits por segundo que saíram da controladora.

Pode-se obter esta taxa utilizando-se das seguintes variáveis:

- ifOutOctets: número de bytes enviados;
- ifSpeed: capacidade máxima de transmissão da interface.

Assim pode-se calcular esta taxa, utilizando os seguintes passos:

1) Calcula-se o total de bytes enviados em um intervalo de tempo entre x e y;
   - Total de bytes = (ifOutOctets_y - ifOutOctets_x);
2) Calcula-se o total de bytes por segundo;
   - Total de bytes por segundo = total de bytes/(y-x);
3) Calcula-se o total de bits por segundo;
   - Total de bits por segundo = total de bytes por segundo * 8;
4) Finalmente, tem-se a taxa de utilização;
   - Taxa de saída = (total de bits por segundo) / ifSpeed.

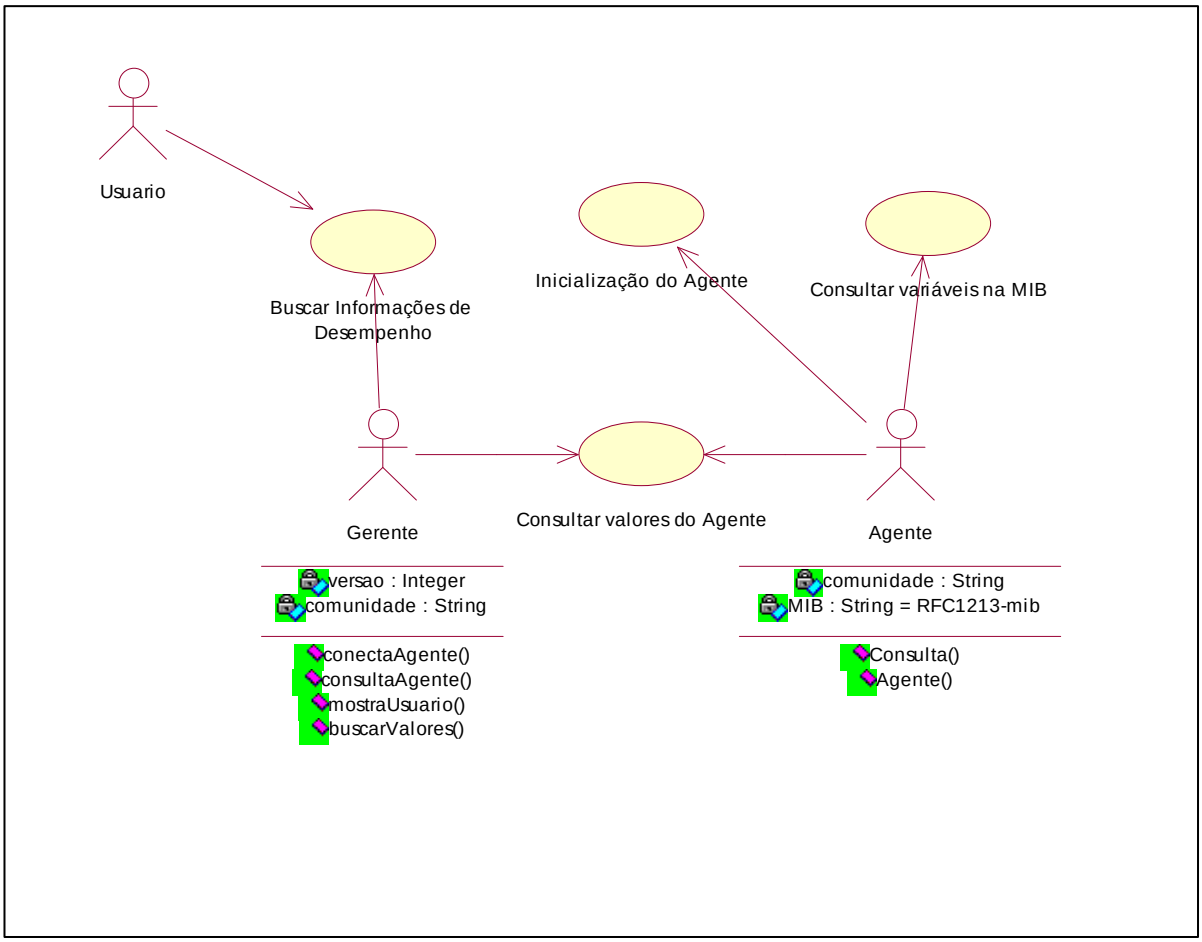
6.2 ESPECIFICAÇÃO

O protótipo desenvolvido propõe controlar algumas variáveis da MIB do protocolo SNMP relacionada a funções de desempenho da rede.

6.2.1 DIAGRAMA DE CASOS DE USO

Conforme notação da UML, e utilizando a ferramenta Rational Rose, foi desenvolvido o diagrama de casos de uso conforme a figura 12, onde destacam-se quatro casos de uso principais do sistema.

Figura 12 - Diagrama de casos de uso



Os casos de uso definidos para o sistema são:

a) buscar informações de desempenho: o usuário acessa a aplicação gerente e solicita determinada informação relacionada a desempenho;

b) inicialização do agente: conjunto de ações necessárias para o funcionamento do agente;

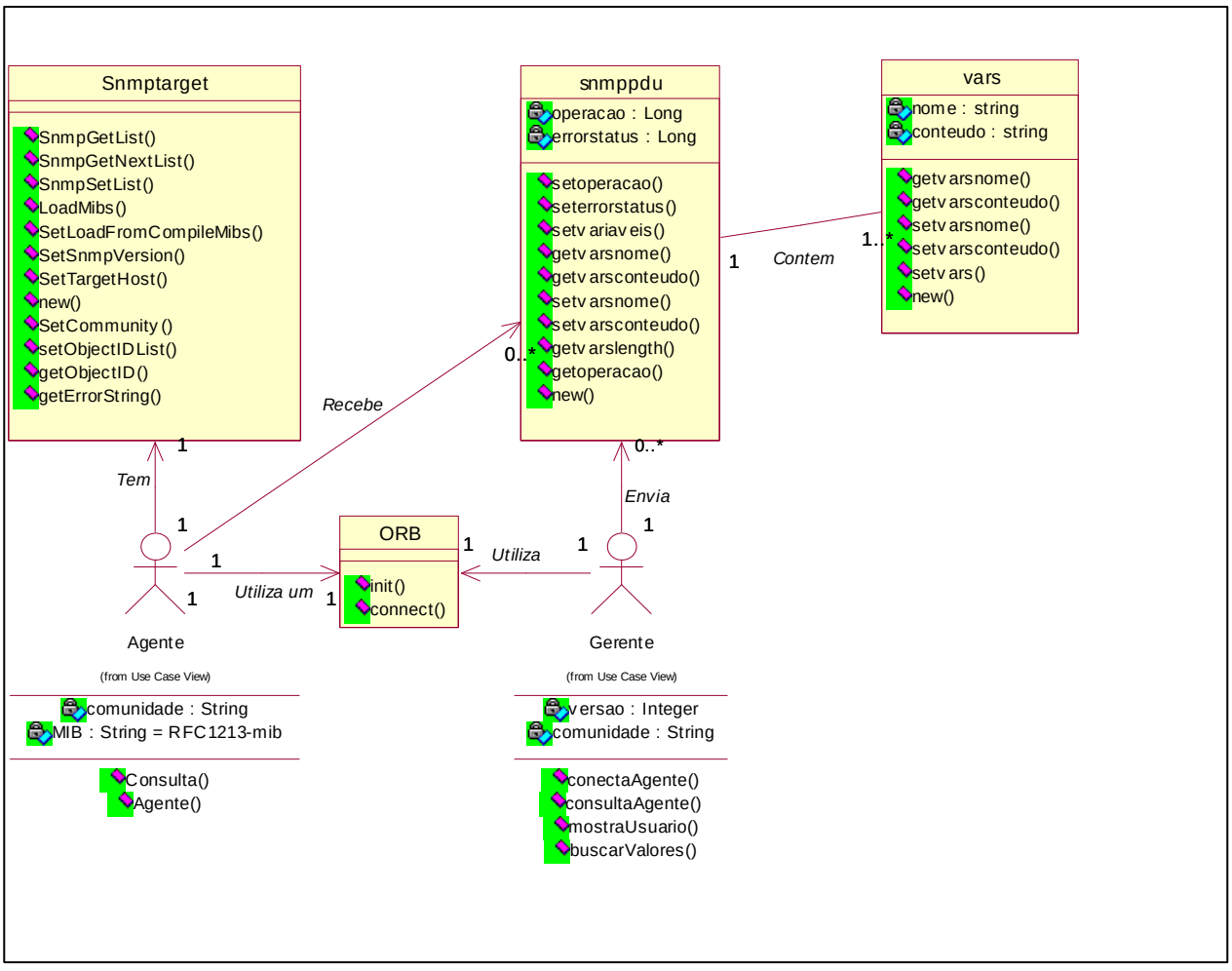
c) consultar variáveis da MIB: quando um agente recebe uma mensagem do gerente, é executada esta solicitação na MIB, podendo ser um get ou um getnext da variável definida;

d) consultar valores do agente: quando o gerente precisa buscar uma informação de um agente, ele informa qual variável ele precisa e o agente retorna como resultado este valor.

6.2.2 DIAGRAMA DE CLASSES

Inicialmente, na fase de análise, identificou-se as classes principais do sistema, seus atributos e relacionamentos, conforme figura 13.

Figura 13 - Diagrama de classes



Foram definidas as classes Agente e Gerente que vão executar os métodos para a funcionalidade do protótipo. A classe agente somente executa dois métodos que são: a inicialização do agente e a consulta as variáveis da MIB. A classe gerente executa alguns métodos para conectar ao agente e consulta as informações.

As classes snmppdu e vars foram definidas para que se possa fazer a comunicação de informações entre agente e gerente, devido ao uso de um ORB para esta comunicação. Através da classe vars são definidas as variáveis da MIB a serem consultas contendo seu

nome e seu conteúdo. E através da classe snmppdu são informadas a operação a ser executada e um controle sobre erros durante a consulta.

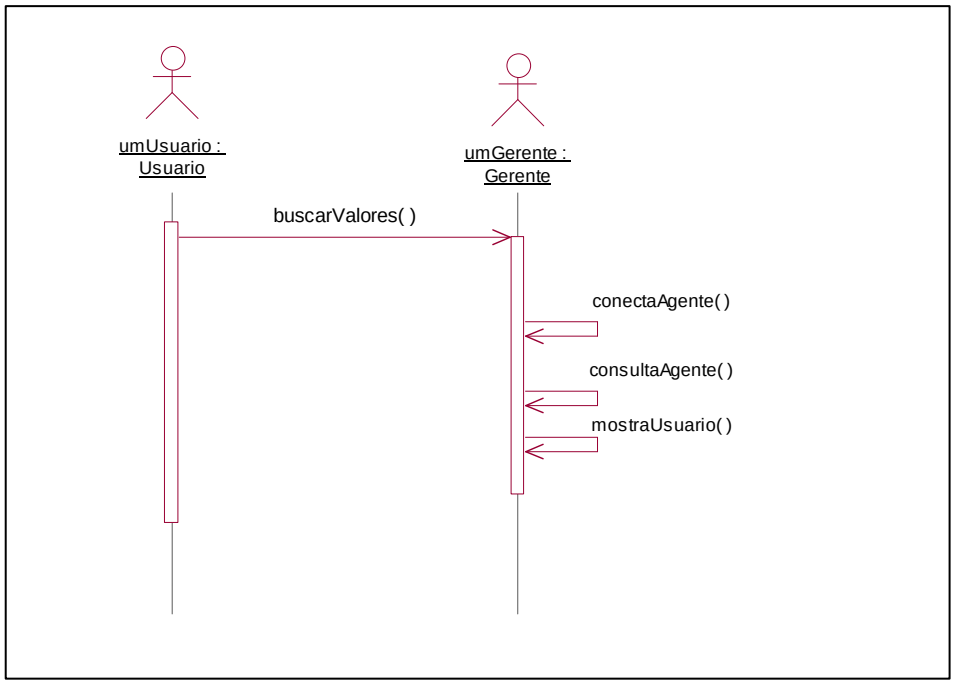
A classe ORB, é uma classes que vem com o *Java SDK Standard Edition* da *Sun*, portanto esta classe não foi desenvolvida neste trabalho, mas somente utilizada. Da mesma forma a classe Snmptarget que vem incluída no *AdvenetSNMP2vc* da *Adventnet Inc*.

6.2.3 DIAGRAMA DE SEQÜÊNCIA

Nos diagramas de seqüência pode-se observar a execução de cada caso de uso e verificar as mensagens trocadas entre os objetos. Os principais diagramas de seqüência encontrados no sistema são:

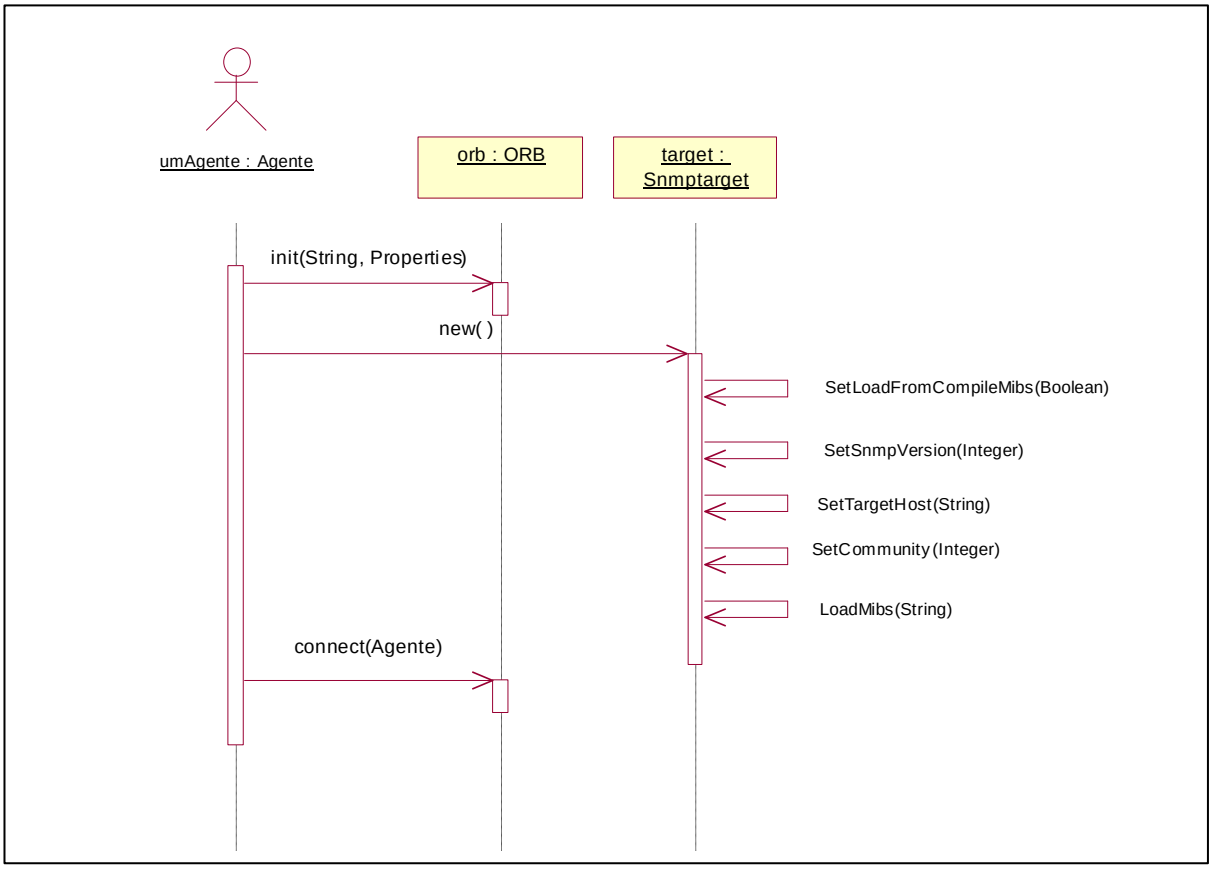
a) buscar informações de desempenho: o usuário utiliza a aplicação gerente para ter acesso às informações e utilizando a interface da aplicação as solicita. Após a solicitação, a aplicação gerente vai se conectar a aplicação agente, fazer a consulta referente as variáveis a serem buscadas e mostrar ao usuário. Esta situação pode ser visualizada pela figura 14;

Figura 14 - Buscar informações de desempenho



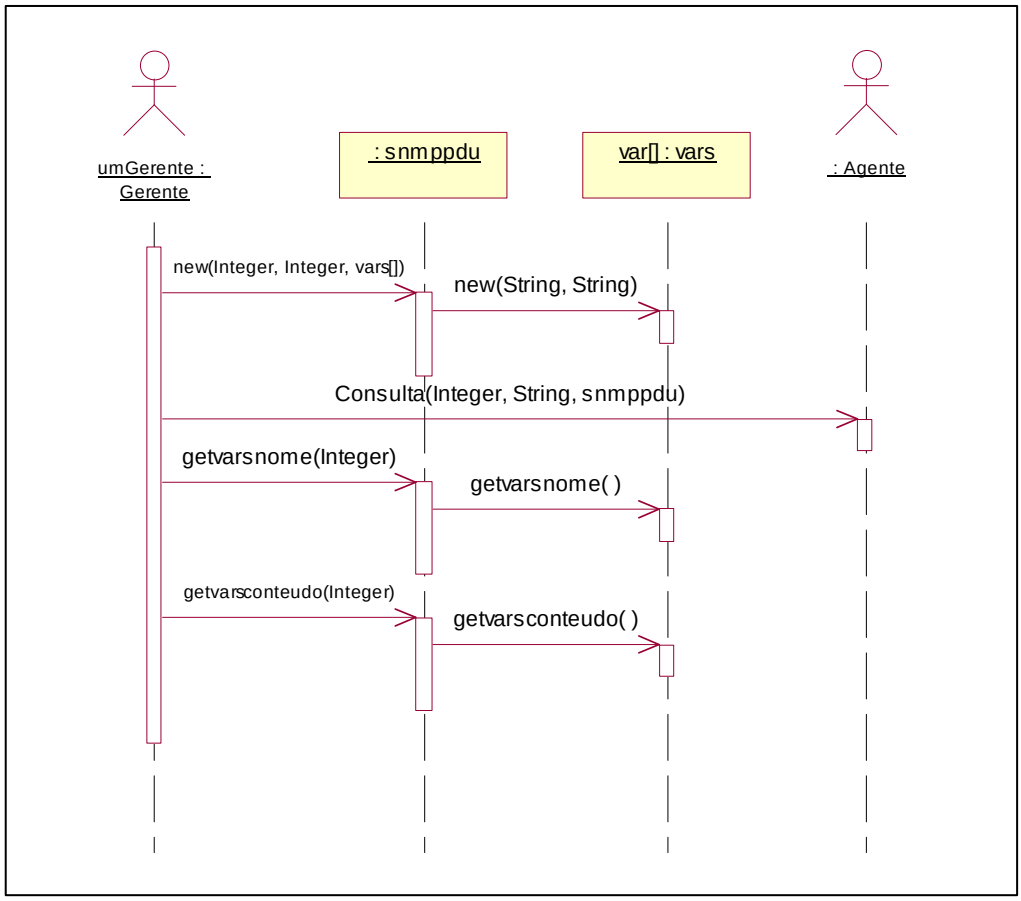
b) inicialização do agente: a aplicação agente inicializa o ORB.A aplicação instância um objeto para que se trabalhe com as variáveis do SNMP. Após isso, deve-se informar ao software que ele deve carregar uma MIB compilada, também é informada a versão do protocolo a ser trabalhada, o endereço ip da máquina que será buscada as informações, a comunidade que este agente pertence e carregada-se a MIB. Finalmente a aplicação agente se conecta ao ORB e fica aguardando uma chamada da aplicação gerente (figura 15);

Figura 15 - Inicialização do agente.



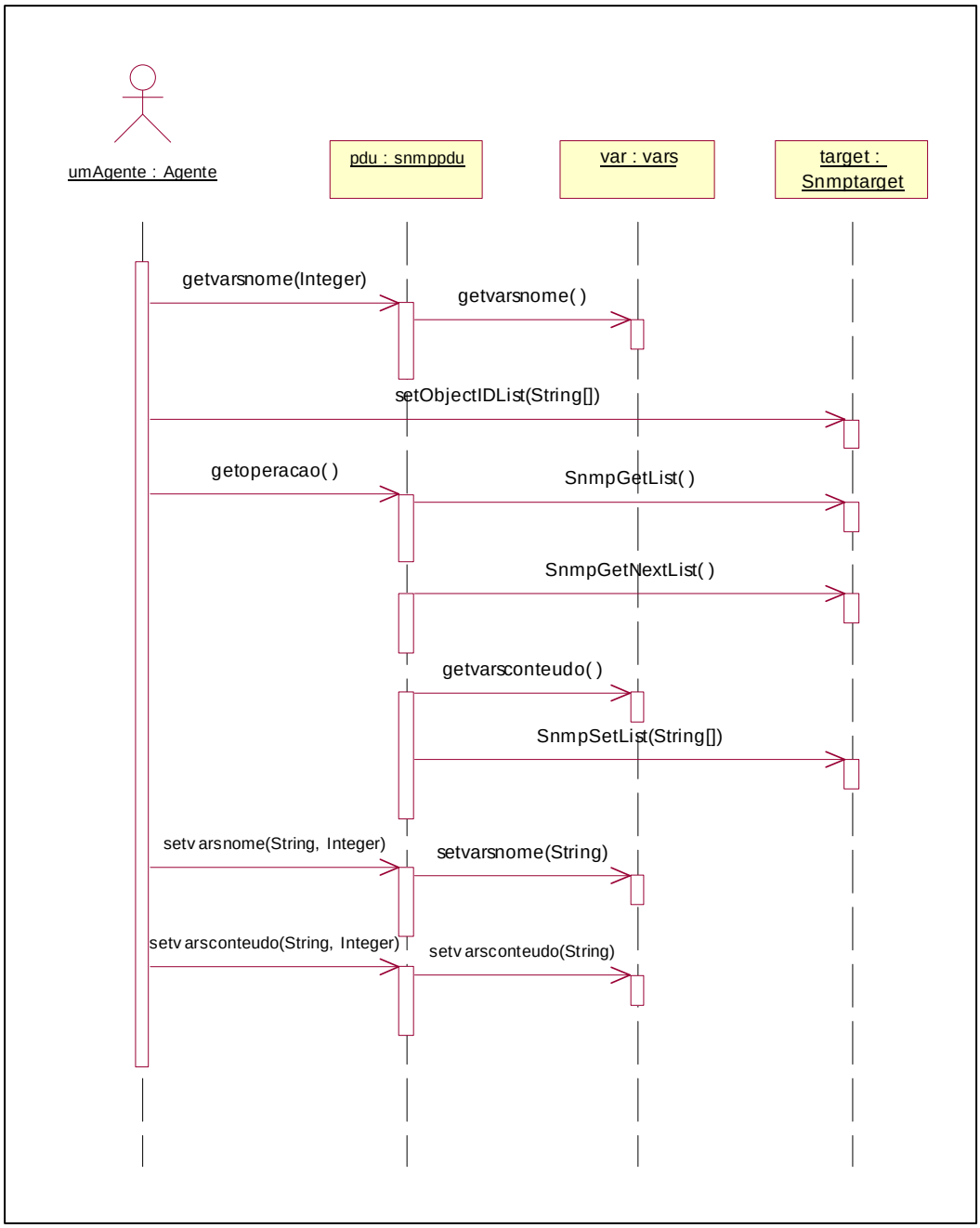
c) consultar variáveis do agente: a aplicação gerente recebe uma solicitação de consulta ao agente. Ao receber esta solicitação, o gerente monta uma pdu contendo a(s) variável (is) e a operação a ser feita. Após isso é chamado o método consulta que busca o valor destas variáveis no agente e finalmente a aplicação gerente consulta o nome e o valor da(s) variável (is) (figura 16).

Figura 16 - Consultar variáveis do agente



d) consultar variáveis da MIB: ao receber uma solicitação de consulta da aplicação gerente, a aplicação agente busca o nome da variável e verifica qual a operação solicitada. Ao verificar a operação, o agente a realiza e após trazer o resultado, é informado o nome da variável e o seu conteúdo a aplicação gerente (figura 17).

Figura 17 - Consultar variáveis da MIB



6.3 IMPLEMENTAÇÃO

O protótipo desenvolvido neste Trabalho de Conclusão de Curso visa uma análise de dados obtidas a partir de agentes sobre segmentos monitorados da rede, visualizados a partir de um gerente, no âmbito do gerenciamento de redes, sendo que a comunicação entre os agentes e o gerente acontece através do protocolo SNMP para troca de dados e fornecimento de informações.

Para a implementação do protótipo proposto foi utilizado o grupo de classes *AdventNetSNMPv2c* da empresa *AdventNet Inc.*, que possibilita o uso do protocolo SNMP através de suas classes. Ainda para a implementação foi utilizado o ambiente de programação *JBuilder 4.0* da empresa *Borland Corporation*, que se baseia na linguagem de programação Java. Para a visualização do conteúdo das MIBs foi utilizado o programa *MIB Browser* incluído no grupo de classes *AdventNetSNMPv2c*.

Inicialmente foi definido um arquivo de interface que executará a comunicação entre aplicações agente e gerente utilizando o CORBA. Através da linguagem de definição de interface foi definida a interface do quadro 1.

Quadro 1 - Arquivo snmpcorba.idl

```
module snmpcorba
{
        /* estrutura para interface de variáveis da MIB */
        struct vars
        {
         string nome;
         string conteudo;
        };

        /* tipo sequência criado para informar arrays de tamanhos diferentes */
        typedef sequence<vars> varsseq;

        /* estrutura para interface das operações do SNMP
        struct snmppdu
        {
         long operacao;
         long errorstatus;
         varsseq variaveis;
        };

        interface agente
        {
         /* método que serve de interface entre agente e gerente */
         snmppdu consulta(in long versao, in string comunidade, in snmppdu pdu);
         };
};
```

Nesta interface IDL, foi criada uma *struct* chamada vars, onde estarão definidas os nomes e o conteúdo das variáveis da MIB. Foi criado um tipo chamado varseq, que será um *array* da *struct* vars onde poderá ser informada em tempo de criação dos objetos o tamanho deste *array*. Também foi criada uma *struct* chamada snmppdu onde estão definidos três atributos: a operação a ser realizada, um código de erro e as variáveis a serem passadas para o agente. Finalmente foi criada a interface agente, onde foi definido o método consulta, onde efetivamente é feita a comunicação entre o agente e gerente. Neste método são passados como parâmetros, a versão do protocolo SNMP, a comunidade a qual pertence o gerente e a pdu onde estão as informações sobre a consulta. Como retorno deste método é trazido uma pdu com os valores solicitados.

Definida a interface entre os objetos agente e gerente, o arquivo acima especificado é submetido a um compilador *idlj* incluso no pacote *Java SDK Standard Edition* da *Sun*. Este compilador realiza o mapeamento desta interface definida pela linguagem de definição de interface para a linguagem de programação Java Após submeter o arquivo geram-se os seguintes arquivos:

a) _agenteImplBase.java;
b) _agenteStub.java;
c) agenteOperations.java;
d) agente.java;
e) agenteHelper.java;
f) agenteHolder.java;
g) snmppdu.java;
h) snmppduHelper.java;
i) snmppduHolder.java;
j) vars.java;
k) varsHelper.java;
l) varsHolder.java;
m) varsseqHelper.java;
n) varsseqHolder.java.

Estes arquivos contem funcionalidades para a utilização do CORBA em ambas as aplicações. Estão inclusos dentro de uma *package* chamada snmpcorba gerada pelo compilador *idlj*.

Para a implementação do aplicativo agente, foi definida uma classe chamada agente que herda características da classe _agenteImplBase, sendo que esta classe define os métodos declarados na interface IDL e implementa as funcionalidades do CORBA. Abaixo os quadros 2, 3 e 4 referentes as principais rotinas do aplicativo agente.

Quadro 2 - Constructor da classe

```
public Agente(List ListLog) {
   this.ListLog = ListLog;
   //Instancia-se o objeto target
   target = new SnmpTarget();
   //informa-se ao objeto que ele deve carregar uma MIB compilada
   target.setLoadFromCompiledMibs(true);
   try {
     //informa-se a versão do protocolo SNMP
     target.setSnmpVersion(SnmpTarget.VERSION1);
   }
   catch(Exception e) {
        ListLog.add("Erro informando versão do protocolo!!");
   }
   //informa-se o host a ser buscada a informação
   target.setTargetHost("127.0.0.1");
   //informa-se a comunidade
   target.setCommunity(COMUNIDADE);
   try {
     //carrega-se a MIB
     target.loadMibs(MIB);
   }
   catch(Exception e) {
     ListLog.add("Erro informando versão da MIB!!");
   }
 }//fim constructor
```

No quadro 2, é mostrado a *constructor* da classe Agente, onde são informados dados como: a versão do protocolo a ser utilizado, o ip da máquina onde é buscado os dados ,o nome da comunidade e onde é carregada a MIB.

Quadro 3 - Inicialização do agente

```
public void inicializaAgente() {
     try {
       // Recupera uma referência para o ORB e inicializa
       Properties props = new Properties();
       props.put("org.omg.CORBA.ORBInitialPort", "1050");
       String[] args ={ };
       orb = ORB.init(args, props);
       // Cria o objeto servidor e registra-o no ORB
       orb.connect(this);
       // Recupera o servidor de nomes
       org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
       NamingContext ncRef = NamingContextHelper.narrow(objRef);
       // Registra a referência do objeto no servidor de nomes
       NameComponent nc = new NameComponent("Agente SNMP", "");
       NameComponent path[] = {nc};
       ncRef.rebind(path, this);
       ListLog.add("Agente Inicializado..");
     }//fim try
     catch(Exception exp) {
       ListLog.add("Erro: " + exp);
       exp.printStackTrace(System.out);
     }//fim catch
 }//fim inicializaAgente
```

No quadro 3, é mostrado o método chamado inicializaAgente, onde basicamente é feita a rotina de inicialização e conexão da aplicação ao ORB. Primeiramente é informado a porta pela qual o ORB vai se comunicar, sendo neste caso a porta 1050. Após isso é recuperado o servidor de nomes, sendo que este é um dos serviços que o CORBA oferece. Finalmente é registrado a referencia deste objeto no servidor de nomes, sendo que o nome deste objeto é “Agente SNMP”.

Quadro 4 - Consulta variáveis da MIB

```
public snmppdu consulta(int versao,String comunidade,snmppdu pdu) {
   if (versao!= target.getSnmpVersion()) { // verifica-se a versão do SNMP
      ListLog.add("Versao do protocolo inválido!!!");
   }
   else {
     if (!target.getCommunity().equals(comunidade)) { // verifica-se a comunidade
      ListLog.add("Solicitação não pertence a esta comunidade!!!");
     }
     else {
       String oids[] = new String[pdu.getvarslength()]; // busca-se o nome das variáveis
       String result[] = new String[pdu.getvarslength()];
       for (int i=0;i <= pdu.getvarslength()-1;i++) {
         oids[i] = pdu.getvarsnome(i);
       }
       target.setObjectIDList(oids);
       switch (pdu.getoperacao()) { // 0 – GET / 1 – GETNEXT / 2 - SET
         case 0:
              result = target.snmpGetList(); // chama método que realiza o GET
              if (result == null) {
                ListLog.add("Erro durante solicitação de GET"+ target.getErrorString());
              }
              break;
         case 1:
              result = target.snmpGetNextList();// chama método que realiza o GETNEXT
              if (result == null) {
                ListLog.add("Erro durante solicitação de GETNEXT"+ target.getErrorString());
              }
               break;
         case 2:
              ListLog.add("Chegando solicitação de SET"); // busca-se os valores a serem
               String variables[]=new String[pdu.getvarslength()]; // informados
               for (int i=0;i <= pdu.getvarslength()-1;i++) {
                 variables[i] = pdu.getvarsconteudo(i);
              }
              try {
                 result = target.snmpSetList(variables); // chama método que realiza o SET
                 if (result == null) {
                   ListLog.add("Erro durante solicitação de SET"+ target.getErrorString());
                 }
              }//fim try
              catch (Exception e) {
                 ListLog.add("Set Error: "+e.getMessage());
              }
              break;
       }//fim switch
       if (result ==null) { // caso não retorna valor, retorna as variáveis com vazio
         for (int i=0; i<oids.length; i++) { // e informa-se o código do erro
           pdu.setvarsnome(" ",i);
           pdu.setvarsconteudo(" ",i);
         }
         pdu.seterrorstatus(target.getErrorCode());
       }//fim if
       else {
         for (int i=0; i<oids.length; i++) { // senão retorna o valor das variáveis
           pdu.setvarsnome(target.getObjectID(i),i);
           pdu.setvarsconteudo(result[i],i);
         }
         pdu.seterrorstatus(0);
       }//fim else
     }//fim else
   }//fim else
   return pdu;
 }//fim consulta
```

No quadro 4, é mostrada toda a rotina de consulta de uma variável na MIB. Inicialmente, é verificado a versão do protocolo SNMP e a comunidade a qual pertence o gerente que solicitou a informação. Logo após, são colocadas os nomes das variáveis a serem consultadas em uma variável do tipo array e chamado um método para que se informe ao objeto target estas variáveis. É verificada qual a operação a ser feita e chamado o método referente do objeto target. Caso seja feita uma operação de SET, ainda é buscado o conteúdo das variáveis e passadas como parâmetro no método respectivo. Finalmente, caso não ocorra erros, são informados os nomes e respectivos conteúdos das variáveis consultadas.

Para a aplicação gerente, foi criada uma classe chamada Gerente, que implementa todas as funcionalidades do gerente, incluindo a montagem das PDU's, a conexão ao agente entre outras. Abaixo os quadros 5, 6 e 7 referentes as principais funções do gerente.

Quadro 5- Conexão com o agente

```
public boolean conectaAgente(String host) {
  try {
    Properties props = new Properties();
    props.put("org.omg.CORBA.ORBInitialHost",host);
    props.put("org.omg.CORBA.ORBInitialPort","1050");
    String[] args ={ };
    orb = ORB.init(args,props);
    // Recupera o servidor de nomes
    org.omg.CORBA.Object objRef = orb.resolve_initial_references ("NameService");
    NamingContext ncRef = NamingContextHelper.narrow(objRef);
    // Recupera a referência para o objeto no servidor de nomes,
    // de acordo com o caminho fornecido
    NameComponent nc = new NameComponent("Agente SNMP", "");
    NameComponent path[] = {nc};
    Agt = agenteHelper.narrow(ncRef.resolve(path));
    // Chama operações do objeto remoto
    org.omg.CORBA.BooleanHolder result = new org.omg.CORBA.BooleanHolder();
    ListLog.add("Conectado a "+ host +"...");
    return true;
  }//fim try
  catch(Exception e) {
    ListLog.add("Erro: " + e);
    return false;
  }//fim catch
}//fim conectaAgente
```

Neste método, é informado um nome da máquina onde está o agente e é feita a conexão ao ORB. Após isto, é recuperada a referência do objeto no servidor de nomes, onde

se informa o nome registrado no servidor de nomes. Finalmente é chamado o método narrow do agente onde finalmente é realizada a conexão.

Quadro 6 - Consulta ao agente

```
private vars[] consultaAgente(vars[] variaveis) {
    pdu = new snmppdu((int)GET,(int)0,variaveis);
    pdu = Agt.consulta(VERSAO,COMUNIDADE,pdu);
    return pdu.getvariaveis();
  }
```

Este método recebe como parâmetro um array de objetos da classe vars e onde é criado o objeto pdu, onde é informado a operação e código do erro e as variáveis informadas. É chamado o método consulta existente na aplicação agente, onde são passados como parâmetros, a versão do protocolo SNMP, a comunidade a qual pertence o agente e a pdu criada. Finalmente o retorno desta função é o array de variáveis informadas com seus respectivos conteúdos.

Quadro 7 – Cálculo das porcentagens de entrada e saída

```
public void calculaTaxas(int taxa,long nrInter,int Tempo) {
    switch (taxa) {
      case 0:
        var2[0] = new vars("ifInOctets."+nrInter," ");
        var2[1] = new vars("ifSpeed."+nrInter,"  ");
        var2 = consultaAgente(var2,GET);
        float txent;
        try {
          long ifInOctets = Long.parseLong(var2[0].getvarsconteudo());
          long totbytes = ifInOctets - vifInOctets;
          vifInOctets = ifInOctets;
          long totbytessec = totbytes / Tempo;
          long totbitssec = totbytessec * 8;
          LMbitsEntrada.setText(Float.toString((float)totbitssec/1048576));
          txent = (float) totbitssec / Long.parseLong(var2[1].getvarsconteudo());
        }
        catch (Exception ArithmeticException) {
          txent = (float)0;
        }
        Ltxent.setText(Float.toString(txent*100));
        break;
      case 1:
        var2[0] = new vars("ifOutOctets."+nrInter," ");
        var2[1] = new vars("ifSpeed."+nrInter,"  ");
        var2 = consultaAgente(var2,GET);
        float txsai;
        try {
          long ifOutOctets = Long.parseLong(var2[0].getvarsconteudo());
          long totbytes = ifOutOctets - vifOutOctets;
          vifOutOctets = ifOutOctets;
          long totbytessec = totbytes / Tempo;
          long totbitssec = totbytessec * 8;
          LMbitsSaida.setText(Float.toString((float)totbitssec/1048576));
          txsai = (float) totbitssec / Long.parseLong(var2[1].getvarsconteudo());
        }
        catch (Exception ArithmeticException) {
          txsai = (float)0;
        }
        Ltxsai.setText(Float.toString(txsai*100));
        break;
    }//fim switch
  }//fim calculaTaxas
```

Neste método, é calculada as porcentagens de entrada e saída de dados da interface, assim é passado como parâmetro a taxa a ser calculada, a interface a ser buscada e o tempo de atualização. Inicialmente é verificado qual a porcentagem a ser calculada, em seguida, buscado o conteúdo das variáveis no agente e realizado o cálculo e finalmente mostrado ao usuário.

6.3.1 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Para a execução dos aplicativos agente e gerente são necessários que seja instalado o serviço SNMP nas estações a serem gerenciadas. Também se faz necessário à instalação do *Java Runtime Enviroment*, que é provê um suporte completo para execução de aplicações desenvolvidas em Java e para a utilização do ORB incluído neste pacote.

A implementação do agente pode ser vista através da figura 18, onde é mostrado a tela do aplicativo agente.

Figura 18 - Aplicativo agente

Para a execução do aplicativo agente, primeiro deve-se executar o ORB digitando o seguinte comando no Prompt do MS-DOS:

```
tnameserv –ORBInitialPort 1050
```

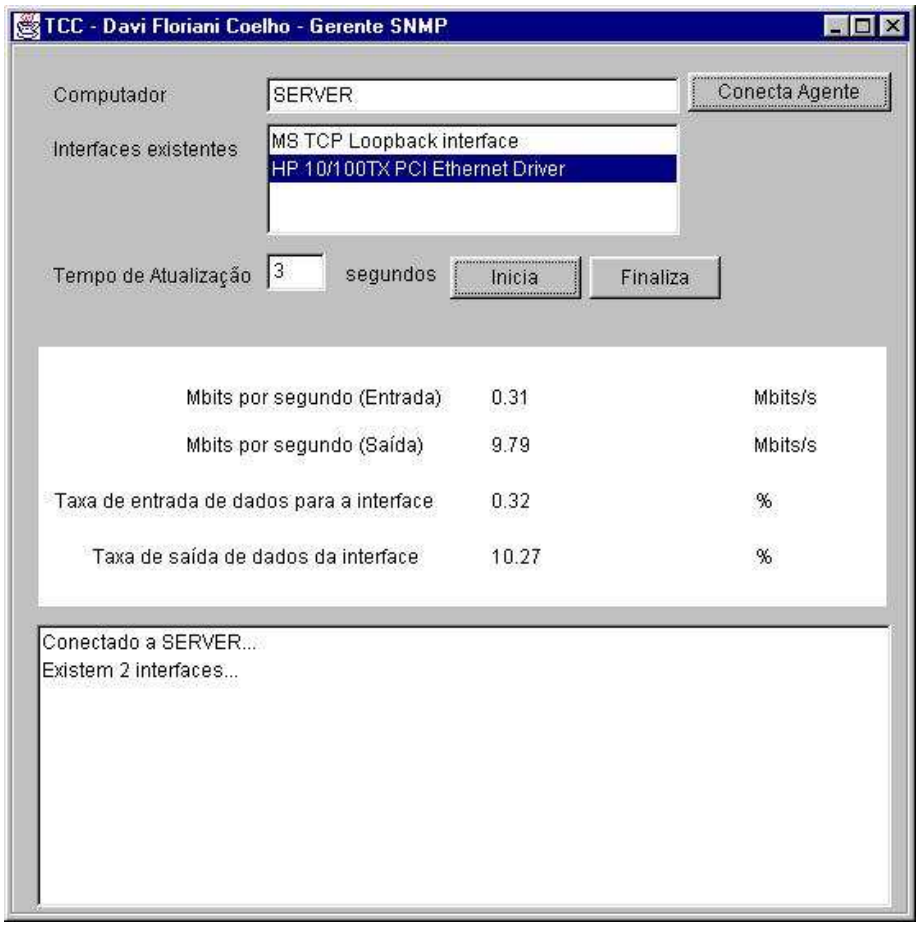
Inicialmente são mostradas algumas informações referentes ao computador onde está instanciado o agente, sendo que estas informações são buscadas da MIB deste agente.

O aplicativo agente é inicializado apertando o botão Inicializa Agente visualizado na interface gráfica do aplicativo. Ao aparecer à mensagem Agente Inicializado, o aplicativo agente está pronto para receber solicitações de consulta do aplicativo gerente. A interface gráfica deste aplicativo é simples, pois ela só demostrará algumas mensagens.

Ao receber uma solicitação de consulta, será mostrada ao usuário pela interface gráfica do agente conforme a figura 18.

A implementação do gerente pode ser vista através da figura 19, onde é mostrado a tela do aplicativo gerente.

Figura 19 - Aplicativo gerente



O aplicativo gerente é mostrado, onde inicialmente deve ser informado o nome do computador a ser buscada as informações de desempenho, digita-se o nome do servidor da rede, neste caso o nome “SERVER” e aperta-se o botão “Conecta Agente”. Ao ser realizada a

conexão, é mostrada a mensagem “Conectado a SERVER”, logo abaixo e também aparece abaixo do nome da máquina as interfaces existentes nela. Seleciona-se a interface na qual se quer ter as porcentagens e informa-se o tempo de atualização das informações e aperta-se o botão “Inicia”. Assim, vão aparecendo na tela as porcentagens de entrada e saída em relação à capacidade da interface consultada. Também aparece a quantidade de bits recebidos e transmitidos na interface em Mbits/s. Assim pode-se ter uma idéia da quantidade de bits trafegados no servidor da rede.

7 CONCLUSÕES

Atualmente, pode-se concluir que a gerência de desempenho está relacionada ao planejamento e avaliação das redes de computadores, assim deve-se definir métricas e valores aos recursos da rede para que se possa reconhecer situações indicativas de degradação de desempenho e também para prever possíveis atualizações de recursos da rede.

Uma das contribuições deste trabalho é a seleção de um conjunto de variáveis da MIB II que podem ser utilizadas com qualquer software genérico de gerência baseado em SNMP para implementação da gerência de desempenho.

Porém, devido o SNMP não suportar a consulta de informações de tráfego a nível de protocolo, por exemplo, o número de pacotes descartados pela controladora, se fosse possível utilizar esta informação, poderia-se através de uma estação da rede, monitorar todo o tráfego da mesma. Assim fica restringida a verificação de desempenho de uma rede de computadores, pois para conseguir esta informação, no mínimo deveriam existir agentes instalados em todos os computadores da rede, monitorando o tráfego de entrada e saída.

Com a utilização da MIB-II no desenvolvimento deste trabalho, constatou-se que a mesma não consegue gerenciar informações sobre atraso e *throughput*, uma vez que não existem variáveis que armazenem dados referentes a estas informações.

Com as experiências adquiridas neste trabalho, pode-se definir um mecanismo de gerência em que os objetos gerenciados, ORBs, aplicações de gerência e agentes são implementados em Java. Com isto, obtém-se um mecanismo de gerenciamento e instanciação de objetos portável para qualquer plataforma que possua um interpretador Java instalado.

Também pode-se analisar que a aplicação agente poderia ser comparada a um *proxy*, pois outras aplicações que utilizem o CORBA e caso necessitassem informações sobre o SNMP poderiam utilizá-la de maneira transparente, bastaria apenas que se utilizasse a interface IDL definida neste trabalho.

Assim, sem a utilização do grupo de classes *AdventNetSNMPv2c*, e o ambiente *JBuilder*, não seria possível implementar este protótipo em tempo hábil, pois ambos oferecem uma rapidez no desenvolvimento destas aplicações.

7.1 LIMITAÇÕES E PROBLEMAS

Uma das limitações constatadas neste protótipo é que utilizando computadores com o sistema operacional Windows 98, o serviço que disponibiliza o gerenciamento SNMP para a rede, só implementa a versão 1 do protocolo SNMP. Computadores que possuem os sistemas operacionais Windows NT, nas versões *Server* e *Workstation* e Windows 2000 na versão *Server* e *Professional* não apresentam esta limitação.

Quando se chama o método inicializaAgente a aplicação vai se conectar ao ORB através das classes definidas, gera-se um certo *overhead* inicial que faz com que a aplicação pareça travada.

7.2 EXTENSÕES

Como sugestão para extensão deste trabalho, pode-se optar por desenvolver aplicações em outras áreas da gerência de redes, como as de configuração, de falhas, de contabilização e de segurança, utilizando o SNMP.

Ainda pode-se buscar saber a causa do *overhead* inicial causado pela conexão da aplicação agente ao ORB, e trazer algum tipo de solução a este problema,

Seria interessante como sugestão, a definição de uma extensão da MIB que gerencie informações referentes a atraso e *throughput* e a criação de um agente que possa alimentar esta MIB.

REFERÊNCIAS BIBLIOGRÁFICAS

BORLAND INTERNATIONAL INC. **Borland disponibiliza o novo JBuilder 4 no Brasil**, São Paulo, Disponível em: <http://www.borland.com.br/releases/release167.htm>. Acesso em: 04-jun-2001.

BRISA, Sociedade Brasileira para Interconexão de Sistemas Abertos. **Gerenciamento de redes – uma abordagem de sistemas abertos**. São Paulo: Makron Books, 1993.

CAPELETTO, Johni J. **Comunicação entre objetos distribuídos utilizando a tecnologia CORBA**. 1999. 60 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

COMER, Douglas E. **Redes de computadores e Internet**. 2. ed. Tradução Marinho Barcellos. Porto Alegre: Bookman, 2001. 522p.

COMER, Douglas E.; STEVENS, David L. **Interligação em rede com TCP/IP**. Vol.2. Rio de Janeiro: Campus, 1999.

FURLAN, José D. **Modelagem de objetos através da UML**. São Paulo: Makron Books,1998. 329p.

KLAUCK, Hugo A. **Gerência de redes ATM utilizando CORBA e SNMP**. 1999. 51 f. Trabalho Individual (Mestrado em Ciências da Computação) - Departamento de Informática e de Estatística, Universidade Federal de Santa Catarina, Florianópolis.

KRUPSKAIA, Karla; BITAR, Leôncio. **Objetos distribuídos: um estudo sobre CORBA e DCOM**, Belém, jan. 2000. Disponível em: <http://planeta.terra.com.br/informatica/krups/>. Acesso em: 03 fev. 2001.

MELLO, Jorge L. **Protótipo de agente SNMP para uma rede local utilizando a plataforma JDMK**. 2000. 87 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

OTSUKA, Joice. **Management Information Base,** Porto Alegre, 1995. Disponível em: <http://penta.ufrgs.br/gr952/trab1/2capa.html>. Acesso em 14 fev. 2001.

SPECIALSKI, Elizabeth. **Gerência de redes de computadores e telecomunicações**. 1999. 51 f. Notas de aula - Departamento de Informática e de Estatística, Universidade Federal de Santa Catarina, Florianópolis.

ANEXOS

VARIÁVEIS DA MIB RELACIONADAS À GERÊNCIA DE DESEMPENHO

GRUPO INTERFACES

Objeto	Informação usada no gerenciamento de desempenho
ifInDiscards	número de pacotes com descartes de entrada
ifOutDiscards	número de pacotes com descartes de saída
ifInErrors	número de pacotes com de erros de entrada
ifOutErros	número de pacotes com erros de saída
ifInOctets	número de bytes recebidos pela interface
ifOutOctets	número de bytes enviados pela interface
ifInUcastPkts	número de pacotes unicast recebidos
ifOutUcastPkts	número de pacotes unicast enviados
iIfInNUcastPkts	número de pacotes no-unicast recebidos
ifOutNucastPkts	número de pacotes no-unicast enviados
ifInUnknownProtos	número de pacotes de protocolos desconhecidos recebidos
ifOutQLen	total de pacotes na fila de saída

FONTE: Otsuka (1995)

GRUPO IP

Objeto	Informação usada para gerenciamento de desempenho
ipInReceives	número de datagramas de recebidos
ipInHdrErrors	número de datagramas com erros de cabeçalho de entrada
ipInAddrErrors	número de datagramas com erros de endereço de entrada
ipForwDatagrams	número de datagramas repassados
ipInUnknownProtos	número de datagramas de entrada para um protocolo desconhecido
ipInDiscards	número de datagramas de entrada descartados
ipInDelivers	número de datagramas de entrada entregues com sucesso
ipOutRequests	número de datagramas de saída (não inclui os datagramas repassados)
ipOutDiscards	número de datagramas de saída descartados
ipOutNoRoutes	número de datagramas descartados ocorridos por falta de informação de roteamento
ipRoutingDiscards	número de entradas de roteamento descartadas
ipReasmReqds	número de fragmentos recebidos necessitando de remontagem
ipReasmOKs	número de datagramas com sucesso na remontagem
ipReasmFails	número de datagramas com falhas na remontagem
ipFragOKs	número de datagramas com sucesso na fragmentação
ipFragFails	número de datagramas com insucesso na fragmentação
ipFragCreates	número de fragmentos gerados

FONTE: Otsuka (1995)

GRUPO ICMP

Objeto	Informação usada para gerenciamento de desempenho
icmpInMsgs	número total de mensagens ICMP recebidas
icmpInErrors	número de mensagens ICMP recebidas com erros
icmpInDestUnreachs	número de mensagens de destino não alcançado recebidas
icmpInTimeExcds	número de mensagens de tempo excedido recebidas
icmpInParmProbs	número de mensagens de problemas com parâmetros recebidas
icmpInSrcQuenchs	número de mensagens de fonte apagada recebidas
icmpInRedirects	número de mensagens de redirecionamento recebidas
icmpInEchos	número de mensagens de ecos (requisição) recebidas
icmpInEchoReps	número de mensagens de respostas a ecos recebidas
icmpInTimestamps	número de mensagens de requisição de Timestamp recebidas
icmpInTimestampReps	número de mensagens de respostas ao pedido de Timestamp recebidas
icmpInAddrMasks	número de mensagens de requisição de máscaras de endereços recebidas
icmpInAddrMaskReps	número de mensagens de resposta a máscaras de endereços recebidas
icmpOutMsgs	número total de mensagens ICMP enviadas
icmpOutErrors	número total de mensagens ICMP não enviadas
icmpOutDestUnreachs	número de mensagens de destino não alcançado enviadas
icmpOutTimeExcds	número de mensagens de tempo excedido enviadas
icmpOutParmProbs	número de mensagens de problema com parâmetros enviadas
icmpOutSrcQuenchs	número de mensagens de origem apagada enviadas
icmpOutRedirects	número de mensagens de redirecionamento enviadas
icmpOutEchos	número de mensagens de eco (requisição) enviadas
icmpOutEchoReps	número de mensagens respostas de eco enviadas
icmpOutTimestamps	número de mensagens de requisição de Timestamp enviadas
icmpOutTimestampReps	número de mensagens de respostas ao pedido de Timestamp enviadas
icmpOutAddrMasks	número de mensagens de requisição de máscara de endereços enviadas
icmpOutAddrMaskReps	número de mensagens de resposta de máscara de endereços enviadas

FONTE: Otsuka (1995)

GRUPO TCP

Objeto	Informação usada para gerenciamento de desempenho
tcpAttemptFails	número de tentativas de conexão falhadas
tcpEstsabResets	número de reinicializações de conexões estabelecidas
tcpRetransSegs	número de segmentos retransmitidos
tcpInErrs	número de pacotes recebidos com erro
tcpOutRsts	número de vezes que a entidade tentou reinicializar uma conexão
tcpInSegs	número total de segmentos TCP recebidos
tcpOutSegs	número total de segmentos TCP enviados

FONTE: Otsuka (1995)

GRUPO UDP

Objeto	Informação usada para gerenciamento de desempenho
udpInDatagrams	número de datagramas entregues
udpOutDatagrams	número de datagramas enviados
udpNoPorts	número de datagramas que não foram enviados para uma porta valida
udpInErrors	número de datagramas UDP recebidos com erro

FONTE: Otsuka (1995)

GRUPO EGP

Objeto	Informação usada para gerenciamento de desempenho
egpInMsgs	número de mensagens EGP recebidas
egpInErrors	número de mensagens recebidas com erro
egpOutMsgs	número de mensagens EGP enviadas
egpOutErrors	número de mensagens EGP não enviadas devido a ocorrência de erros
egpNeighInMsgs	número de mensagens recebidas deste vizinho EGP
egpNeighInErrs	número de mensagens recebidas com erro deste vizinho EGP
egpNeighOutMsgs	número de mensagens enviadas a este vizinho EGP
egpNeighOutErrs	número de mensagens não enviadas a este vizinho EGP devido a erros
egpNeighInErrMsgs	número de mensagens de erro recebidas deste vizinho EGP
egpNeighOutErrMsgs	número de mensagens de erro enviadas a este vizinho EGP

FONTE: Otsuka (1995)

GRUPO SNMP

Objeto	Informação usada para gerenciamento de desempenho
snmpInPkts	número de mensagens SNMP recebidos
snmpOutPkts	número de mensagens SNMP enviados
snmpInTotalReqVars	número de get/get-next-requests recebidas
snmpInTotalSetVars	número de set-requests recebidas
snmpInGetRequests	número de get-requests recebidas
snmpInGetNexts	número de get-next-requests recebidas
snmpInSetRequests	número de set-requests recebidas
snmpInGetResponses	número de get-responses recebidas
snmpInTraps	número de traps recebidas
snmpOutGetRequests	número de get-requests enviadas
snmpOutGetNexts	número de get-next-requests enviadas
snmpOutSetRequests	número de set-requests enviadas
snmpOutGetResponses	número de get-responses enviadas
snmpOutTraps	número de traps enviadas

FONTE: Otsuka (1995)